VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

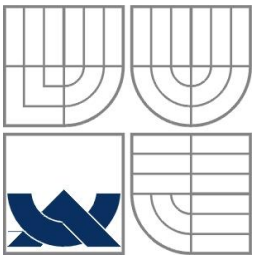
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

INFORMAČNÍ SYSTÉM TECHNICKÉ PODPORY

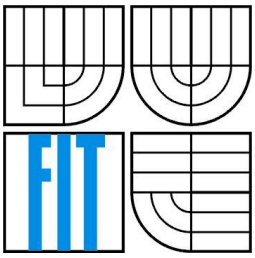
DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE Bc. Petr Boháček
AUTHOR

BRNO 2007



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

INFORMAČNÍ SYSTÉM TECHNICKÉ PODPORY

INFORMATION SYSTEM OF TECHNICAL SUPPORT

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — Bc. Petr Boháček
AUTHOR

VEDOUCÍ PRÁCE — Ing. Šárka Květoňová
SUPERVISOR

BRNO 2007

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav informačních systémů Akademický rok 2006/2007

Zadání diplomové práce

Řešitel: **Boháček Petr, Bc.**
Obor: Informační systémy
Téma: **Informační systém technické podpory**
Kategorie: Web

Pokyny:

1. Seznamte se s požadavky kladenými na tvorbu dynamických webových stránek.
2. Podrobně analyzujte požadavky na systém technické podpory zvolené firmy zabývající se výrobou software. Zaměřte se zejména na vhodnou formu vyhodnocování dat o prováděných činnostech zaměstnanců.
3. Seznamte se s databázovým systémem MySQL a skriptovacími jazyky HTML, PHP, JavaScript a s možnostmi jejich využití při implementaci systému.
4. Systém realizujte a jeho funkčnost ověřte na vhodném vzorku testovacích dat.
5. Zhodnoťte dosažené výsledky a diskutujte další možný rozvoj systému.

Literatura:

- Williams, H. E., Lane, D.: PHP a MySQL. - Vytváříme webové databázové aplikace. Computer Press, 2002, 552 s. ISBN 8072267604
- Welling, L., Thomsonová, L.: PHP a MySQL - rozvoj webových aplikací. Softpress, 2003, 720 s. ISBN 8086497607
- Kosek, J.: HTML, tvorba dokonalých www stránek. Praha: Grada Publishing, 1998, 291 s. ISBN 80-7169-608-0
- PHP: Hypertext Preprocessor. Dostupné na: www.php.net
- DeLisle, M.: PHPMyAdmin - efektivní správa MySQL. Brno: Zoner Press, 270 s. ISBN 8086815099

Při obhajobě semestrální části diplomového projektu je požadováno:

- Body 1-3.

Podrobné závazné pokyny pro vypracování diplomové práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva diplomové práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap, které byly vyřešeny v rámci ročníkového a semestrálního projektu (30 až 40% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním paměťovém médiu (disketa, CD-ROM), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Květoňová Šárka, Ing.**, UIFS FIT VUT
Datum zadání: 28. února 2006
Datum odevzdání: 22. května 2007

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav informačních systémů
612 66 Brno, Božetěchova 2

L.S.

doc. Ing. Jaroslav Zendulka, CSc.
vedoucí ústavu

Licenční smlouva

Licenční smlouva je uložena v archivu Fakulty informačních technologií Vysokého učení technického v Brně.

Abstrakt

Tato diplomová práce se zabývá analýzou požadavků na informační systém technické podpory společnosti vyvíjející software. Hlavními funkcemi systému jsou sběr informací o pracích prováděných zaměstnanci firmy a možnost generování statistik. Systém však umožňuje integraci rozšíření, která pomohou integrovat další funkce, přímo nesouvisející s původním posláním systému, či definici dalších uživatelských rolí, které budou mít přístup k různým částem systému.

Klíčová slova

Technická podpora, testování softwaru, operativní plánování, taktické plánování, návrh IS, životní cyklus IS, modelování IS, RUP.

Abstract

This master's thesis deals with the analysis of the requirements placed on the technical support information system which should be used in the software developing company. The basic functions of this system are the collecting of information about all kinds of work which the company employees concentrate on and the possibility of statistics generating. Moreover, it is also possible to implement further extensions to the system which will not only enable other functions, that are not related to the primary goal of the system, to be integrated; but also to define new user roles, for which various limited parts of the system will be accessible.

Keywords

Technical support, software testing, operative planning, tactic planning, IS design, IS life cycles, IS modeling, RUP.

Citace

Petr Boháček: Informační systém technické podpory, diplomová práce, Brno, FIT VUT v Brně, 2007

Informační systém technické podpory

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením Ing. Šárky Květoňové. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Petr Boháček
16.4.2007

Poděkování

Děkuji vedoucí diplomové práce, Ing. Šárce Květoňové za její čas a cenné rady, které mi věnovala na konzultacích. Děkuji své rodině a partnerce Bc. Janě Pěgřímové za podporu a trpělivost, kterou se mnou měli při psaní této práce a v průběhu celého studia.

Obsah

1 Úvod

Informační systémy jsou v naší Informační době nejen nástrojem vyšší efektivity práce, ale stávají se nezbytnou nutností pro každou firmu a instituci, která chce efektivněji řídit své zdroje. Práce s informacemi (jejich sbírání, třídění, sumarizování a dolování v nich) může být základem rozvoje firmy a často přináší konkurenční výhody na trhu. Podnikový informační systém je především databází historických informací, které mohou být kdykoliv použity, a nástrojem managerů pro lepší rozhodování o budoucnosti firmy a plnění firemních cílů.

Informační systém technické podpory, který je předmětem této diplomové práce, má tedy především uspokojovat potřeby podniku, který je zastoupen jeho vedením. Systém by měl také být vstřícný k uživatelům, měl by uchovávat všechny hodnotné informace a v neposlední řadě by měl poskytovat přidanou hodnotu oproti přežitým papírovým agendám, které nahrazuje. Výhodami jsou poté nejen ekonomičtější a ekologičtější jednání související například se schvalováním dovolených, ale i informace používané k získání jiných obecně netriviálních informací. Propracované systémy mohou například zjišťovat korelaci mezi délkou pracovního poměru a výkonem zaměstnance. Takové netriviální vztahy by bez počítačové podpory bylo možné zpracovávat velmi těžko a dá se říci, že bez informačních systémů by ekonomické hledisko dolování z dat značně předčilo získané informace.

Tato diplomová práce se zabývá především problematikou ukládání informací o pracích přidělených zaměstnancům Technologické podpory firmy Grisoft, s. r. o. a jejich následným využitím k efektivnější dělbě práce a vytvoření detailnějšího výkazu výkonů na pracovišti. Ve druhé kapitole zmíním vybrané partie z teorie analýzy informačních systémů obsahující také představení různých životních cyklů vývoje softwaru. Zvláštní pozornost v této kapitole budu věnovat metodice RUP. V rámci téže kapitoly popíši i jedno z možných dělení podnikových informačních systémů a paradoxy při hodnoceních jejich efektivity. Ve třetí kapitole blíže představím Technologickou podporu a popíši návrh implementace systému od rolí až po akce, které bude systém svým uživatelům umožňovat. Ve čtvrté kapitole nastíním další možnosti informačního systému, které mohou být součástí informační strategie podniku a prostředí, v němž bude systém nasazen. V páté kapitole zmíním základy testovacího procesu, který jsem navrhl a implementoval v rámci své bakalářské práce [8]. Tento proces je také možným budoucím rozšířením popisovaného systému pro podporu testování.

2 Informační systémy

Vývoj softwarových informačních systémů spadá do kategorie vývoje softwaru, který se vždy podřizuje životnímu cyklu určeného *softwarovým procesem*. Softwarový proces je po částech uspořádaná množina kroků směřujících k vytvoření nebo úpravě softwarového díla. Úroveň vyspělosti procesu (Capability Maturity Model) lze určit například podle pětibodové stupnice SEI (Software Engineering Institute). Její jednotlivé úrovně jsou:

1. Počáteční (Initial) úroveň znamená stav, kdy firma nemá definován softwarový proces a každý projekt je řešen případ od případu (ad hoc).
2. Opakovatelná (Repeatable) úroveň je stav, v němž firma definovala v jednotlivých projektech opakovatelné postupy. Tyto postupy je schopna znovu používat v dalších projektech.
3. Definovaná (Defined) úroveň popisuje softwarový proces na základě integrace dříve identifikovaných a opakovatelných kroků.
4. Řízená (Managed) úroveň se vyznačuje schopností firmy řídit a monitorovat definovaný proces.
5. Optimalizovaná (Optimised) úroveň navíc získává dlouhodobým monitorováním softwarového procesu zpětnovazební informaci, která je následně využita pro jeho optimalizaci.

Během vývoje softwarových produktů vznikly tradiční vývojové cykly, mezi něž patří především modely Vodopád, Spirála, či RUP. V následujícím textu přiblížím hlavní modely těchto životních cyklů. V poslední kapitole zmíním dělení a podstatu jiného pohledu na informační systémy.

2.1 Tradiční modely životních cyklů

Mezi tradiční modely životních cyklů patří zejména referenční model Vodopád, který byl jemně modifikován a vylepšen modely Výzkumník a Prototypování. Mezi modely, které představily jako součást svého vývojového cyklu iterace, lze dále zařadit Inkrementální model a Spirálový životní model. V následujícím textu popíši tyto základní metody vývoje, jejich přednosti a nedostatky a v další kapitole představím podrobněji metodiku RUP, která je jedním z nejlepších současných procesů vývoje.

Životní model Vodopád

První ucelenou metodikou vývoje softwaru je Vodopádový model. Jeho charakteristickým znakem jsou po sobě jdoucí fáze bez iterací, mezi kterými probíhá schvalovací proces nutný k postupu projektu do další fáze.

Základní fáze Vodopádového modelu jsou:

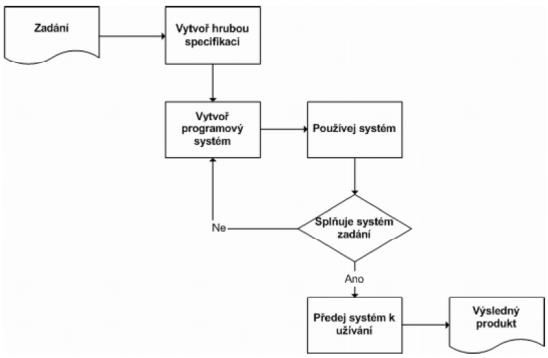
1. Specifikace problému, sběr požadavků a definice problémové domény.
2. Specifikace a analýza požadavků.
3. Návrh.
4. Implementace.
5. Testování a předání zákazníkovi.
6. Provoz a údržba.

Mezi přímo sousedícími fázemi je možné se pohybovat v obou směrech právě o jednu fázi. Při přechodu do jakékoliv fáze, a to i když se vracíme podruhé, je nutné znovu přepracovat a schválit generované dokumenty. Teprve až se projekt dostane do fáze údržby, je možné vrátit se do jakékoliv jiné fáze a software propracovat. Tento rys systému je také hlavním kamenem úrazu celého modelu. Při striktním dodržení sekvenčnosti fází a omezení, které jsem popsal, není možné dynamicky reflektovat požadavky zákazníka, které přijdou před fází údržby, když se systém začne implementovat. Tato vlastnost systému měla zaručit nekontrolovanou propagaci změn a degeneraci kvality kódu, ovšem v dnešní době je nutné na požadavky zákazníka reagovat co nejrychleji a komunikovat s ním co nejčastěji, což Vodopád neposkytuje. Častá komunikace se zákazníkem v jakékoliv fázi je především krokem k produktu splňujícímu požadavky zadavatele, které nemusí být při analýze správně pochopeny. Naopak nespornými výhodami tohoto modelu jsou jeho jednoduchost a jednoznačnost v definici fází, která do procesu vývoje vnáší řád.

V době, kdy se Vodopádový model objevil, bylo podle něj vytvořeno mnoho softwarových produktů. Dnes se používán spíše jako referenční model pro srovnání s ostatními modely, nebo pro vývoj malých softwarových systémů, u kterých jsou jasné a z větší míry neměnné požadavky.

Model Výzkumník

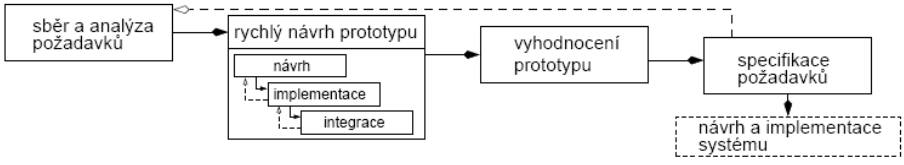
Snaha odstranit zásadní nedostatky Vodopádového modelu vedla k jeho různým modifikacím. Jednou z nich je Model výzkumník (obr. 1), který zavádí inkrementální způsob vývoje aplikace (viz. další strana). Jedná se v podstatě o řadu malých Vodopádů s výrazně kratším životním cyklem, z nichž každý odpovídá nově vzneseným požadavkům. Model Výzkumník vylepšuje Vodopádový model zejména v přístupu k zákazníkovi, se kterým se snaží více komunikovat, což vede zejména k lepšímu pochopení jeho požadavků na systém. I přes tuto jednu dobrou vlastnost je tento model v dnešní době již také překonaný. Model Výzkumník se hodí především pro malé projekty. Hlídání změn v kódu a složité řízení projektu jsou pro něj hlavními limitujícími faktory. Při nedůsledném programování může projekt skončit neúspěchem, protože v rámci iterací degraduje kód, jehož změny se nemusí nutně projevit v návrhových dokumentech systému.



Obr. 1: *Model životního cyklu Výzkumník [11]*

Model Prototypování

Stejně jako u modelu Výzkumník je pozornost i v tomto modelu (obr. 2) soustředěna na zákazníka a jeho požadavky na systém. Cílem tohoto modelu je tedy lepší pochopení zákazníkových požadavků tvorbou prototypu. Prototyp plní dva hlavní účely, z nichž prvním je pochopení problémových požadavků zákazníka a druhým je předání funkční verze části systému ještě před dokončením celého projektu.



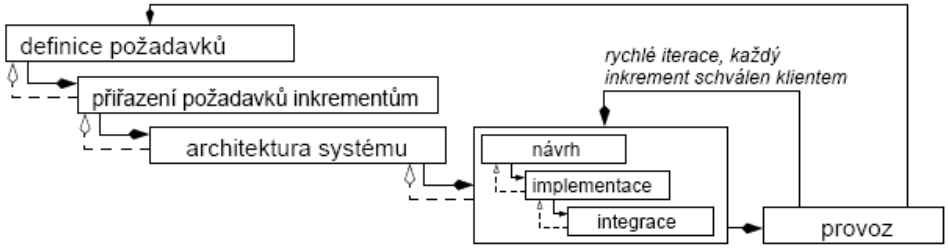
Obr. 2: *Model životního cyklu Prototypování [11]*

Hlavním kladem Prototypovacího životního cyklu je testování a ověřování nekompletních verzí budoucími uživateli, kteří mohou vznášet nové požadavky a zpřesňovat specifikaci systému,

který ještě není zcela vyvinut. Záporem mohou být větší náklady na výsledný produkt, které jsou dány především vyššími nároky na vedení týmu a požadavky na speciální nástroje a techniky.

Inkrementální model vývoje

Byl navržen jako model, který má předcházet přepracovávání částí systému v důsledku změn požadavků. Za přírůstek (inkrement) je potom považována jakákoliv část, která je v procesu přidána mezi následující kroky. Přírůstek se může týkat jak kódu (například v podobě nové funkce), tak i samotného modelu (například modelu případu použití nebo byznys modelu).



