VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

# SOFTWAROVÁ PODPORA PROJEKTOVÉHO ŘÍZENÍ

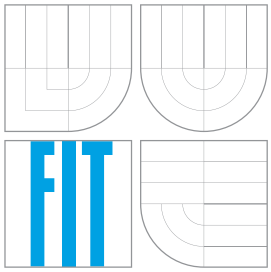
DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. VOJTĚCH MATES

BRNO 2008

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

# SOFTWAROVÁ PODPORA PROJEKTOVÉHO ŘÍZENÍ

SOFTWARE SUPPORT FOR PROJECT MANAGEMENT

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — Bc. VOJTĚCH MATES
AUTHOR

VEDOUCÍ PRÁCE — Ing. ŠÁRKA KVĚTOŇOVÁ
SUPERVISOR

BRNO 2008

# Abstrakt

Hlavním cílem práce je vytvoření aplikace pro podporu projektového řízení. Tato aplikace obsahuje základní funkcionalitu v součastnosti dostupných aplikaci, které se zabývají projektovým řízením, ale navíc přidává funkce, jako jsou podpora týmové komunikace, zavedení specifických rolí v systému(projektoví manažeři, techničtí speciálisté, správci zdrojů, běžní pracovníci) atd. Systém je také schopen odhadovat délku a náklady úkolů na základě databáze znalostí získané z již realizovaných projektů.



# Klíčová slova

projektové řízení, Microsoft Project, OpenProj, GanntProject, KPlato, ASP.NET



# Abstract

The main purpose of this work is to create an application for project management support. This application contains basic functionality of current available applications which deal with project management, but moreover adds functions such as team communication support, establishing specific roles in the system (project managers, technical specialists, resource managers, regular employees), etc. The proposed system is able to estimate time and cost of task based on knowledge database created from previous realized project.



# Keywords

Project Management, Microsoft Project, OpenProj, GanntProject, KPlato, ASP.NET



# Citace

Vojtěch Mates: Softwarová podpora projektového řízení, diplomová práce, Brno, FIT VUT v Brně, 2008

# Softwarová podpora projektového řízení

## Prohlášení

Prohlašuji, že jsem tuto diplomovou práce vypracoval samostatně pod vedením paní Ing. Šárky Květoňové.

.....
Vojtěch Mates
14. května 2008

## Poděkování

Na tomto místě bych rád poděkoval vedoucí své diplomové práce Ing. Šárce Květoňové.

# Obsah

# Kapitola 1

# Úvod

Za dobu své dosavadní praxe se neustále potýkám s mnoha problémy vznikající převážně ze špatné organizace. Jsou jimi zejména nejasná definice zodpovědnosti za úkol, nedodržení termínů, pozdní reakce na vzniklý problém během projektu atd. Bohužel se mi nepodařilo najít vhodný systém pro řešení těchto problémů, který by vyhověl mým požadavkům, a proto jsem se jej rozhodl vytvořit v rámci této diplomové práce.

Tématem této práce tedy bude problematika softwarové podpory projektového řízení a týmové spolupráce. Výsledkem bude realizovaný software, který bude napomáhat při řízení projektu zejména v oblasti plánování a komunikace.

V první části této práce bude obecně problematika vysvětlena projektového řízení. Budou zde vysvětleny nejvýznamnější pojmy, které se týkají projektového řízení obecně, neboť uvedení do problematiky je vhodné pro lepší pochopení výsledných požadavků na systém.

V následující kapitole pak bude provedeno představení již existujícího software a následně jeho porovnání. Na základě srovnání jednotlivých produktů budou následně stanoveny požadavky na výsledný produkt.

V další části budou po provedení analýzy vytyčeny základní požadavky, které by měl výsledný systém splňovat. Dále bude představena vybraná technologie a vysvětlen základní model této aplikace. V zápětí bude proveden návrh aplikace, který bude zohledňovat veškeré požadavky, které byly vytyčeny již dříve.

V části implementace bude proveden nástin realizace některých vybraných operací. Dále bude popsán návrh grafického uživatelského rozhraní (zejména základní logika ovládání aplikace). Posléze bude diskutováno možné rozšíření stávající aplikace.