Obr. 3: *Model inkrementálního vývoje softwaru [11]*

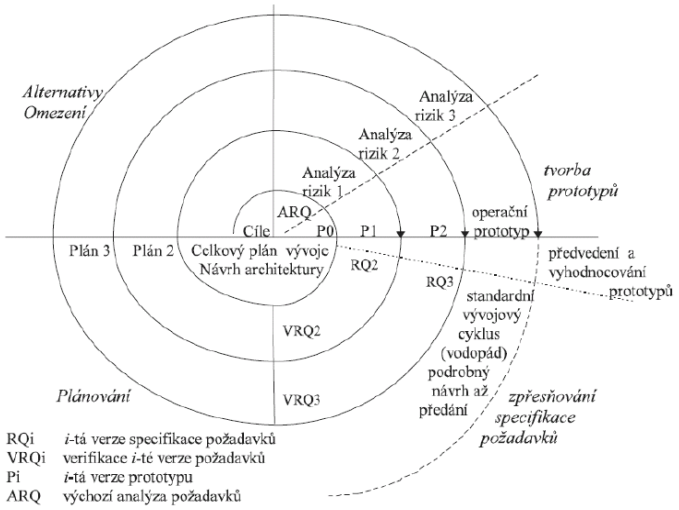
V průběhu procesu jsou napřed definovány přehledové požadavky, které jsou přiřazeny k jednotlivým inkrementům. Poté je navržena architektura systému a vývoj se dostává do fáze vytváření rychlých iterací. Opakovaně jsou pro každý inkrement prováděny akce návrhu, implementace, testování, integrace a validace systému, které mohou být předány zákazníkovi. Model předpokládá, že:

- Na počátku jsou přehledově definovány služby, které zákazník od systému požaduje.
- Zákazník určí priority implementovaných služeb.
- Podle priority je definována množina inkrementů, z nichž každý poskytuje jistou dávku přidané funkcionality.
- Jsou specifikovány požadavky na první inkrement.
- Inkrement je vytvořen nejvhodnějším procesem vývoje, v jehož průběhu nejsou akceptovány změny požadavků.
- Paralelně s vývojem může probíhat analýza požadavků pro následující inkrementy.
- Po dokončení je inkrement předán zákazníkovi, který jednak získá další funkcionalitu systému, jednak může na základě jeho používání vznést další požadavky.

Hlavní klady tohoto přístupu lze spatřovat v rychlosti nasazení nejdůležitějších částí systému, které jsou součástí prvních inkrementů, a menším riziku selhání projektu.

Spirálový životní cyklus

Vodopádový životní cyklus byl brzy po svém vzniku překonán Spirálovým modelem Barryho Boehma (1985). Tento model vychází z Vodopádového modelu, který obohacuje o dvě zásadní vlastnosti. První je iterativní přístup a druhou je podrobná analýza rizik. Proto je také někdy označován za Model řízený riziky (risk-driven approach). Hlavním problémem při sekvenčním vývoji bylo vypořádání se s nově kladenými požadavky. Tento problém lze vyřešit definicí rámce architektury v prvních iteracích a jejich následným propracováním v dalších. Cyklické opakování kroků vývoje, které se nazývá iterační, znamenalo v době vzniku Spirálového modelu přelom v chápání životního cyklu. Analýza rizik je prováděna v každém cyklu a předjímá rozhodnutí o dalším vývoji projektu. Riziko je hlavním pojmem, který znamená jakoukoliv situaci potencionálně ohrožující projekt. Analýza poté každému riziku přiřadí pravděpodobnost jeho vzniku a míru ohrožení úspěšného dokončení projektu. Častá a podrobná analýza rizik má za úkol s dostatečným předstihem odhalit špatný návrh architektury a skryté problémy, které mohou projekt zničit. Hlavními výhodami Spirálového modelu jsou nezávislost na metodice, která se při vývoji použije, a hlídání rizik v rámci každé iterace. Pokud se vývoj začne ubírat špatným směrem, vedoucí vývojových týmů o tom budou brzy uvědoměni a mohou učinit protiopatření (například rozhodnout neimplementovat některé části v dané iteraci, či projekt zastavit).



Obr. 4: *Model spirálového životního cyklu [12]*

Model je dostatečně komplexní, aby vyhověl dnešním požadavkům kladeným na vývoj softwaru větších rozměrů. Model na druhou stranu zcela spoléhá na metodiku a nezmiňuje se o činnostech, které je nutné v jednotlivých fázích provádět.

2.2 Metodiky vývoje RUP

Metodika vývoje software, kterou budu dále rozvádět je v poslední době velice používaná a dala by se označit za nejprogresivnější. Z těchto důvodů ji zde věnuji více prostoru. Rational Unified Process (RUP) je rámcovým procesem iterativního vývoje softwaru vytvářený od roku 2002 firmou Rational Software Corporation (odtud i jeho název). Proces definuje disciplinovaný přístup k přidělování úkolů, zodpovědnost v rámci vývojové organizace a klade si za cíl vytvoření produktu požadované kvality za stanoveného rozpočtu a podle časového rozvrhu. RUP tedy není konkrétním, striktně předepsaným procesem, kterým by se měla adaptující firma řídit, ale je to procesní rámec, který musí být přizpůsoben požadavkům firmy a vývojářským týmům.

Rational Unified Process je také softwarový produkt zabývající se procesy při vývoji, původně vyvinutý firmou Rational Software a nyní spravovaný firmou IBM. Tento produkt obsahuje znalostní bázi příkladů artefaktů a dopodrobna popsané různé typy aktivit, které jsou při vývoji používány. Firma IBM nabízí RUP jako část svého produktu Rational Method Composer (RMC), který slouží k přizpůsobení procesu.

Historie

Kořeny Rational procesu vychází z původního Spirálového modelu životního cyklu Barryho Boehma, který společně s Kenem Hartmanem, jedním z hlavních mozků RUP, pracoval v 80. a 90. letech na Rational Approach. První verze Rational Unified Process 5.0 byla vydána v roce 1998 pod vedením Philippe Kruchtena.

Poslání procesu

Tvůrci a vývojáři procesů se zaměřili na sledování různých charakteristických rysů, které vedly k neúspěchům při vývoji softwaru. Hlavní příčiny neúspěchu viděli:

- v ad hoc správě požadavků zákazníka,
- nejednoznačné a nepřesné komunikaci,
- křehké architektuře (architektura, která neodolá "stresujícím" podmínkám),
- zbytečné složitosti,
- neobjevených nekonzistencích v požadavcích, návrhu, nebo implementaci,
- nedostatku testování,
- subjektivním hodnocení stavu projektu,
- špatném vypořádání se s problémy,
- nekontrolovaném šíření změn a nedostatečné automatizaci.

Výsledkem studie bylo vypracování nejlepších postupů, které byly nazvány Rational Unified Process. Při popisu Procesu bylo použito stejných technik jako při vývoji softwaru. To znamená především objektový popis a používání standardu Unified Modelling Language (UML).

Principy a nejlepší postupy

RUP vychází z množiny postupů softwarového vývoje, které jsou považovány za dobré. Jsou jimi především:

- Iterativní vývoj softwaru,
- správa požadavků,
- využívání již existujících softwarových komponent,
- vizualizace modelu softwarového systému,
- průběžné ověřování kvality softwaru a
- řízení změn.

Iterativní vývoj softwaru

Vývoj systému je jako živý organismus: požadavky se mohou měnit během celého vývoje nehledě na to, že všechny požadavky nemusí být po fázi jejich sběru správně pochopeny. Kvůli těmto a mnoha dalším důvodům není možné vyvinout složité a velké systémy v jednom kroku. Vytváření systému je nutné rozdělit do několika částí, kterým se říká iterace. Iterace pomáhají jak vedoucím projektu lépe sledovat průběh projektu, tak zákazníkovi, kterému může vývojová firma poskytnout částečně funkční systém ještě před jeho plným dokončením. Na konci každé iterace by měl být za ideálních podmínek spustitelný kód, který eliminuje část problémů s projektem a poskytuje lepší podporu od zákazníka, který se může k systému vyjádřit v každé z iterací.

RUP používá iterativní a inkrementální vývoj z těchto důvodů:

- Integrace je prováděna krok za krokem během procesu vývoje a omezuje se tak na několik málo částí.
- Jednodušší implementace je vždy levnější.
- Části systému jsou vyvíjeny a implementovány odděleně a mohou být později snáz znovu použity.
- Správou změn lze zaručit záznam všech nutných a možných změn systému i priorit, ve kterých budou implementovány.
- Problémy jsou identifikovány v raných fázích vývoje a každá iterace nabízí možnost objevit další.
- Opakující se inspekce (návrhu, kódu atd.) neustále vylepšují architekturu systému.

Za jedno z největších pozitiv Procesu lze označit větší zapojení zákazníka, který podává zpětnou vazbu vývojářům a vedoucím projektu v rámci každé iterace. Tímto Proces zabraňuje pozdnímu nasazení systému, které s sebou většinou přináší překročení nákladů na vývoj.

Správa požadavků

RUP správa požadavků se snaží pochopit a správně realizovat zákazníkovi požadavky softwarovým produktem. Pozitiva lze nalézt ve:

- Správném pochopení požadavků, které povede k vytvoření správného produktu.
- Snížení nákladů na údržbu systému dané přítomností všech nezbytných funkcí softwaru.

Správa požadavků sestává z těchto aktivit:

- Analýzy problému zabývající se správným pochopením problémové domény a vytvořením metrik, které dokáží zhodnotit přínos systému pro firmu.
- Pochopením potřeb všech uživatelů, jejichž participace na projektu je klíčová.
- Popisu systému, který znamená především modelování důležitých rysů systému obecnějšími případy použití.
- Správy zaměření vývoje určující prioritu následných prací podle zpětné vazby zákazníka.
- Zpřesnění popisu systému, které zahrnuje detailnější propracování případů použití ve spolupráci s uživateli, mající za cíl vytvořit specifikaci požadavků na software (Software Requirements Specification). Tato specifikace může být použita jako část smlouvy mezi vývojovou firmou a klientem a bude jedním z hlavních východisek při definování akceptačních testů.
- Správy změn požadavků, která určuje, jak bude naloženo s novými požadavky.

Komponentní architektura

Architektura založená na komponentách je snadno rozšiřitelná, intuitivně pochopitelná a především umožňuje opětovné použití částí kódu, což může zlevnit a zrychlit vývoj. Komponenty často souvisejí s objekty, a proto je přirozené používat je v objektově orientovaném programování.

Architektura systému postupně nabývá na významu, jak se systém stává větším a složitějším. RUP se snaží vytvořit co nejpřesnější základ architektury již v prvních iteracích. Tato architektura se poté stává prototypem v prvotních fázích vývoje a v dalších iteracích je laděna až do finální fáze, která je předána uživateli. RUP v této fázi pomáhá s výběrem komponent, které mají být vyvinuty, koupeny, či znovu použity. Tyto komponenty jsou založeny na existujících technologiích, jako je CORBA, COM, či Java EE. V dnešní době již existuje mnoho znovupoužitelných komponent a značná část softwarového průmyslu se snaží této situace využít skládáním produktu z dříve vyrobených komponent.

Vizualizace modelu softwarového produktu

Oproštění se od nejnižších stupňů implementace, které jsou reprezentovány například zdrojovým kódem, a následná grafická reprezentace vyšších stupňů, představují velice efektivní způsob, jak si vytvořit obrázek o komplexním řešení projektu. S použitím grafické reprezentace systému je snazší určit vnitřní vztahy mezi funkcemi systému a zároveň vytvořit střední vrstvu mezi popisem byznys procesů a zdrojovým kódem. Modelování je tedy zobrazením a zjednodušením složitosti systémů a RUP popisuje, kdy se má který model použít. Pro modelování diagramů případů použití, diagramů tříd a dalších objektů lze s výhodou použít Unified Modelling Language (UML).

Ověřování kvality softwaru

Nízká kvalita výsledného produktu bývá hlavním kamenem úrazu softwarových produktů a jednou z příčin neúspěchu projektů a firem. RUP pomáhá při plánování kontrol kvality v celém procesu, která je hlídána každým členem týmu zvlášť (žádný pracovník však není určen pouze ke kontrole kvality). Celý proces se snaží kvalitu hlídat a zaručit ji i v předávaném produktu systémem kvalitativních metrik.

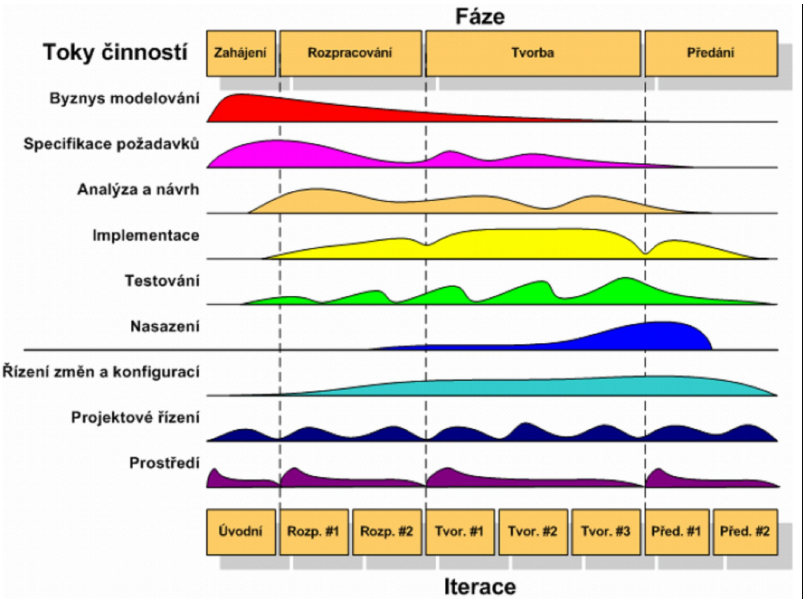
Řízení změn

V každém projektu dochází ke změnám, a proto je potřeba s nimi počítat a myslet na jejich možná řešení. Řízení změn obecně umožňuje zaručit, že všechny změny systému jsou sledovatelné. RUP definuje metody kontroly a sledování změn, které jsou silně spjaty s komponentní architekturou. Neoddělitelnou součástí řízení změn je také vytvoření bezpečného pracovního prostředí poskytujícího maximální možnou ochranu před změnami zvenčí.

Životní cyklus RUP

Proces má jak statickou strukturu, která popisuje toky činností a aktivit, tak dynamickou stránku, která ukazuje, jak je produkt vyvíjen v čase. Životní cyklus vychází ze Spirálového modelu a typický projekt je rozdělen do čtyř fází:

1. Zahájení,
2. rozpracování,
3. tvorba a
4. předání.



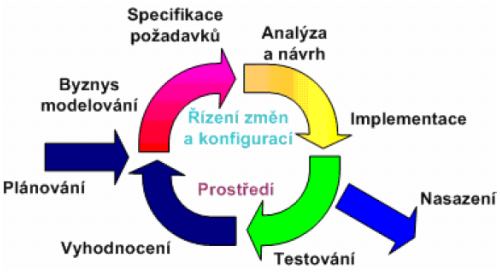
Obr. 5: *Fáze RUP [10]*

Základními rozdíly oproti životnímu modelu Vodopád jsou tedy paralelní provádění činností a plynulé přechody mezi fázemi. První fáze se zabývají především sběrem požadavků a byznys modelováním, kdežto posláním těch posledních je především rozmístění softwarů v počítačové síti a nasazení systému u zákazníka.

Cykly, fáze a iterace Procesu

Každý cyklus by měl vést k vytvoření takové verze systému, která implementuje zadané požadavky a která je připravena k předání uživateli. Jak jsem již uvedl, vývojový cyklus lze rozdělit do čtyř sekvenčních fází, které jsou popsány dále v pořadí provádění. Každou fázi lze dále dělit do iterací. Iterací rozumíme úplnou vývojovou smyčku vedoucí k vytvoření spustitelné verze systému, která reprezentuje podmnožinu vyvíjeného cílového produktu a je postupně rozšiřována každou další iterací až do výsledné podoby. V každé iteraci tedy proběhnou činnosti vázané na byznys

modelování, specifikaci požadavků, analýzu, návrh, implementaci a předání. V rámci těchto činností navíc probíhá řada činností z podpůrných toků týkajících se správy konfigurací, řízení projektu a přípravy prostředí, ve kterém bude systém nasazen.



Obr. 6: *Iterace vývoje produktu [10]*

Fáze zahájení (Inception)

V této fázi jsou vypracovány byznys modely, faktory úspěchu (očekávané příjmy, průzkumy trhu atd.) a finanční analýza (očekávaná návratnost). Dále jsou vytvořeny základní případy použití, projektový plán, počáteční analýza rizik a popis projektu (nejdůležitější požadavky na systém, omezení a klíčové funkce). Poté je možné ověřit tato kritéria:

- Výpočet nákladů na systém a doby dodání prvních dílčích výsledků (či celého systému) a identifikace osob ze strany zákazníka, které se budou podílet na podpoře různých částí systému (informačně, materiálně atd.).
- Pochopení požadavků zákazníka zaznamenaných v případech použití.
- Správné určení nákladů, časů dodání, priorit, rizik a vývojového procesu.
- Hloubku a šíři architektury prototypů.
- Porovnání současných a celkových nákladů.

Pokud projekt nemůže projít tímto milníkem, který se nazývá *Milníkem cílů* (Lifecycle Objective Milestone), může být ukončen nebo může dojít k zopakování fáze po provedení nutných úprav, které budou kritériím lépe vyhovovat.

Fáze rozpracování (Elaboration)

V této fázi začne dostávat projekt reálnější podobu vytvořením analýzy problémové domény a vytvořením základů architektury systému.

Tato fáze musí projít *Milníkem architektury* obsahujícím tato kritéria:

- V diagramech případů použití jsou správně definováni aktéři a případy použití. Model případů použití by měl být hotov nejméně z 80% (je vytvořena velká většina případů použití).

- Je definována architektura, která bude použita při vývoji.
- Je vytvořena spustitelná architektura, která realizuje nejdůležitější případy použití.
- Jsou ověřeny byznys případy a seznam rizik.
- Je vytvořen plán vývoje celého projektu.

Dokud projekt neprojde tímto milníkem, je možné jej stále zrušit nebo přepsat. Po překročení Fáze rozpracování se projekt dostává do velmi nebezpečné fáze, ve které je každá změna velice drahá a složitá.

Fáze tvorby (Construction)

V této fázi je kladen hlavní důraz na vývoj komponent a dalších funkcí systému, což znamená především programování. Ve větších projektech je realizováno i více konstrukčních iterací, které mají za cíl vytvořit funkční prototypy zvolených případů použití. Tyto prototypy mohou sloužit k demonstraci funkcí systému nebo přímo jako části, které je možné nasadit u zákazníka.

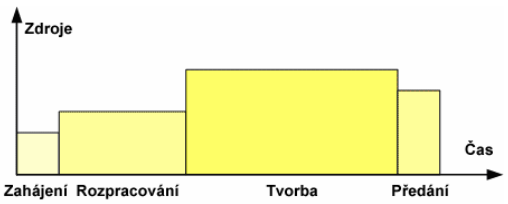
Fáze předání (Transition)

V této fázi je projekt předán z vývoje koncovému uživateli. Součástí této fáze jsou školení uživatelů, údržby systému a beta testování (testování uživateli, nejčastěji v prostředí, kde bude systém používán). Dále je ověřeno, zda produkt splňuje kvalitativní požadavky, které byly definovány ve Fázi zahájení. Pokud je produkt nesplňuje, zopakuje se v této fázi celý cyklus. Pokud jsou všechny požadavky zadavatelů splněny, je dosaženo *Milníku vydání produktu* (Product Release Milestone) a vývojová fáze končí.

Statická struktura procesu

Součástí každého procesu je stanovení:

- aktérů, kteří v něm vystupují, neboli odpovědi na otázku „kdo?“.
- „Co“ má výsledný produkt tvořit.
- „Jak“ to má vytvořit a
- „kdy“ to má vytvořit.



Obr. 7: *Graf vynaloženého času a zdrojů na jednotlivé fáze*

Z tohoto pohledu lze hovořit o následujících elementech, které jsou součástí struktury každého softwarového procesu:

- Role definující chování, zodpovědnosti a kompetence osob zainteresovaných na projektu (analytik, programátor, projektový vedoucí atd.). Jednotlivci poté zastávají při vývoji jisté role, které jim přiřazují kompetence a povinnosti.
- Artefakty reprezentují entity, které jsou v procesu vytvářeny, modifikovány nebo užívány. Jedná se například o modely, dokumentaci, či zdrojový kód. Základním artefaktem, na kterém stojí celý systém, je model systému.
- Aktivity (činnosti) prováděné zaměstnanci s cílem vytvořit či upravit artefakty (kompilace zdrojového kódu, vytvoření dalšího případu použití atd.).
- Toky (workflow) činností reprezentují posloupnosti aktivit, které vytvářejí požadované produkty (specifikace požadavků, byznys modely apod.).

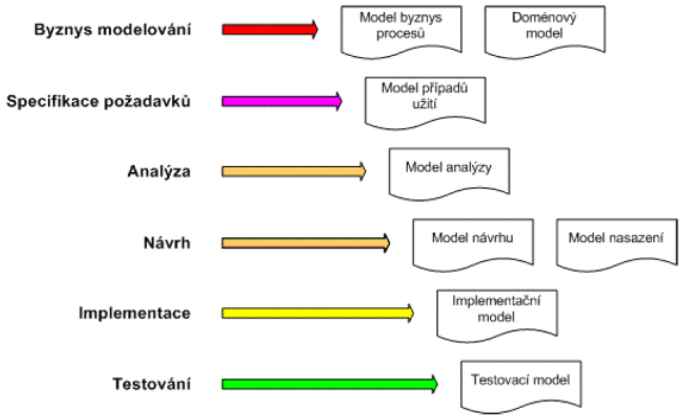
Základní a podpůrné toky činností

Vývoj systému se skládá z řady aktivit a činností, které lze uspořádat do toků majících specifický účel. Z tohoto pohledu lze hovořit o takzvaných základních tocích, jejichž výsledkem je artefakt, a o tocích podpůrných, které nevytváří žádnou hodnotu, ale jsou nutné pro realizaci toků základních.

Základní toky jsou:

- Byznys modelování, které popisuje dynamiku a strukturu podniku.
- Specifikace požadavků, která definuje za pomoci případů použití jeho funkce.
- Analýza a návrh, které jsou zaměřeny na specifikaci architektury.
- Implementace, která reprezentuje programování softwaru, testování jeho částí či komponent a jejich následnou integraci.
- Testování, které má ověřit správnost řešení.
- Integrace, která se zabývá správou konfigurace produktu v rámci cílové infrastruktury.

Výsledkem těchto toků činností jsou modely, které zobrazují systém z daného pohledu abstrakce.



Obr. 8: *Toky činností a jimi vytvářené modely [10]*

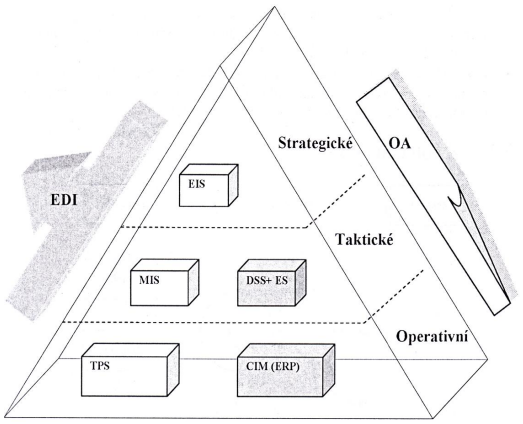
Nejdůležitější podpůrné toky spjaté s vývojem jsou:

- Řízení změn a konfigurace, které se zabývají problematikou správy verzí artefaktů.
- Projektové řízení zahrnující řízení zaměstnanců, zajištění a dodržení rozpočtu, aktivity plánovaní a kontroly výsledků. Neoddělitelnou součástí je i řízení rizik, které se snaží identifikovat problematické situace a umožnit jejich řešení.
- Správa prostředí, která poskytuje organizaci metodiky, nástroje a infrastrukturu podporující vývojový tým.

2.3 Podnikové informační systémy

Pohled na vývoj informačních systémů očima programátora či analytika je především pohledem na proces. Jiným možným nazíráním na systém je pohled očima firmy, která ho hodlá implementovat. Je to především pohled manažerů a vlastníků firem. Informační systémy jsou z pohledu komerční sféry velice užitečným nástrojem zajišťujícím při správném použití vysokou efektivitu a konkurence schopnost. Podle [6] se řízení v podniku, a z něj vyplývající druhy informačních systémů, dělí do tří vrstev. Tyto vrstvy jsou definovány podle typu řízení, které dále determinuje druhy použitých informací či možnost automatizace rozhodování.

Do **operativního řízení** spadají informační systémy typu **TPS** (Transaction Processing system) a **CIM** (Computer Integrated Manufacturing). Systémy TPS jsou nástupci dávkových systémů, které jsou umístěny přímo u zaměstnanců a jejich příkladem může být agenda „Objednávka zboží“. Naproti tomu systémy CIM se používají pro podporu výroby, která zahrnuje přímé řízení technologických procesů. Příkladem mohou být počítačem řízené NC stroje. Operativnost těchto systémů znamená příjem velkých objemů dat z výroby a jejich potencionální zpracování například mistry na oddělení.



Obr. 9: *Informační systémy z pohledu řízení [7]*

Do taktického řízení spadají informační systémy typu **MIS** (Management Information Systems) a **DSS** (Decision Support Systems). Tyto systémy mají především podporovat rozhodování managerů vyšší úrovně, než jsou vedoucí provozů nebo mistři. S podporou souvisí schopnost těchto systému provádět různé agregační a sumarizační úkony na přání uživatele. Systémy MIS mají kořeny v účetních a ekonomických systémech a DSS slouží k podpoře rozhodování s akcentem na předdefinované úlohy s přehlednými grafy a jinými výstupy.

Poslední a nejvyšší vrstvu tvoří strategické řízení firmy. Jak naznačuje vrchol pyramidy na obrázku 9, jedná se o nejhůře automatizovatelnou úroveň řízení, využívající velice agregovaná data, dolování frekventovaných vzorů a názorné, informačně nejhodnotnější způsoby prezentace přijatelné pro nejvyšší vedení podniků. Takovému typu informačních systémů se říká **EIS** (Executive Information Systems).

Části systému označované jako **OA** (Office Automation) a **EDI** (Electronic Data Interchange) se prolínají v celé hierarchické struktuře řízení. Část systému OA se stará o automatizaci administrativy, jako je použití elektronického kalendáře, či integrace elektronické pošty. Část EDI je zaměřená na komunikaci podniku s jeho okolím, které tvoří zákazníci, dodavatelé, či banky.

2.4 Efektivita a efektivnost informačních systémů

Před vlastním vysvětlením zde představím definice těchto dvou, v češtině často zaměňovaných, pojmů. **Efektivita** neboli účinnost (anglicky Efficiency) je poměr mezi přínosem procesu a náklady na něj vynaloženými. Ve výrobním procesu, ale i v jiných každodenně prováděných činnostech, se snažíme, aby naše efektivita či efektivita výroby byla co nejvyšší. Jinými slovy, abychom dosáhli maximální úrovně uspokojení našich potřeb, či potřeb vlastníků při daných vstupech a zvolené technologii.

Efektivnost neboli účelnost (anglicky Effectiveness) je vztahem mezi stanoveným cílem a výsledky našeho snažení, které se mohou obecně lišit, ač byly konány s dobrou vírou.

Z finančního pohledu tvoří investice do informačního systému výdaj jehož efektivita se dá v mnoha případech měřit jen ztěží. Zavedení informačního systému totiž většinou znamená především změnu agend z papírových na elektronické, či možnost lepšího dohledu nad procesy ve firmě a lepší rozhodování o jejím běhu. Ačkoliv lze takové „zlepšení kvality“ výstupů, které mnohé informační systémy poskytují, jen velmi těžko vyčíslit, vyznačuje se dlouhodobými přínosy (v rámci let). Jejich nasazení pak přináší spíše konkurenční výhodu než okamžitý zisk či jinou exaktně měřitelnou veličinu. Dalším problémem mohou být nadhodnocená či mlhavá očekávání, která jsou s nasazením systému spojována. V takových případech se velice špatně zpětně deklaruje přínos systému oproti přínosu jiných okolních vlivů a změn, které na podnik působí.

Užitek z informačních systémů předpokládají především:

- **majitelé** podniku, kteří očekávají dlouhodobě se zvyšující zisk podniku a informační systém berou jako jednu z možností konkurenční výhody a zefektivnění pracovních a jiných procesů,
- **manažeři** podniku, kterým by měl informační systém sloužit jako zdroj snadno pochopitelných, kompaktních a ucelených informací podpory rozhodování,

a v neposlední řadě také **zákazník**, který bude nakupovat kvalitnější produkt za nižší cenu.

2.5 Shrnutí

V předchozích kapitolách jsem nastínil funkci různých modelů životních cyklů s detailnějším popisem RUP, který je předním procesem softwarového vývoje dneška. V druhé části, zabývající se podnikovými informačními systémy, jsem se snažil nastínit jiný pohled, který na IS mají firemní zákazníci. Ti považují systém za produkt, který jim může umožnit vyšší efektivitu a efektivnost provozu a zajistit jim konkurenční výhodu na trhu.

3 Návrh Implementace

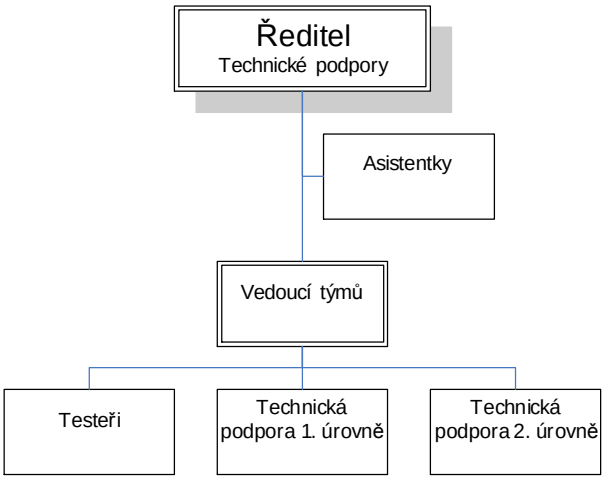
Praktická část je stěžejní částí diplomové práce. Jejím cílem je podle zadání vedoucích zaměstnanců firmy a vedoucího práce implementovat informační systém pro oddělení technické podpory firmy Grisoft s.r.o. V následujícím textu firmu představím a uvedu některé důvody, které vedou firmu k implementaci nového informačního systému. Další část práce se zabývá návrhem implementace informačního systému z pohledu použitých uživatelských rolí a akcí, které s nimi korespondují.

3.1 Popis firmy

Firma Grisoft s. r. o. sídlí na Brněnské ulici Lidická 31 a od roku 1991 působí na trhu s produkty na ochranu počítačových dat. Komplexní nabídku bezpečnostního softwaru poskytuje nejen domácím uživatelům, ale také živnostníkům a malým či velkým firmám. Vlajkovým produktem je Antivirový systém AVG Anti-Virus, který si v různých edicích již nainstalovalo více než 40 milionů uživatelů po celém světě. Zaregistrovaní uživatelé komerčních verzí AVG mohou využívat i profesionální technickou podporu, která jim v nepřetržitém provozu pomáhá řešit případné problémy. V roce 2005 vstoupili do firmy zahraniční investoři Intel Capital a Enterprise Investors, kteří za podíl v mateřské společnosti zaplatili téměř 1,25 miliardy Kč. Cílem nových majoritních vlastníků je podpořit GRISOFT v expanzi na nové trhy a upevnění pozice v USA, ČR i dalších státech Evropy. Vstup zahraničních vlastníků byl na poli softwarových firem jednou z největších investičních akcí v ČR a tak se o ní lze dočíst, krom stránek investora a Grisoftu, například i na zpravodajském serveru Živě, Idnes a dalších. Firma v současné době zaměstnává 190 stálých pracovníků a její obrat se za poslední čtyři roky vždy meziročně zdvojnásobil (v roce 2005 to konkrétně znamenalo 27 milionů dolarů).