## 1.1 Cíle práce

Cílem této práce je realizovat systém, který by řešil některé problémy vyskytující se v projektovém řízení. Tento systém měl za úkol převzít nejdůležitější funkcionalitu z již existujícího software, ale zároveň jej v určitých oblastech vylepšit.

Systém je postaven od základu na víceuživatelském modelu. Je zde jasná definice rolí v systému. Díky tomuto modelu může dojít k významně větší paralelizaci při tvorbě plánu, což může významně urychlit proces plánování.

Cílem návrhu architektury bylo umožnit průběžný sběr dat a později tato data využít pro plánování. Část dat (kvalifikace lidí, zdrojů, míst, požadavky expertů na provedení úkolu) se do systému dostává nezávisle na projektu po celou dobu chodu organizace. Tato data jsou pak opakovaně využívána při tvorbě plánu pro konkrétní projekt.

Dalším cílem bylo získaná data maximálně zužitkovat. Neboť tím, že se do systému budou centralizovat znalostí všech zaměstnanců, systém bude moci tyto informace agregovat a díky tomu například tvořit odhady délky práce či nákladu pomocí dat z již realizovaných projektů, či doporučit nejzkušenějšího pracovníka pro daný úkol. Výhodou z hlediska uživatele pak bude to, že nemusí do systému zadávat žádné speciální informace, ani se o tuto část nějakým způsobem starat. Systému bude postačovat, když uživatelé budou systém normálně používat.

# Kapitola 2

# Projektové řízení

Hlavní podstatou projektového řízení je činnost, jejímž posláním je splnění stanoveného cíle, který byl v projektu vytyčen. Tento cíl může být pokaždé jiný. Někdy je prioritou zejména rychlost provedení (například při uvedení nového výrobku dříve než konkurence), jindy zase náklady (zejména u menších projektu, či neziskových organizací) a v některých případech nám záleží zejména na precizním provedení (u kritických systému je velký důraz kladen i na formální ověření správnosti, nicméně toto se pak samozřejmě promítne do ceny).

Cílem této kapitoly je stručným způsobem seznámit čtenáře s touto problematikou. Detailnější informace je možno získat v těchto článcích: [9], [10], [11], [12], [13] případně v těchto publikacích: [2], [3], ze kterých jsem také čerpal.

## 2.1 Projekt

S projekty různého typu se setkáváme v našem životě prakticky denně. Snad každý člověk je v nějakém projektu zapojen a některé (většinou menší) projekty i sám řídí. Dalo by se také říci, že už jen řízení svého osobního času je vlastně realizací projektu a my se tedy vlastně stáváme projektovými manažery. Nicméně existují i velké projekty co do rozsahu činnosti či množství zdrojů, u kterých je toto řízení velmi obtížné, neboť obsahují velké množství úkolů a zdrojů, které je třeba koordinovat.

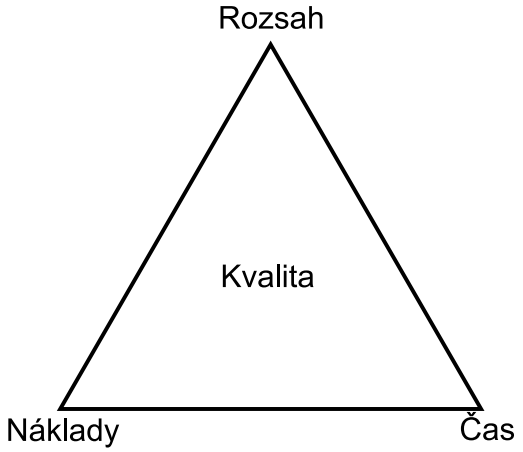
Projekt může být chápán jako sled na sebe navazujících událostí s daným počátkem a koncem. Projekt může také být definován jako organizovaná dočasná spolupráce mezi lidmi zaměřená na vytvoření jedinečného produktu nebo služby za určených podmínek a s omezenými zdroji.

Každý projekt je jedinečný. Nejedná se tedy o opakovanou činnost, jako je například výroba stejného výrobku nebo poskytování naprosto identické služby. Jedinečnost každého projektu je zajištěna minimálně časem, ve kterém má být projekt realizován.Také lidské zdroje jsou často jiné. Dokonce i kdyby byly stejné, je zde stále faktor času. Neboť lidé se stále vyvíjejí.