3.2 Organizační schéma technické podpory

Informační systém bude nasazen v oddělení technické podpory firmy, kde pracuje zhruba 60 stálých zaměstnanců. Z pohledu organizační struktury jsou jimi abecedně uspořádané asistentky, ředitel technické podpory, technická podpora dvou úrovní, testeři a vedoucí týmů.



Obr. 10: *Organizační schéma technické podpory*

Z pohledu procesního jsou vedoucím týmů přiřazeni pracovníci z technické podpory obou úrovní a testeři, kterým tito vedoucí pracovníci zadávají práci. Jedná se svým charakterem o plánovanou i nárazovou práci, kterou je nutno upravovat například podle virové situace. Vedoucí týmů také přiřazují podřízené z daných skupin na projekty a projektují plán jejich specializace. Technická podpora první úrovně (nižší) pracuje jako telefonická a emailová podpora zákazníků firmy. Dále se zabývá zajištěním kvality podporovaných produktů (Quality assurance). Technická podpora druhé úrovně je složena ze zaměstnanců, kteří jsou v pracovním poměru s firmou delší dobu a mají hlubší znalosti produktů (problémy spojené s jejich používáním, správou atp.). Testování produktů a vytváření balíčků aktualizací provádí vybraní pracovníci z technické podpory obou úrovní.

3.3 Uživatelské role v systému

V informačním systému bude implementováno mnoho rolí, pod kterými mohou uživatelé provádět akce odpovídající zaměření jejich práce. Ač je systém určen především k záznamu prací, není nezbytně nutné, aby všechny role tento záznam prováděly. Využití systému může totiž po rozšíření ve firmě přerůst svůj původní záměr a mohou do něj být implementovány i jiné moduly, se kterými budou pracovat další, při návrhu neznámé, uživatelské role. Z předchozího tedy mimo jiné plyne, že

uživatelské role v systému mohou být jak pracovní (uživatelé poskytují systému záznamy o své práci), tak nepracovní. Každá pracovní role vyžaduje v systému svou část, kterou bude moci využívat k volbě a záznamu typu práce. Všechny role budou mít přístup k různým modulům zpřístupňujícím další funkce (většinou převzaté z papírových agend). Následuje stručný popis možných akcí prováděných rolemi a UML diagramy případů použití:

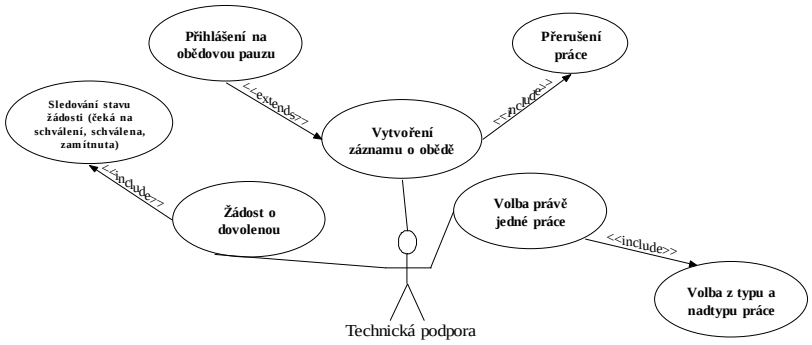
Role pracovníků technické podpory

Jedná se o role, pod kterými v systému vystupují zaměstnanci technické podpory. Tato role sestává ze dvou pod-rolí – **Support** a **Nováčci TP**. Nováčci technické podpory jsou nově najatí pracovníci, kteří musí projít zaškolením podle firemních metodik. Jejich práce jsou proto jiného druhu, což musí zohledňovat i IS ve své nabídce. Po proškolení se z nováčka stává řádný člen Technické podpory, která je v systému označena rolí Support. Významnou částí pracovní náplně těchto zaměstnanců je emailová a telefonická podpora zákazníků firmy.

Role pracovníka testování (Tester)

Jedná se o roli, pod níž vystupují pracovníci zabývající se testováním firemních produktů. Po splnění firemních kritérií mohou za jistých okolností přejít do role nováčka TP.

- **Akce role Support a Tester:**
  - Přihlášení a odhlášení ze systému,
  - výběr jazyka systému,
  - volba právě jedné práce,
  - záznam odchodu na oběd (pouze Support),
  - (modulem) žádost o dovolenou.

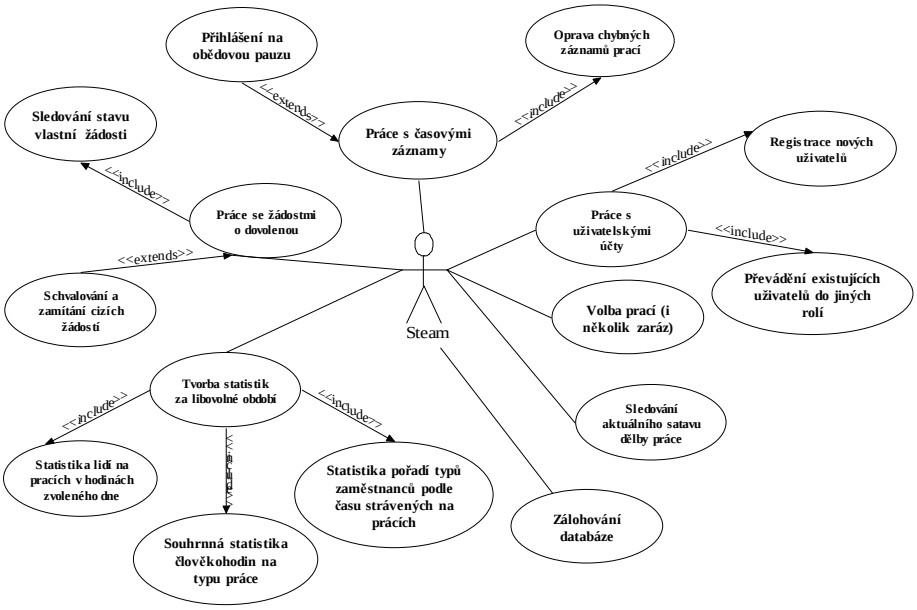


Obr. 11: *Diagram případu použití role Technická podpora*

Role vedoucího pracovníka (STeam)

Jedná se o roli, pod kterou budou v systému vystupovat vedoucí předchozích rolí. Jsou to manažeři, kteří vydávají především operativní a taktická rozhodnutí. Pod tuto roli spadá i ředitel technické podpory.

- **Akce role Steam:**
  - Přihlášení a odhlášení ze systému,
  - výběr jazyka systému,
  - volba práce (možné zvolit i několik zaráz),
  - prohlížení časových záznamů prací všech pracovníků,
  - (modulem) prohlížení záznamů o denní obědové pauze,
  - editace chyb v časových záznamech prací,
  - zálohování dat z databáze,
  - registrace nových uživatelů,
  - změna typu role existujícího uživatele,
  - sledování aktuální situace na pracovišti přes přihlášené zaměstnance,
  - tvorba souhrnných informací a grafů,
  - (modulem) žádost o dovolenou,
  - (modulem) schvalování dovolených.

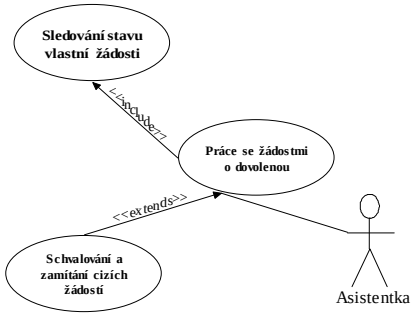


Obr. 12: *Diagram případu použití role Steam*

Nepracovní role Asistentka (Asistentka)

Asistentky jsou podřízeny pouze řediteli technické podpory a náplní jejich práce je především podpora ředitelových agend. V systému tedy nebudou vést záznamy o své práci, ale budou využívat jiné případné moduly.

- **Akce role Asistentka:**
  - Přihlášení a odhlášení ze systému,
  - výběr jazyka systému,
  - (modulem) žádost o dovolenou,
  - (modulem) schvalování dovolených.



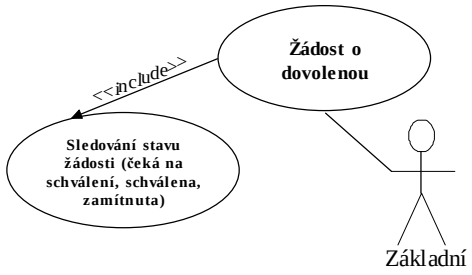
Obr. 13: *Diagram případu použití role Asistentka*

Zbývající nepracovní role (Základní)

Do zbývajících rolí, které mohou potencionálně využívat IS, spadají další zaměstnanci pracující většinu času nebo stále na pracovišti technické podpory, jejichž práce nemusí být v systému vedena. Příkladem mohou být zaměstnanci starající se o údržbu a instalaci technického vybavení.

Akce role Základní:

- Přihlášení a odhlášení ze systému,
- výběr jazyka systému,
- (modulem) žádost o dovolenou.

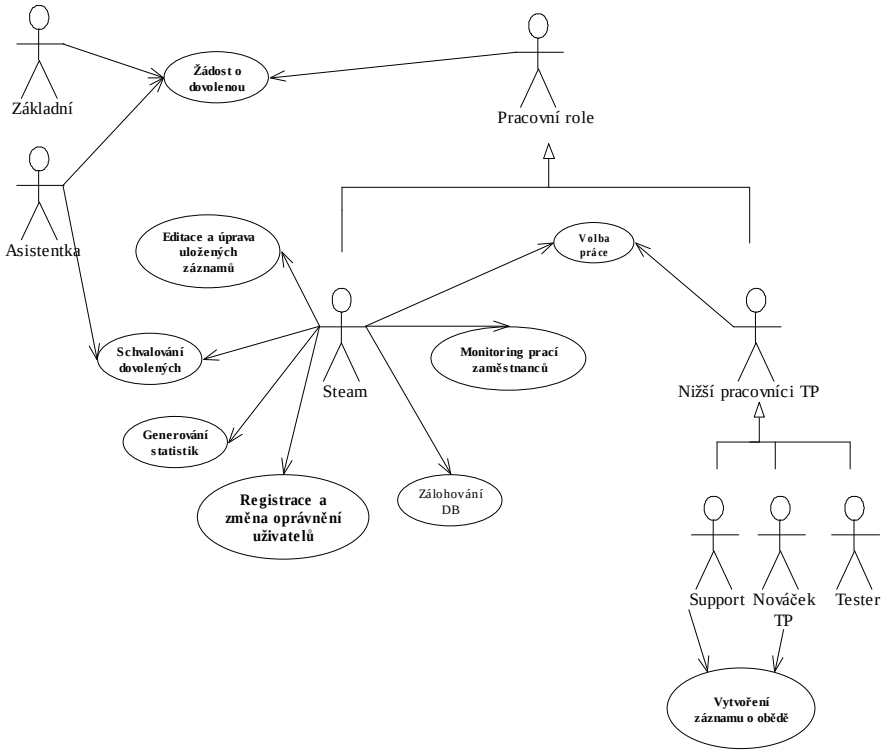


Obr. 14: *Diagram případu použití role Základní*

3.4 Stručné vysvětlení akcí

Akce ***Přihlášení a odhlášení ze systému*** poskytuje uživateli možnost autentizace a ISu správné nastavení dalších akcí, které může uživatel podniknout. Uživatel se autentizuje svým jménem a heslem. ***Výběr jazyka systému*** umožňuje volbu z jazykových mutací systému, aktuálně se jedná o českou nebo anglickou mutaci. ***Výběr práce*** umožňuje pracovním rolím zadat do databáze ISu právě prováděnou pracovní náplň. STeam může pracovat na více pracích zaráz, zatímco ostatní pracovní role pouze na jedné. ***Žádost o dovolenou*** je funkce, přes kterou mohou všechny role bez výjimky žádat o dovolenou, která je ***schvalována*** STeamem či Asistentkami. ***Záznam odchodu na oběd*** je akcí, která do databáze uloží počet a identifikace lidí, kteří si na začátku směny volí variantu hodiny obědové pauzy. Tato informace může sloužit k vyhodnocení později zpozorovaných potíží a k rozdělení pracovníků odcházejících na oběd do dvou stejně početných skupin.

V systému má výhradní postavení role managerů STeam. Co se týče řízení a participace na provozu, tato role může pracovat na více pracích zaráz a může jako jediná sledovat aktuální rozdělení pracovníků na pracích, generovat sumární statistiky a zobrazovat informace historického charakteru (o přihlášeních zaměstnanců na obědy i o jejich pracích).



Obr. 15: *Diagram případu použití systému technické podpory*

3.5 Podrobnější vysvětlení akcí

V následujícím textu navrhnu a blíže popíši zmíněné akce a jejich příslušnost k jednotlivým rolím vystupujícím v sytému.

Akce přihlášení do systému

Uživatel se bude autentizovat přiděleným uživatelským jménem a heslem. Heslo by nemělo být používáno k přístupu do jiných systémů a pro uživatele pracující pod rolí STeam by mělo být složité (dlouhé a složené z alfanumerických a speciálních znaků). Větší zabezpečení přihlášení pod tímto účtem může zabránit vyzrazení informací o fungování firmy a zcizení či změně údajů uložených v databázi.

Všechny údaje o přihlášeném uživateli včetně jeho oprávnění jsou **ukládána** a předávána prohlížeči přes **cookies**. V první fázi nasazení informačního systému bude z bezpečnostního hlediska

použito pouze ukládání a práce s hesly ve formě jejich hash otisků. Za použití šifrovaného spojení se serverem a uvážíme-li, že systém slouží pouze pro vnitřní potřebu firmy, bude bezpečnost dostatečná.

Akce odhlášení ze systému

Po ukončení práce se systémem by se měl uživatel odhlásit, což automaticky ukončí všechny jeho otevřené práce (systém odhlásí uživatele z prací, na kterých je přihlášen). Systém však nabízí možnost odhlášení i bez ukončení rozpracovaných činností. Protože technologie webové aplikace nemá možnost ověřit zavření okna aplikace, je nutné odhlášení z práce provést. Pokud bude například zavřeno okno prohlížeče, aniž by se uživatel řádně odhlásil z práce, bude záznam této poslední započaté činnosti stále aktivní. Po následném přihlášení však pracovník uvidí, že je stále přihlášen na práci a bude moci tuto chybu ohlásit vedoucímu, který má možnost ji změnit zásahem do databáze.

Výběr jazyka systému

Protože se do budoucna počítá s externími uživateli informačního systému, je nutné, aby poskytoval podporu více jazyků. V počáteční fázi se bude jednat o českou a anglickou mutaci systému. Tato mutace zahrnuje všechny dialogy, informativní nápisy i názvy prací a jiné údaje získávané z databáze. V neposlední řadě se jedná i o jazykové mutace uživatelské dokumentace.

Odchod na oběd

Aby se zabránilo časovým prodlevám v obsluze zákazníků, kteří čekají na technickou podporu, musí se zaměstnanci na začátku směny (do času první obědové pauzy) přihlásit na jednu z variant obědových pauz. Systém zaručuje rozdělení pracovníků do dvou obědových pauz půl na půl. Zmíněné rozdělení je zaručeno s přesností na jednoho pracovníka v případě, že si všichni zaměstnanci volící práci, zvolí i termín odchodu na oběd. Nemůže se tedy stát, že by na podpoře nikdo nezbyl, protože všichni pracovníci odešli ve stejnou dobu na oběd.

Žádost o dovolenou

Modul v systému může řešit zasílání žádosti o dovolenou a její schválení. Pracovník uvede datum nástupu a ukončení dovolené, které se ověří se zadanou délkou trvání. Další vkládanou informací je místo pobytu a další popis. Zaměstnanec bude mít možnost si žádanku zrušit (před schválením i po něm) například z důvodu změny situace, která dovolenou vyžadovala. Informace o plánovaných dovolených lze využít při sestavování přesnějšího plánu směn a taky pro větší spravedlnost při přidělování dovolených. Spravedlnost může být například popsána větou: „Pokud o dovolenou ve stejnou dobu žádají dva pracovníci, měl by být upřednostněn ten, který měl poslední dovolenou dříve.“

Schvalování dovolených

V rámci technické podpory existuje systém oprávnění, který definuje, kdo může komu schvalovat dovolené. Systém se tedy nebude starat v úvodní fázi implementace o zaručení těchto

práv, ale všechny pověřené role budou moci dovolenou udělit. Jsou to role STeam a Asistentky, které mají ve firmě možnost podílet se na sestavování a kontrole plánu směn. Při zamítnutí dovolené je nutné, aby se zaměstnanec zpětně dozvěděl důvod a případně i jiný termín, kdy mu bude dovolená umožněna. Přenos této informace by se měl také uskutečnit skrze informační systém.

Sledování aktuálního dění na pracovišti

Vedoucí týmů mají za úkol rozdělovat práci podřízeným, zejména pracovníkům technické podpory tak, aby nedocházelo ke zbytečným prodlevám při vyřizování požadavků zákazníků. Zároveň ale musí myslet i na plánování času potřebného k dalšímu vzdělávání zaměstnanců, nebo zajištění kvality vydávaného produktu (například testováním).

Pro rozdělení prací musí mít vedoucí pracovníci v informačním systému nástroj, který je bude přehledně informovat o situaci na pracovišti, zejména o rozdělení zaměstnanců na jednotlivé typy úkolů (například práce na telefonní podpoře či testování) ve sledovaném čase. Pokud se bude schylovat k závažnějším problémům v některé z oblastí podpory zákazníků, měli by mít přehled, kteří zaměstnanci jsou k dispozici na jiných pracích, či studiu, a mohli by být přeřazeni na aktuálně potřebnější práci.

Sledování historických dat

Jedním z hlavních přínosů informačního systému je možnost snadno a efektivně uložit a zpracovávat historická data. Ve vyvíjeném systému půjde především o zobrazování dat týkajících se dělby práce v jednotlivých dnech, zobrazení využité obědové pauzy zaměstnanci a generování souhrnných údajů o pracích zaměstnanců za delší časová období.

Generování statistik

Problematika zobrazení agregátních informací je pro každý systém umožňující podporu rozhodování hlavní a manažery nejpoužívanější funkcí. Systém musí umožnit v rozumné době zpracovat a zobrazit přesné údaje o práci zaměstnanců a jejich dělbě práce za různé časové úseky. Především se jedná o týdenní, měsíční a kvartální statistiky práce. Systém by měl ovšem také vedoucím pracovníkům umožnit generovat statistiku za libovolně dlouhé období. Statistiky mohou být tří hlavních typů. **Podrobná denní statistika** přidělených pracovníků na jednotlivé práce po hodinách. **Souhrnná statistika** za dané období zobrazující počet člověkohodin na daném typu práce, počet odpracovaných hodin zaměstnanci na všech pracích a celkovou sumu člověkohodin přes všechny zaměstnance a všechny práce. **Pořadová statistika** zobrazí pracovníky seřazené podle hodin strávených na jednom typu práce a typy prací seřazené podle odpracované doby jedním pracovníkem.

Zálohování databáze

Pověření pracovníci by měli mít možnost uložit obsah databáze s možností její obnovy při fatální chybě. Zálohování by mělo být co nejsnadnější a povinnost provádět zálohy a doručit je na bezpečné místo by měla být součástí firemních metodik povinností zaměstnanců starajících se o běh informačního systému.

Registrace a změna oprávnění uživatelů

Jak jsem naznačil v části zabývající se rozdělením a hierarchií rolí ve firmě, existuje pracovní postup, v rámci kterého mohou jednotliví zaměstnanci v různých časech zastávat různé role. Například po zaškolení Nováčka technické podpory se ze zaměstnance stává řádný člen technické podpory. Tuto změnu role odráží z pohledu zaměstnance v informačním systému především změna nabízených typů práce. Registraci nového zaměstnance, stejně jako změnu jeho role, provádí vedoucí pracovník.

Volba práce

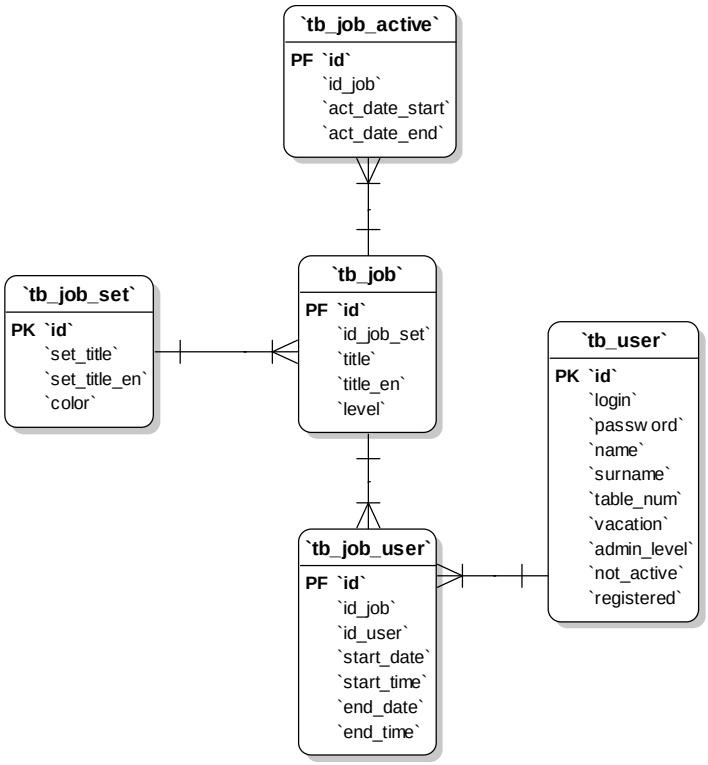
Nejdůležitější funkcí systému je sbírání informací o čase prací prováděných zaměstnanci. Konkrétně se jedná o záznam typu práce, na kterém je přidělen určitý pracovník po určitou dobu. Typy prací, či přesněji řečeno úloh (může se jednat i o úlohy odchodu na oběd, samostudia, či výjezdu na školení), se nemusí mezi rolemi lišit. Příkladem může být právě zmiňovaná obědová pauza, která je samozřejmě stejná pro testera i vedoucího pracovníka. Systém tedy musí zajistit ukládání informací o stejných pracích jakožto položkách stejného typu a, aby byla situace ještě složitější, měl by také umožňovat souběžný průběh více prací. V implementaci budou poté role STeam moci provádět více prací naráz, což se musí ve výsledných součtech časů projevit jako poměrné rozdělení času mezi všechny souběžně prováděné práce. Tento temporální problém implementovaný v relační databázi bude jednou z nejtěžších implementačních výzev.

3.6 Návrh databázového schématu

V následující části stručně popíši návrh databázového schématu pro uživatele a práce. Systém ve své databázové implementaci obsahuje další tabulky pro moduly, které však nejsou mým výlučným vlastnictvím. Pokaždé se jedná v podstatě o tabulku uchovávající údaje a číselník, ze kterého si zaměstnanec vybírá (směnu, obědovou pauzu, typ poznámky atp.). Nepovažuji tedy za nutné popisovat v této práci i návrh modulů.

3.6.1 Uživatelé a práce

Obrázek 16 zachycuje ER diagram, jehož entity uchovávají informace o zaměstnancích a jimi provedených pracích. Informace o zaměstnanci jsou uloženy v databázové tabulce tb_user. Jde o jeho přihlašovací jméno (login), heslo (password), jméno a příjmení (name a surname), číslo stolu, u kterého na pracovišti sedí (table_num), počet dní zbývající dovolené (vacation), úroveň oprávnění (admin_level), příznak neaktivního pracovníka (not_active) a datum registrace (registered).



Obrázek 16: *ER diagram části pracovníků a jejich prací*

Entita uživatelé je ve vztahu M:N k pracím, které jsou v databázi reprezentované tabulkou tb_job. Každá práce je součástí jednoho pracovního nadtypu, jehož informace jsou uloženy

v tb_job_set. Dále je o každé práci veden časový interval, po který byla, třeba opakovaně, proveditelná (bylo ji možné zvolit a pracovat na ní). Tato informace je uložena v tabulce tb_job_active. Tato tabulka je sice navržena a implementována, ale zatím není v systému nijak využívána. Jinými slovy, není možné „deaktivovat“ práci a poté ji znovu zařadit jako volitelnou. Návrh počítá s definicí intervalu platnosti práce s přesností dne od počátku (act_date_start) po konečný den platnosti (act_date_end).

Každá práce, která je definovaná relací tb_job, musí být součástí pracovního nadtypu (množina prací, která některé práce sdružuje pro lepší orientaci a možnost agregace dat). Informace o nadtypu jsou uloženy v tabulce tb_job_set. Jedná se o český název nadtypu (set_title), anglický název nadtypu (set_title_en) a barvu nadtypu (color). Samotné práce mají také svůj anglický a český název a navíc i oprávnění, ze kterého lze zjistit, které role si ji mohou volit.

Uživatelé pracující se systémem, které popisuje informace v tabulce tb_user, volí práce z tabulky tb_job. Poněvadž mohou zaměstnanci volit v rámci své role obecně více prací a práce jsou samozřejmě voleny více nežli jedním zaměstnancem, je nutné aby i relace popisující tyto entity byly ve vztahu M:N. S výhodou lze poté využít vlastnosti relace implementované tabulkou tb_job_user, ve které lze uchovat při volbě práce její parametry, kterými jsou datum a čas začátku (start_date, start_time) a datum a čas konce intervalu zvolené práce (end_date, end_time).

4 Informační systém technické podpory

Informační systém musí splňovat požadavky kladené

- vedením technické podpory,
- zaměstnanci technické podpory,
- zaměstnanci pracujícími na úseku testování a
- dalšími interesovanými skupinami.

Vedení technické podpory potřebuje především nástroje pro účelné sledování informací taktického a operativního charakteru. Informační systém by jim tedy měl poskytovat informace o pracujících zaměstnancích, počtu lidí na jednotlivých pracích a další informace operativního charakteru. Agregovaná data potom mohou být použita pro efektivnější plán školení zaměstnanců, dokladování aktivit, které na oddělení technické podpory probíhají, či odhady časů, které zaberou budoucí úkoly.

Zaměstnancům by měl IS pomocí komfortního a jednoduchého prostředí ulehčit poskytování informací nadřízeným a redukovat papírové agendy. Zmiňované agendy se mohou týkat například vyplňování žádostí o dovolenou. V ISu může existovat modul, který zpřístupní žádost i zobrazení průběhu schvalovacího procesu přímo v pracovním prostředí uživatele.

Zaměstnanci provádějící testování by měli mít snadný přístup k testovacím programům přes rozhraní IS. Systém tak bude plnit centralizační funkci a zaměstnanci nebudou muset pracovat s více systémy.

Další skupiny mohou být zastoupeny vedením firmy či akcionáři, kteří mohou být lépe a kvalitněji informováni o pracích prováděných v technické podpoře. Jedná se například o sumy časů za dlouhá období strávené na jednotlivých typech prací, či meziroční nárůsty odpracovaných hodin.

4.1 Výběr implementačních nástrojů

Informační systém v sobě nese silné vazby na implementaci skrze databázové technologie. Není proto vhodné implementovat jej v jazyku či prostředcích, které databáze přímo nepodporují. Rozhodl jsem se implementovat tento systém v programovacím jazyce PHP v součinnosti s databází MySQL a serverem Apache. Hlavními přednostmi těchto systémů je jejich **nezávislost na platformě** a **cena**.

S výhodou lze také využít **předem nakonfigurovaný** a aktualizovaný **balík** těchto tří nástrojů pro Windows s názvem Php Home Edition (http://sourceforge.net/projects/phphome/) či PHP Triad (http://sourceforge.net/projects/phptriad/). Další výhodou použití PHP je existence aplikace **PhpMyAdmin**, která výrazně zjednodušuje správu databáze. Tyto systémy jsou též **součástí** některých **distribucí** Linuxu a FreeBSD.

Z pohledu serveru

Všechny tři systémy jsou k dispozici zdarma pro oba nejpoužívanější OS (Windows, Linux). Server tedy může běžet na kterémkoliv moderním stroji a systému. Další zvažované varianty jako například ASP, PWS, MsSQL a Oracle jsou placené (v případě PWS/IIS je součástí licence zakupované s OS Windows). Systém ASP je navíc silně vázán na řešení firmy Microsoft. Výhodou Apache či IIS je jejich rozsáhlá dokumentace z různých zdrojů, která není tak patrná u dalších webových serverů, jako je např. Xitami (další informace jsou dostupné na http://www.xitami.com).

Z pohledu klienta

Přistupovat na stránky PHP není problémem pro žádný z moderních internetových prohlížečů. Na všech OS Windows může být instalován prohlížeč Internet Explorer verze 5.5 a vyšší. V těchto prohlížečích je zároveň systém implementován. Z tohoto důvodu je možné zavrhnout placené systémy aplikačního typu jako Gupta či Oracle.

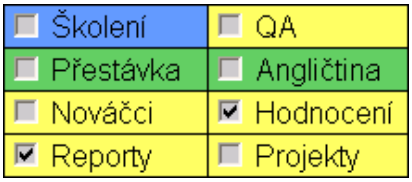
Shrnutí

PHP, MySQL a Apache jsou dostupné na mnoha OS a jsou dostupné zdarma. IS bude používán interně, a není tedy třeba brát zřetel na zabezpečení či robustnost (potřeby budou bez problému pokryty zmiňovanými systémy). Všechny systémy jsou dobře dokumentované a jejich dokumentace je k dispozici zdarma (http://cz.php.net, http://www.mysql.com, http://www.apache.org). Existence instalačních balíků a konfiguračních programů značně zjednodušuje nasazení systému.

4.2 Implementace systému

Zaměstnanci vystupující v systému pod jednou z pracovních rolí mají možnost volit práci, kterou právě provádějí. Systém umožňuje vybraným rolím vybírat pouze jednu práci z dané nabídky pro tuto roli nebo více prací současně. V druhém případě je práce na jednotlivých, paralelně prováděných činnostech brána jako rovnoměrný podíl času vynaloženého na všechny zvolené práce. Například pokud budou souběžně hodinu prováděny práce A i práce B, systém bude s tímto faktem nakládat (například při vypočítávání statistik), jakoby každá práce byla zaměstnancem prováděna půl hodiny. Právě kvůli výpočtu statistik a sledování odpracovaného času je nutné takové pravidlo zavést. Pokud by například obě práce v předchozím příkladu vykonával pracovník celou osmihodinovou směnu,

a čas na nich strávený by se žádným způsobem nepřepočítával, znamenalo by to v důsledku, že pracovník strávil prací 16 hodin (2x osmihodinovou směnu). Tento fakt by mohl významně degradovat data od všech zaměstnanců pracujících pod účty rolí, které by dovolovaly zvolit více prací zaráz.



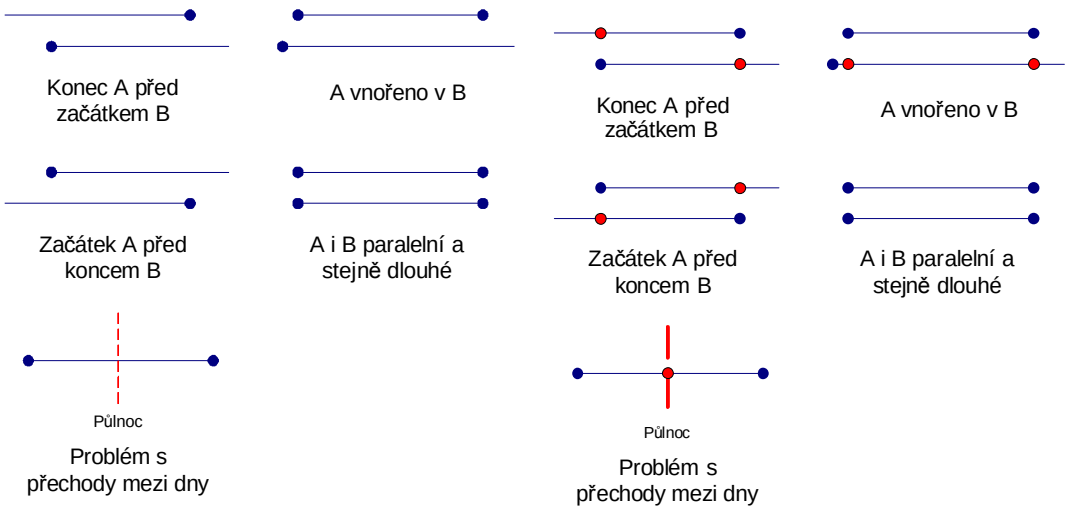
Obr. 17: *Tabulka pro volbu paralelně prováděných prací*

Pokud zaměstnanec vybere prováděnou práci a potvrdí ji, je záznam o začátku práce uložen do databáze. Taková práce je rozpracována a není znám její konec do doby, než ji zaměstnanec explicitně ukončí (odhlášením ze systému s ukončením všech prací či jejich převolením). Práce s rozpracovaným záznamem přináší jistý problém s vyhodnocováním jejího konce. Například pokud má být v systému zobrazeno, jak dlouho se zaměstnanec práci věnoval (vždy za nějaké dané období), je nutné spočítat rozdíly mezi začátky a konci intervalů prací, které byly v daném intervalu aktivní. V případě rozpracovaných prací je tedy možné buď takové práce nezobrazovat (což je problematické například pokud zaměstnanec je za celou směnu přihlášen pouze na jedné práci), či aplikačně nebo na straně databázového dotazu nahradit neznámou hodnotu konce práce hodnotou aktuálního data a času. Samostatným problémem je ukládání temporálních dat v relačních databázích.

Problematika temporálních dat v relačních databázích

Systém využívá, jako drtivá většina internetových aplikací dneška, relační databázi. V mém případě mimo jiné pro uložení temporálních dat. Temporální databáze poskytují nástroje pro granularizaci času, ta je v relačních databázích řešitelná v podstatě jen na aplikační úrovni (například volba časového kroku při zobrazení přidělení pracovníků na práce by mohla být jakkoliv elastická, například po minutě, ale i po třech minutách). V relační databázi není možné jednoduše vymezit interval času, o který se zajímáme. Pokud chceme například generovat statistiku za období jednoho měsíce od začátku ledna do začátku února, bylo by vhodné přímo v databázovém dotazu definovat časové okno, které nás zajímá. V relačních databázích je nutné takový dotaz ošetřit podmínkami (jejich výskyt stoupá exponenciálně s množstvím relací jejichž časy musíme spojovat). Relační kalkul dále nedisponuje temporálními operacemi. V mém případě především operací, která by zjistila interval vzniklý průnikem intervalů. Dalším problémem je nemožnost explicitní změny relačního modelu tak, aby se středem pozornosti (především z hlediska indexovacích klíčů) staly časové atributy relace. V mém případě počátek a konec intervalu prováděné práce, nebo datum volené směny (o této vlastnosti systému se zmíním později). V relačním návrhu jsem se tedy snažil alespoň o imitaci těchto temporálních vlastností vytvořením indexu nad atributy, které se týkají data. Navíc

jsem logický atribut datum a čas začátku (reprezentovatelný v MySQL například datovým typem DATETIME) rozdělil na dva atributy relace datum (datový typ DATE) a atribut času (datový typ TIME). Datum a čas začátku a konce intervalu jsou potom realizovány celkem čtyřmi atributy. Tento návrh sice dotazy ztíží, ale v kombinaci s indexy nad všemi zmíněnými atributy a optimalizací dotazů SŘBD (které se ptá zvlášť na datum a zvlášť na čas, což by mělo výsledek dotazu značně redukovat) by měla být aplikace rychlejší.



Obr. 17: *Možné kombinace paralelismu časových intervalů a problém mezníku půlnoci.*

Na ilustraci jsou vyobrazeny lineární časové intervaly A a B a jejich možné kombinace. Modré body znázorňují začátek či konec intervalu a modrá čára, jeho trvání. Problém při zpracování však nezpůsobuje pouze situace, v níž jsou A i B paralelní a zároveň stejně dlouhé (mají stejnou dobu trvání). Speciálním případem je dělení intervalu dle jistého mezníku, jímž může být například půlnoc při vyhodnocování statistik odpracované doby za předpokladu granularity na dny (při granularitě na hodiny by mezníkem byl konec každé hodiny atp.). Konkrétně tedy může jít o spočtení odvedené práce zaměstnancem za období jednoho dne v případě, kdy má právě směnu, která končí až následující den. Interval potom může být buďto bez konce (pracovník stále pracuje), nebo může končit až druhý den, který nás však nezajímá, protože statistiku zpracováváme za den aktuální.

Řešením, jak se všemi takovými problémy naložit, je filozoficky jediné, jeho implementace se však lze zhostit mnoha způsoby. Jde o rozbití intervalů, které se ve výsledku buď vůbec nepřekrývají, nebo se překrývají zcela. Na ilustraci 18 jsou rozdělení naznačena červenými body. Tuto situaci lze opět vztáhnout na příklad s pracemi, které jsou paralelně prováděné a proto je tedy nutné čas na nich strávený při výpočtu dělit počtem prací (abychom ve výsledku pracovali jednu osmihodinovou směnu jak jsem popisoval výše). Máme tedy například práci A, která začala dříve než práce B ($\alpha_A < \alpha_B$) a také dříve skončila ($\omega_B > \omega_A$). Znamená to, že pracovník napřed začal provádět práci A, v jejím průběhu začal paralelně provádět po jistou dobu B, A ukončil a pracoval obecně jinou

dobu jen na práci B. Máme tedy časovou posloupnost α_A, α_B, ω_A, ω_B. Výpočet je tedy v tomto případě potřeba provést tak, že se započítá 100% času věnovaného práci A v intervalu $<\alpha_A, \alpha_B)$, 50% času práci A i B v intervalu $<\alpha_B, \omega_A)$ a nakonec 100% času věnovaného práci B v intervalu $<\omega_A, \omega_B>$. Samotná implementace tohoto rozdělení může být obsloužena buď na straně databáze v podobě databázových triggerů nebo na straně aplikace.

Použití triggerů

Při použití této techniky rozdělování intervalů je práce přesunuta na bedra databázového systému. Pro programátora je to nejtransparentnější řešení, a pokud je databázový systém dostatečně robustní, dalo by se říci, že je to řešení nejlepší. MySQL ve verzi pět databázové triggery podporuje, je to ovšem jedna z nově přidaných vlastností, a proto by jejich použití mohlo způsobit celou řadu problémů. Pokud by triggery nefungovaly stoprocentně správně, mohlo by být například počítání statistik nefunkční, nebo by mohlo obsahovat chyby.

Zpracování na straně aplikace

Rozpoznání překrytí intervalů na straně aplikace sestává z několika kroků. Prvním, je získat záznamy ze sledovaného období (například jeden měsíc od 1.1. do 31.1.), druhým je rozpoznání paralelismů a třetím je samotný výpočet. První krok je zcela zřejmý, a neexistuje mnoho způsobů, jak jej modifikovat. Data jsou jednoduše získána z databáze za použití vhodného dotazu a podmínky, která omezuje vrácenou relaci na daný časový úsek. Rozpoznání paralelismů a výpočet je však část, která může být implementována různě.

Naivní implementace skrze databázové dotazy ve smyčce

Data jsou v databázi uložena s jistou přesností. V mém případě je použito datového typu TIME, který také určuje maximální možnou přesnost výpočtu statistik (přesnost na sekundy). Je však možné data počítat i s přesností například na minuty. Naivní implementace rozpoznání paralelismů spočívá v programové smyčce, která vykonává dotaz nad databází po časových úsecích dané přesnosti. Dotaz poté vrátí počet paralelně prováděných prací za daný úsek. Například můžeme chtít prozkoumat hodinový časový úsek <11:00, 12:00) s přesností na minuty. Můžeme k tomu použít smyčku po minutách, která bude v dotazu vracet, kolik prací bylo paralelně prováděno. Pokud počítáme například denní statistiku, musí být tyto práce prováděny jen jedním zaměstnancem. Pokud bychom chtěli získat tato data pro všechny zaměstnance, museli bychom vytvořit ještě jeden nadřazený cyklus, který by přes ně iteroval. Cyklus by měl tedy pro hodinu a minutovou přesnost šedesát provedení, ve kterých by se vracela čísla od 0 (pokud pracovník danou práci vůbec neprováděl) až po X (které by znamenalo maximální počet paralelně prováděných prací). Výpočet stráveného času na práci by se potom dal popsat jako suma $\Sigma(1/X)$ minut.

Tato implementace je naivní, protože neúměrně zatěžuje síťovou komunikaci a databázový server neustálými dotazy. Navíc exponenciálně roste s novými zaměstnanci a nově zaváděnými

pracemi. Například pro výpočet statistiky za jeden den (24*60 = 1440 minut), 20 zaměstnanců a 15 typů práce, by bylo nutné provést 432 000 dotazů na databázi. Za jeden nepřestupný rok by to bylo více než 157 milionů. Jedinou výhodou této metody tedy je její snadná implementace a nenáročnost na aplikační logiku. Implementována může být pouze suma v rámci asociativního pole, které má klíče dle typů prací.

Implementace zásobníkovým automatem

Jak jsem uvedl výše, rozpoznání paralelismů je také možné zásobníkovým automatem. Pokud zjistí zanoření začátků, jsou práce prováděny paralelně. Pokud narazí na konec, je daná práce ukončena a symbol ukončené práce je ze zásobníku vyjmut. Počet paralelně prováděných prací je dán počtem symbolů (například identifikátorů typů prací) na zásobníku. Implementace spočívá v získání všech prací zaměstnance za sledované období, jejich seřazení dle začátků a zpracování zásobníkovým automatem. Přesnost této metody je maximální (v navržené implementaci na sekundy). Tento způsob výpočtu jsem se však nakonec také rozhodl neimplementovat kvůli nepřehlednosti kódu.

Implementace rozdělujícím polem dané přesnosti

V systému je použita implementace, kterou popíši na tomto místě. Jedním dotazem jsou zjištěny práce zaměstnance za dané období, které jsou již v dotazu zaokrouhleny na celé minuty (udávající také přesnost výpočtu). Aplikace poté iteruje přes minuty ve všech navrácených intervalech provedených prací (nemůže dojít k tomu, že počet prováděných prací je v dané minutě rovno nule). Do pole jsou zapisována čísla vyjadřující počet paralelně prováděných prací (jejich časová hodnota je tedy 1/X minut, kde X je zjištěné číslo).

Tento způsob je rychlý díky jedinému dotazu, který je nutné položit. Není zcela optimální, ale reálná přesnost se pohybuje kolem deseti minut na jedné osmihodinové směně, což je přípustné. Implementace takového způsobu není na počítání příliš náročná.

Implementace průniku volitelných prací rolemi

V rámci systému mají různé role na výběr různé činnosti, na jejichž provádění se hlásí. Množiny těchto činností mají obecně neprázdný průnik. Tuto situaci je tedy možné například implementovat jako různé řádky tabulky (různé práce) se stejnými názvy i jinými informacemi, avšak rozdílnou náležitostí k roli. Další možností je spojovací tabulka, ve které bude uspořádaná dvojice (id role, id práce). Ve své práci jsem implementoval třetí variantu, čili speciální kódování příslušnosti práce v rámci entity práce (tabulka tb_job). Kódování pak zajišťuje přímo atribut relace (level).

Každá role má v systému své číslo oprávnění, které je zároveň prvočíslem. Každá práce má také své oprávnění, které znamená, které role ji mohou volit. Pokud je příslušná práce volitelná, tak platí, že oprávnění práce MOD oprávnění role je rovno nule. Pokud je tedy práce volena právě jednou rolí, je číslo jejího oprávnění rovno oprávnění role. Pokud mají možnost práci volit role dvě, je číslo oprávnění práce dáno jako číslo vzniklé vynásobením oprávnění obou rolí.

Sledování aktuálního dění na pracovišti

Systém poskytuje přehledné zobrazení aktuálního rozdělení práce na pracovišti. Nástroj, který tyto informace poskytuje pro každou uživatelskou roli, se nazývá monitor. V systému lze tedy najít monitory všech pracovních uživatelských rolí, jimiž jsou Testeři, Nováčci, Support a STeam.

Ve sloupcích monitoru jsou názvy všech prací, které může daná role provádět i s jejich nadtypy. Nadtyp práce zde znamená pojmenované sdružení několika prací, například nadtyp Support může obsahovat práce Fax, Maily, Telefony (technická podpora zákazníkům na těchto komunikačních médiích). První sloupec (nejvíce vlevo) zobrazuje čas od půlnoci zvoleného dne do aktuálního času či půlnoci dne následujícího (v případě, že se jedná o uplynulý den). Sloupec nejvíce vpravo zobrazuje počet zaměstnanců přítomných na pracovišti (dané role). V buňce, ležící na průsečíku typu práce a hodiny je zobrazeno číslo, udávající počet zaměstnanců, kteří v daném čase práci prováděli. Na ilustraci je například vidět, že v čase mezi 7:00 a 7:59 bylo na Mailové podpoře nasazeno sedm pracovníků. Uživatel systému může snadno zjistit jejich jména najetím myší nad toto číslo. Na ilustraci jsou zvlášť vyznačeny začátky ranní, odpolední a noční směny (RA, NO, OD). Ve sloupci zcela vpravo lze vidět nárůst přítomnosti zaměstnanců na pracovišti v čase příchodu do práce, a pokles při konci směn.

Čas	Support			Testovani			Počet lidí celkem
	Fax	Maily	Telefony	Testovani	Update	Bugzilla	
0:00-0:59			1				1
1:00-1:59			1				1
2:00-2:59			1				1
3:00-3:59			1				1
4:00-4:59			1				1
5:00-5:59			1				1
6:00-6:59			3				3
7:00-7:59	RA	7	3				12
8:00-8:59		12					15
9:00-9:59		12					15
10:00-10:59		11					15
11:00-11:59		10					15
12:00-12:59		8					15
13:00-13:59		12			1		15
14:00-14:59		13	2		1		17
15:00-15:59	OD	18	3				21
16:00-16:59		17	2		1		21
17:00-17:59		15	3		1		20
18:00-18:59		14	2		1		18
19:00-19:59		6	1				6
20:00-20:59		5	1				6
21:00-21:59		5	2				6
22:00-22:59	NO	5	3				7
23:00-23:59		5	3				8

Pracovníci
Zdeňka Červenková
Ludmila Sedlářová
Jana Pelcová
Helena Hlavatá
Zdeňka Veselá
Jitka Hloušková
Jiřina Ludvíková

Obr. 19: *Monitor s vyznačenými začátky ranní, odpolední a noční směny*

Statistiky

Z uložených dat o pracích lze v systému generovat přehledné statistiky i s grafy, jak je ilustrováno obrázkem 19. Statistiky ukazují souhrnný počet člověkohodin a času strávených na jednotlivých pracích i na pracovních nadtypech (množinách prací) za sledované období. Je zobrazen i denní průměr a čas přepočítaný na dny a hodiny, pokud je sledované období delší než jeden den.

Souhrn počtu osob od 01.05.2006 do 07.05.2006 (Celkem 7 dní)

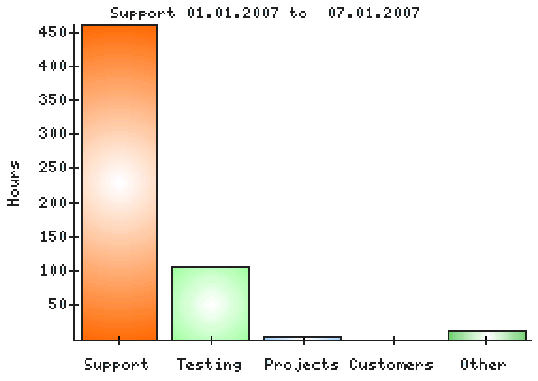
Support	Fax	Maily	Telefony	Testovani	Update	Bugzilla	FAQ/ LAB/ IKB	Avir	Skoleni	Servis	Prestavka	Anglictina	Studium
Suma 8h/1 člověk	1,14	8,17	54,3	18,26	0,78	3,67	8,28	0	0	0	0,82	3,29	0,09
Suma lidí	63,61			22,71			8,28		0		4,2		
Denní průměr	9,09			3,24			1,18		0		0,6		

Souhrn stráveného času za období od 01.05.2006 do 07.05.2006 (Celkem 7 dní)

Support	Fax	Maily	Telefony	Testovani	Update	Bugzilla	FAQ/ LAB/ IKB	Avir	Skoleni	Servis	Prestavka	Anglictina	Studium
Suma časů	9:09 h	65:21 h	434:23 h	146:03 h	6:17 h	29:20 h	66:15 h	0:00 h	0:00 h	0:00 h	6:34 h	26:18 h	0:45 h
Suma typů	508:52 h (21 dní 4:52 h)			181:40 h (7 dní 13:40 h)			66:15 h (2 dny 18:15 h)		0:00 h		33:37 h (1 den 9:37 h)		
Denní průměr	72:42 h			25:57 h			9:28 h		0:00 h		4:48 h		

Obr. 20: *Týdenní statistika*

Tento typ statistik, v systému označený jako kompletní statistiky, navíc generuje i graf jednotlivých typů práce a jejich nadtříd. Graf nadtříd slouží k okamžité orientaci v odpracovaných objemech práce za sledované období, jeho ilustrace je na obrázku 21. Graf koresponduje s tabulkou z obrázku 20. Lze z něj snadno vysledovat, že nejvíce hodin bylo věnováno nadtypu Support (tento nadtyp zahrnuje práce Fax, Maily, Telefony), druhý v pořadí, co do odpracovaného času je nadtyp testování, třetí jsou ostatní akce, poté projekty a nakonec nadtyp zákazníci, kterým ve sledovaném období nebyl věnován žádný čas. Ilustrace 21 navíc ukazuje internacionalizaci systému. Veškeré popisky grafu i práce jsou na ilustraci v angličtině (ne proto, že by musely být, ale protože byly generovány při zvoleném anglickém jazyku).



Obr. 21: *Graf statistik nadtříd za sledované období*

Posledním typem statistik jsou pořadové statistiky, které jsou v systému nazvány Toplist statistiky. Jedná se o zobrazení pracovníků a jejich prací, které jsou seřazené dle času, který jim byl za sledované období věnován (obrázek 22). Druhou částí těchto statistik je naopak práce a zaměstnanci, kteří ji vykonávali, zase seřazení dle věnovaného času (obrázek 20).

Support	Mlcak
Bugzilla	9:51 h
Maily	7:16 h
Testování	2:47 h
Přestávka	1:00 h
Suma	20:54 h

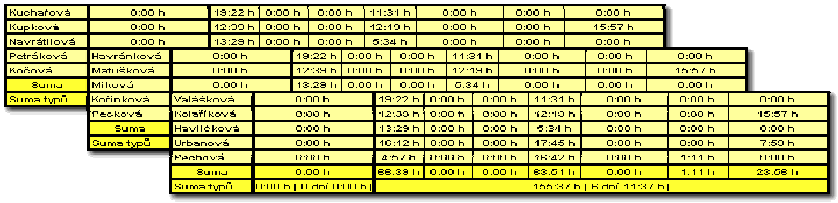
Obr. 22: *Pořadová statistika zobrazující zaměstnance, seřazená dle času strávených na vykonaných pracích*

Support	Bugzilla
Mlcak	9:51 h
Kropac	6:52 h
Janata	5:48 h
Stavik	5:14 h
Suma	27:45 h

Obr. 21: *Pořadová statistika práce bugzilla dle časů strávených všemi zaměstnanci, kteří ji vykonávali*

Zálohování správa a anonymizace

Systém umožňuje také zálohování celé databáze včetně veškerých dat tak, jak jsou v ní uloženy nebo anonymizovaně. Anonymizace spočívá ve smazání hesel všech registrovaných osob (je použito přednastavené heslo), smazání čísel stolů u kterých pracovníci sedí, a anonymizaci změnou jejich jmen. Jména jsou nahrazena zcela jinými jmény náhodně vygenerovanými skrze externí program napsaný v jazyce Python (který je také součástí této práce). Ilustrace ukazuje tři výstupy po anonymizaci. První řádek každé tabulky (jména Kuchařová, Havránková a Valášková) zachycuje tři po sobě jdoucí anonymizace jediného reálného zaměstnance. Jak je vidět, údaje o času stráveném na pracích zůstal zachován.

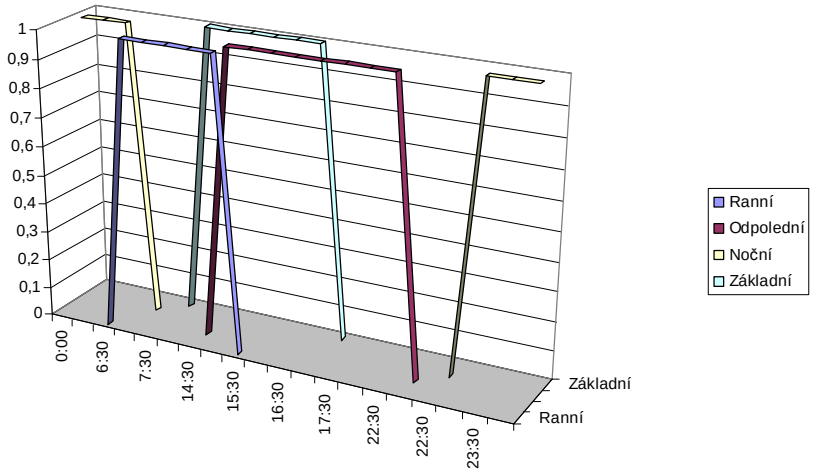


Obr. 22: *Anonymizované záznamy poskytují pro statistiky stále správné údaje*

Směnný provoz a jeho podpora v systému

Firma pracuje ve směnném provozu. Před samotným přihlášením na práci je tedy nutné, aby se zaměstnanec přihlásil na směnu, ve které bude pracovat. Volba směny přímo od zaměstnance ve

svých důsledcích může eliminovat chyby v plánu směn, nebo naopak příchod zaměstnance na nenaplánovanou směnu. Ilustrace 25 zobrazuje fuzzy množiny směn, které jsou v systému implementovány. Při registraci nové směny do systému je zadán její začátek a konec. Fuzzy množina začíná 30 minut před začátkem a končí 30 minut před koncem dané směny. Její pokles je v tomto čase rovnoměrný po minutách od 100% po 0% (systém hodnotu po výpočtu normalizuje a zaokrouhluje na celá procenta dolů). Při přihlašování na směnu systém odhaduje nejpravděpodobnější směny, na které se zaměstnanec hlásí dle času jeho přihlášení do systému a dne (pracovní den či víkend). Systém zobrazí procentuální ohodnocení možných směn (ilustrace 26) a v pozdějších fázích vývoje systému budou zobrazeny především tyto směny (ostatní budou dostupné na požádání).



Obr. 23: *Fuzzy množiny příslušnosti pracovníka do směn dle času přihlášení*

Ranní (7:00-15:30) 33%	Základní (8:00-17:00) 33%
Odpolední (15:00-23:00) 0%	Noční (23:00-7:00) 0%
Víkend ranní (7:00-15:00) 0%	Víkend odpolední (15:00-23:00) 0%
Víkend noční (23:00 7:00) 0%	Normální provoz (8:00 17:00) 33%
Víkend poledni (11:00-19:00) 0%	Odpolední zkrácená (15:00-21:30) 0%

Obr. 24: *Tabulka výběru směn pro pracovníka hlasícího se v pracovní den v 10:00*

Lokalizace systému

Části systému, které jsem vyvíjel osobně, jsou plně lokalizovány do českého a anglického jazyka. Některé moduly, na jejichž tvorbě jsem se nepodílel, nemusí být lokalizovány v plné míře, nebo můžou obsahovat chyby. O těchto modulech i o jejich odpovědnosti se zmíním dále v textu.

Souhrn stráveného času za období od 01.01.2007 do 07.01.2007 (Celkem 7 dní)

Support	Fax	Telefony	Maily	Testování	Update	Bugzilla	Manuály	Avir	Školení	Servis	Přestávka	Angličtina	Studium	Suma
Suma	4:23 h	335:25 h	125:07 h	62:33 h	16:17 h	27:45 h	2:38 h	2:42 h	0:00 h	0:00 h	7:55 h	0:00 h	3:40 h	Suma celkem
Suma typů	464:55 h (19 dní 8:55 h)			106:35 h (4 dny 10:35 h)			5:20 h (0 dní 5:20 h)		0:00 h (0 dní 0:00 h)		11:35 h (0 dní 11:35 h)			588:24 h (24 dní 12:24 h)

Sum of time spent in period from 01.01.2007 to 07.01.2007 (Together 7 days)

Support	Fax	Phones	Mails	Testing	Update	Bugzilla	Manuals	Avir	Schooling	Service	Brake	English	Study	Sum
Sum	4:23 h	335:25 h	125:07 h	62:33 h	16:17 h	27:45 h	2:38 h	2:42 h	0:00 h	0:00 h	7:55 h	0:00 h	3:40 h	Total sum
Sum of types	464:55 h (19 days 8:55 h)			106:35 h (4 days 10:35 h)			5:20 h (0 days 5:20 h)		0:00 h (0 days 0:00 h)		11:35 h (0 days 11:35 h)			588:24 h (24 days 12:24 h)

Obr. 25: *Příklad lokalizovaných statistik do češtiny a angličtiny*

Lokalizace je technicky založena na vložení správného lokalizačního souboru při načítání jakékoliv stánky (krom přihlašovací). Lokalizační soubory jsou ve své podstatě definice PHP konstant. Lokalizační soubor češtiny lze najít v \includes\trans_cz.inc, angličtinu v \includes\trans_en.inc. Pokud by měl systém podporovat ještě jiný jazyk, bylo by pouze potřeba přeložit jeden z těchto souborů a zavést volbu tohoto jazyka do výběrů. Jazyk systému lze nastavit při přihlašování a poté kdekoliv v systému (vlajka jazyka). Po volbě jazyka je tato informace zaznamenána do cookies prohlížeče. Je tedy nutné aby prohlížeč, který uživatel používá, cookies podporoval a měl je zapnutá.

Lokalizace se však netýká pouze textů v systému. Do angličtiny jsou přeloženy manuály pro pracovníky vystupující v různých rolích, obrázkové grafy, statistiky a samozřejmě je lokalizované i menu.

Obr. 26: *Menu lokalizované v češtině a angličtině*

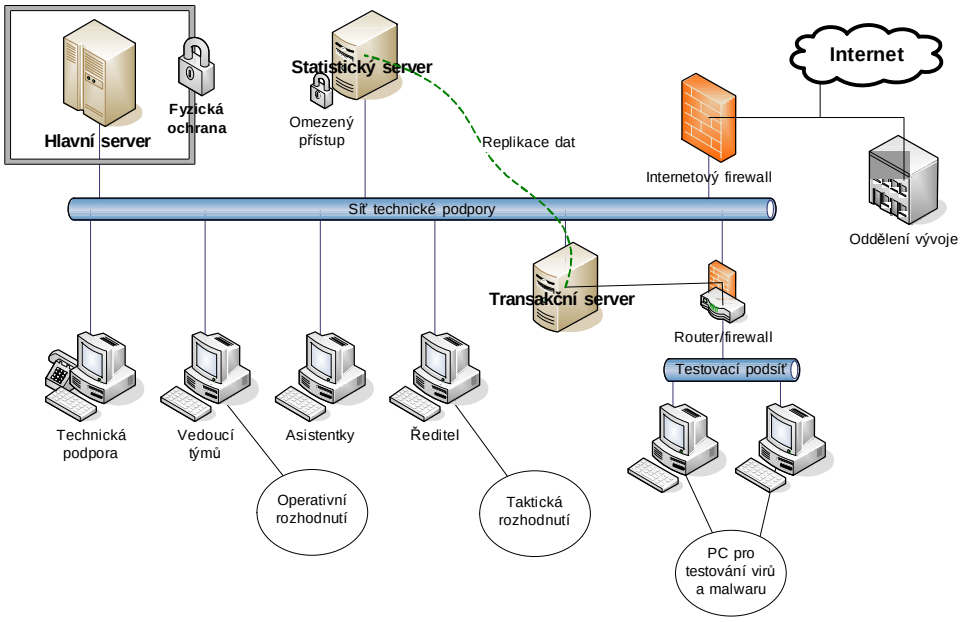
Moduly

Funkční jádro systému poskytuje především možnost zaznamenávat práce a generovat z těchto prací statistiky a věci s tímto režijně související (správa uživatelů, zálohování databáze, změna hesel atp.). Moduly nebyly vyvíjeny pouze mnou, ale považuji za vhodné je zmínit alespoň prostřednictvím

příloh, které popisují práci se systémem. Více o práci s implementovanými moduly lze najít v příloze 1.

4.3 Popis topologie sítě a plán rozvoje informačního systému

Následující ilustrace zobrazuje model pracoviště s vyznačenými pracovními úseky a důležitým hardwarem.



Obr. 27: *Model topologie sítě technické podpory*

Topologii sítě lze s určitou dávkou abstrakce popsat dvěma oddělenými sítěmi (Síť technické podpory a Testovací podsíť). V **Testovací podsíti** zaměstnanci pracují se vzorky aktivních virů a hrozí v ní tedy reálné nebezpečí napadení ostatních počítačů. Aktivní prvek (na ilustraci označen jako router/Firewall) brání jakémukoliv přístupu z Testovací podsítě do Sítě technické podpory. Z této podsítě však musí být zpřístupněn **Transakční server** (ten uchovává informace o práci, kterou zaměstnanec zvolil).

V **Síti technické podpory** pracují všechny ostatní role (pracovníci technické podpory, vedoucí týmů, asistentky, ředitel a ostatní nepracovní role). Tito lidé mají také přístup k Transakčnímu serveru. Hlavní server je fyzicky oddělený stroj schraňující všechna aktuální data a zálohy. Na ilustraci je vpravo naznačeno i existující síťové spojení s oddělením vývoje.

Střednědobý záměr počítá s nasazením **Statistického serveru**, který bude přes Síť technické podpory replikovat data z Transakčního serveru a při získávání statistik (a jiných souhrnných informací z IS), které nemají aktuální charakter (v rámci hodin nebo dní), poskytne aplikaci svá data. Toto řešení má několik výhod a jeho nasazení ve střednědobém horizontu má svá opodstatnění. Především jde o:

1. **Rychlost odezvy** serveru na vkládací dotazy. Transakční server je při generování statistik za dlouhá období nucen počítat řádově s tisíci záznamů. To způsobuje zatížení (především procesoru) stroje, na němž server běží.

2. **Větší ochranu údajů**. Při replikaci dat a uchovávání historických dat pouze na zabezpečenějším Statistickém serveru bude systém bezpečnější (alespoň co se týče množství potencionálně kompromitovaných dat v případě neautorizovaného průniku do systému). Zabezpečení statistického serveru může spočívat v silnější autentizaci uživatele, ale i v omezeném okruhu počítačů, které budou moci statistiky získávat (typicky to jsou pouze vedoucí týmů a ředitel).

Nasazení Statistického serveru není nutné v počátečních fázích implementace, protože ještě nebude dostatek historických dat, která by měl zpracovávat. Statistický server tedy může být zařazen až po půl roce fungování Transakčního serveru.

Dlouhodobý záměr počítá s integrací dalších funkcí do IS a použitím dat z dalších externích systémů, jako jsou například data z **elektronického příchodového systému**. IS může sledovat informaci o příchodu pracovníka do práce (jeho přihlášení zaměstnaneckou kartou) a po limitní době (například 15 minut od příchodu) zobrazit vedoucímu týmu varování, že pracovník sice v budově je, ale zatím se nepřihlásil do systému (nezvolil žádnou práci). IS také může sledovat odchod pracovníků na oběd (na příchodovém systému je přímo volba „Oběd“, ale i „Lékař“ nebo „Dovolená“) a tuto informaci reflektovat do dat uchovávaných v databázi IS. Poté se tedy nemůže stát, že se pracovník zapomene odhlásit při odchodu na oběd a zůstane přihlášen na práci, aniž by byl o této chybě informován.

Dalším záměrem je **integrace systému rozvrhu směn**. Směny potom mohou být zpětně kontrolovány a mohou podávat informace jak operativního, tak i taktického charakteru. Operativní charakter mohou mít informace o příchodu konkrétních pracovníků v konkrétní den na jistou směnu (ranní, odpolední, noční), nebo ověření, zda zaměstnanec, který má odpolední směnu, opravdu přišel na pracoviště (popřípadě varování, že se nedostavil) atp. Taktickou podporu mohou nést informace o produktivitě práce jednotlivých zaměstnanců, kteří byli přiřazeni na konkrétních úkolech ve sledovaném období, a z toho vyplývající informace o ceně (počtu člověkohodin) potřebné pro splnění obdobného úkolu v budoucnosti.

Integrace systému hodnocení technické podpory. Hlavní náplní technické podpory je telefonický a emailový kontakt se zákazníky firmy. Tyto emaily i celá komunikace jsou sledovány a uchovávány pro pozdější vyhodnocení. IS by měl nabízet možnost vybrání pracovníků, jež chceme hodnotit, a ohodnocení s poznámkami, které bude později pracovníkovi předloženo jako zpětná vazba k jeho práci. S tím souvisí i návrh systému, který musí umožňovat přehrávání elektronické hlasové nahrávky ze serveru, konfiguraci hodnotícího formuláře, sledování emailové komunikace atp.

Integrace podpory pro testování. Především testeři by měli mít možnost zadávat do této části systému informace o probíhajícím testování a o jeho výsledcích. Nadřízení potom budou mít přehled o jejich pracovní náplni (a pracovním nasazení). Data navíc mohou sloužit přímo vývojářům, kteří budou moci sledovat stav testování jimi opravené chyby (respektive urgovat její brzké otestování). Základním požadavkem je prvotní převod papírových formulářů do elektronické podoby a jejich následná implementace v IS.

Jiným příkladem podpory testování může být modul usnadňující automatizované testování v prostředí informačního systému. Modul by umožňoval výběr druhu testu, jeho konfiguraci a následné spuštění s automatickým sběrem výsledků a jejich zobrazení testerovi.

5 Automatizace testovacích procesů

V této části práce popíši hlavní myšlenku možného rozšíření automatickým testováním aplikací. Následující text je převzat z mé bakalářské práce [8], v rámci které jsem tento testovací systém implementoval.

Hlavním požadavkem na systém automatického testování je možnost znovu opakovat vytvořené testy a navozovat je v pokud možno stále stejných podmínkách. Takto koncipovaný systém, který by měl k dispozici vhodné testovací aplikace, by mohl regresně testovat jakékoliv programové vybavení. Takové prostředí tedy bude centrem zátěžového regresního testování (to vyžaduje programování zátěžově testujících aplikací), které je zadavatelem požadováno. Tento systém jsem navrhl a implementoval jako součást své bakalářské práce [8] a v následujícím textu shrnu hlavní myšlenky, koncept a přínosy systému.

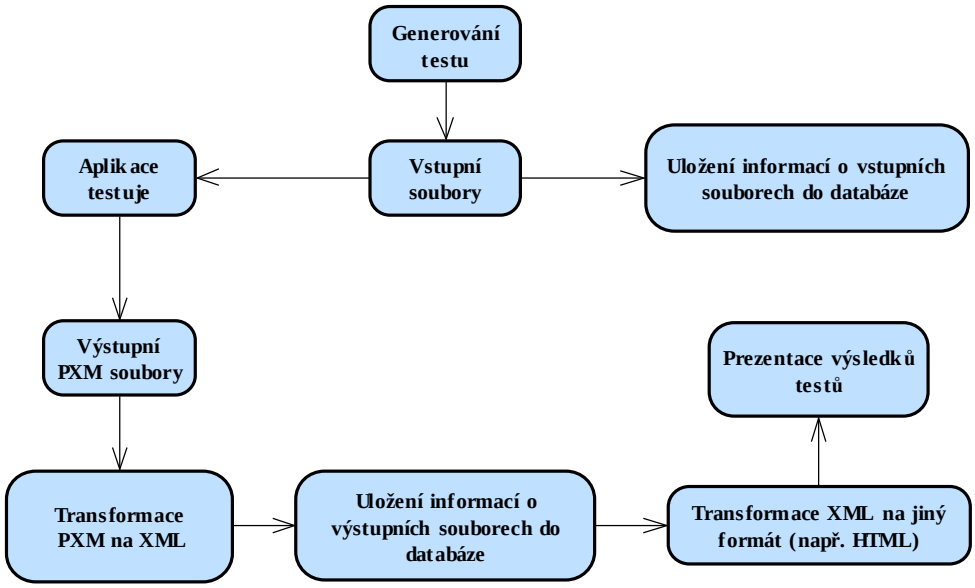
Návrh části systému pro automatické testování

Automatický testovací systém předpokládá existenci testovacích programů, které umožňují běh bez zásahu uživatele. Několik takovýchto programů vyvinul Bc. Ivo Řezníček jako součást své bakalářské práce [7].

Se spolupracovníky jsme se dohodli na formátu vstupních a výstupních souborů testovacích aplikací. Tímto formátem je XML, pro který existuje podpora ve formě analyzátorů v mnoha programovacích jazycích. XML je široce podporován prohlížeči a díky své formě bez sémantiky je zvláště vhodný jako formát pro výstupní informace. Síla XSLT (transformací XML) spočívá v možnosti generování různých druhů dokumentů. Mezi obvyklé transformace XML patří převod na:

1. XHTML či
2. prostý text.

S použitím formátovacích objektů (XSL-FO) je možné z XML vytvořit soubor formátu PDF, RTF, JavaHelp, HTMLHelp, nebo Postskript. Ze zjevných důvodů však bohužel není možné výstupní soubory generovat testovací aplikací přímo ve formátu XML. Abychom se s tímto problémem vypořádali, domluvili jsme se s kolegy na jiném formátu, který bude do XML snadno převoditelný (dále jen Pseudo XML, čili PXM). Na následující ilustraci je zobrazena abstrakce procesu testování:



Obr. 28: *Jednotlivé kroky automatizovaného testování*

Prvním krokem při testování je použití generátoru testu příslušejícího k testovací aplikaci. Snažil jsem se navrhnout systém i design webové aplikace (ve které jsem tento systém implementoval) tak, aby bylo generování vstupního souboru hlavním vstupním bodem. V systému si pracovník vytvoří příslušný test, což znamená vytvoření vstupního XML souboru pro testovací aplikaci. Záznam o tomto souboru se uloží do databáze a soubor se přesune do úložiště generovaných vstupních souborů.

Výhodami použití generátoru jsou jasný první krok při zahájení testování, jednoduché (rychlé) vytvoření komplexních testů a odstranění potencionálních chyb při vytváření vstupního souboru. Tyto chyby mohou být, podle mých zkušeností, v zásadě trojího druhu.

Pracovník nezná zásady XML, nebo se nevyzná ve značkách vstupního souboru. Značky musí být samozřejmě do jisté míry dané, ale některé se budou lišit podle poslání aplikace. Aplikace kopírující soubory bude mít značku vyjadřující vstupní a výstupní adresář, kdežto testovací aplikace firewallu je nepotřebuje. Značky, které by mohly být ve všech vstupních souborech stejně pojmenovány, jsou například první otevírající a poslední uzavírající značka (ze zřejmých důvodů je velmi žádoucí, aby byla tato značka pro všechny soubory stejná), značka pro nastavení parametrů a informační značky společné pro jakékoliv testy. Těmi jsou např. značka pro vstupní a výstupní soubor, či značka určující testovací aplikaci, pro kterou je vstup určen.

Druhou chybou pracovníků může být překlep. XML rozlišuje v názvech značek velká a malá písmena. Není tedy možné očekávat správnost při chybném zadání velikosti znaku nebo písmene.

Třetí příčinou potíží mohou být implementační specifika samotných testovacích aplikací nebo okolního prostředí (například operačního systému). Program se může například snažit o otevření příliš mnoha vláken nebo síťových spojení (například v řádu miliard, podle přijímaného datového typu). Programátor totiž vůbec nemusel provést testování, zda je vůbec možné otevřít takové množství spojení, či zda při pokusu otevřít velké množství (například milion) vláken neskončí aplikace chybou (nebo nezhavaruje).

Všechna tato omezení lze zadat do generátoru, který nedovolí vytvořit test vedoucí k jistému nezdaru. Pracovník si vybere testovací vstup a spustí ho s příslušnou aplikací. Díky uložení všech testovacích vstupů na jednom místě je zřejmé, kde soubory hledat. Při použití databáze vstupů je možné jednoduše vybírat informace o použitelných vstupních souborech. Aplikace během testování ukládá informace do výstupního PXM souboru, který uloží do předem určeného adresáře. Tím předejdeme nepořádku ve výstupních souborech, které pracovníci zapomenou přesunout do úložiště. Po ukončení testu, či jednou za daný časový úsek, je spuštěna aplikace, která převede PXM soubory do formátu XML a uloží informaci o výsledcích testů do databáze. XML soubory mohou být, jak jsem již předeslal, transformovány do sémantického jazyka. Tím je například HTML při použití XSLT transformačního přepisu. Tato forma výstupu může mít zvýrazněny některé podstatné části testu (například výsledek, či dobu trvání testu) a může tím usnadnit čtenáři získávání informací o výsledku.

6 Závěr

V této práci jsem se snažil shrnout teorii zabývající se testováním a modelováním podnikových informačních systémů. Zvláštní pozornost jsem věnoval metodice RUP, v současné době nejprogresivnějšímu procesu vývoje software, která využívá celou řadu pokročilých technik umožňujících rychlý vývoj aplikací. Ve třetí a čtvrté kapitole jsem popsal potřeby Oddělení technické podpory, včetně rozdělení zaměstnanců do uživatelských rolí. Systém již nyní spravuje větší počet hierarchicky sdružených rolí, ale je samozřejmě možné přidat i role další. Hlavní informací, kterou systém o všech zaměstnancích poskytuje, jsou podrobné záznamy o prováděných pracích. Z těchto záznamů je možné generovat několik druhů statistik, které jsou cenným zdrojem informací o dělbě práce a potažmo o její ceně. V páté kapitole jsem stručně popsal proces, který může pomoci automatizovat testovací procesy, což povede k výrazné úspoře velmi cenného času. Tato automatizace probíhá za použití informačního systému a speciálních testovacích aplikací, které budou se systémem komunikovat pomocí XML dokumentů a budou sloužit k podpoře testování implementovaného informačního systému. Testovací rozšíření však není v současné době do systému implementováno, jde pouze o možnost jednoduchého rozšíření o tuto funkcionalitu. Tento systém jsem navrhl a implementoval již dříve v rámci své bakalářské práce [8].

V současné době je informační systém pro sběr informací o pracích používán ve firmě již více než rok. V počátečních fázích iterativního vývoje se jednalo především o prioritní funkce sběru informací o pracích, v dalších fázích byly přidány funkce pro opravu chyb, základní statistiky a další moduly, které jsou vyvíjeny i jinými pracovníky firmy. Za tuto dobu bylo denně do databáze uloženo v průměru 130 záznamů o změně práce, což ve výsledku znamená dohromady přes 40000 záznamů. Dále bylo za tuto dobu vloženo přes 14000 záznamů o směnách, přes 3000 přihlášek na oběd a přes 200 žádostí o dovolenou. Při psaní této diplomové práce jsem si uvědomil především důležitost dokumentace vývoje a potřebu správy požadavků zákazníka.

Literatura

[1] Patton, R.: *Testování softwaru*. Computer press, Praha 4, CZ, 2002. ISBN 80-7226-636-5.

[2] Whittaker, J. A.: *How to break software: a practical guide to testing*. Addison-Wesley, EN, 2003. ISBN: 0-201-79619-8

[3] Ullman L.: *Php a MySQL*. Computer press, Brno, CZ, 2004. ISBN: 80-251-0063-4.

[4] Whittaker J. A., Herbert H. H.: *How to Break Software Security*. Addison-Wesley, EN, 2004. ISBN: 0-321-19433-0.

[5] Bieliková M.: *Úvod do softvérového inžinierstva*. Skripta poskytnuty na přednáškách předmětu IUS FIT VUT v Brně, SK, 2003.

[6] Koch, M.: *Skripta Manažerské informační systémy*. Akademické nakladatelství CERM, s.r.o. Brno, CZ, září 2006. ISBN: 80-214-3262-4.

[7] Řezníček, I.: *Návrh systému automatizovaného testování antivirových programů*. Bakalářská práce na Fakultě informačních technologií VUT v Brně. Vedoucí bakalářské práce Doc. Ing. Zdeňka Rábová.

[8] Boháček, P.: *Automatizace testovacích procesů*. Bakalářská práce na Fakultě informačních technologií VUT v Brně. Vedoucí bakalářské práce Doc. Ing. Zdeňka Rábová.

[9] Wikipedie: *IBM Rational Unified Process* [online]. 2006 [cit. 28-12-2006]. Dostupný z WWW: <http://en.wikipedia.org/wiki/Rational_Unified_Process>.

[10] Vondrák, I.: *Úvod do softwarového inženýrství v. 1.1* [online]. 2002 [cit. 28-12-2006]. Dostupný z WWW: <http://vondrak.cs.vsb.cz/download/Uvod_do_softwaroveho_inzenyrstvi.pdf>

[11] Perlík, L.: *Základy softwarového inženýrství (KIV/ZSWI 2005)* [online]. 3. ledna 2005 [cit. 28-12-2006]. Dostupný z WWW: <http://www.kiv.zcu.cz/~luki/vyuka/zswi/predn/zswi2005.pdf>.

[12] Hajdin, T.: *Agilní metodiky vývoje software* [online]. Brno, květen 2005 [cit. 28-12-2006]. 77 s. Diplomová práce na Fakultě informatiky Masarykovy univerzity v Brně. Vedoucí diplomové práce Mgr. Barbora Zimmerová. Dostupný z WWW: <http://is.muni.cz/th/39440/fi_m/dp.pdf>.

Seznam příloh

Rejstřík