Důvodem vzniku projektu je obvykle nějaký cíl, který má být po jeho realizaci splněn. Tento proces ke splnění cíle má své datum začátku a konce a také vymezeny zdroje, které jsou potřebné k tomu, aby se proces uskutečnil. Je tedy svázán i s určitými omezeními.

## 2.2 Trojimperativ

Základní omezení, se kterým je možno se v projektovém řízení setkat, se nazývá Trojimperativ. Tento imperativ definuje tři, navzájem se ovlivňující, omezení. Jsou to čas, náklady a rozsah projektu. Trojimperativ bývá často znázorňován jako trojúhelník, kde jednotlivé strany reprezentují tři základní omezení. Časové omezení uvádí množství času, který máme k dispozici, abychom dokončili projekt. Náklady specifikují finanční prostředky, které máme k dispozici na realizaci projektu. Rozsah specifikuje, co musí být uděláno, abychom dokončili projekt.




Obrázek 2.1: Trojimperativ - Tripple Constraints

Zvýšíme-li rozsah projektu, zvednou se obvykle také náklady a čas. Pokud máme na projekt k dispozici příliš málo času, bude nutné snížit rozsah projektu nebo zvýšit náklady na projekt (například najmutím více pracovníků, najmutím externího subjektu atd.). Pokud máme na realizaci nízký rozpočet, může dojít k nárůstu potřebného času či snížení rozsahu projektu. Vždy je pro nás některé z těchto hledisek významnější.

Například pro firmu, která by plánovala propagaci nějaké společenské akce, by realizace reklamy, která by byla až po datu konání, znamenala, že projekt se nepovedl. Neboť propagovat akci, která se již udála nemá smysl. Je tedy primární snahou za každou cenu stihnout termín.

Pro armádu bude důležité, aby veškeré vybavení bylo důkladně otestováno a prověřeno i za cenu vzniku enormních nákladů či delšího trvání projektu.

Někdy je hlavním cíle dokončit rozsah za minimálních nákladu. V tomto případě se snažíme, abychom používali co nejlevnější zdroje i za cenu delší realizace projektu, či kvalitu provedení. Úlohou Projektového řízení je realizace projektu v rámci těchto definovaných omezení.

### 2.2.1 Čas

Jedním z častých pojmu v této oblasti je WBS neboli Work Breakdown Structure viz. [1]. Pracovní úsilí jednotlivých úkolu je odhadováno postupným seskupováním těchto úkolů do obecnějšího úkolu. Nakonec dostaneme odhad pro celkovou dobu trvání projektu. Mezi úkoly mohou být různé závislosti. Závislosti mezi těmito úkoly mohou ovlivnit celkovou dobu projektu. Důvod pro prodloužení projektu je čekání na výstupy z předchozího úkolu.

V některých případech není možný paralelismus kvůli těmto závislostem, a proto nelze ani zkrátit celkovou dobu projektu i za cenu zvýšení nákladů.

### 2.2.2 Náklady

Náklady na vývoj projektu závisí zejména na kvalitě zdrojů, efektivitě práce, materiálu, managementu rizik, vybavení, stupňování cen, nepřímých nákladů a v neposlední řadě i zisku.

### 2.2.3 Rozsah

Požadavky, které musí být vykonány, aby projekt splňoval konečná kritéria. Hlavní složkou rozsahu je kvalita koncového produktu. Množství času je rozděleno do jednotlivých úkolů a určuje celkovou kvalitu koncového produktu. Čím více času věnuje konkrétnímu úkolu, tím kvalitněji by měl být proveden.

## 2.3 Okolí projektu

Každý projekt je realizován v určitém prostředí neboli okolí projektu. Okolí a projekt na sebe vzájemně působí, mají mezi sebou vazby.

Projekt je obvykle realizován v určitém kulturním a sociálním prostředí. Toto prostředí zahrnuje různé zvyky a rozhodování osob zainteresovaných v projektu.

Mezinárodní a politická situace může rovněž velmi ovlivňovat průběh projektu. Odráží se zde celá řada aspektů od legislativního prostředí až po náboženství. Zejména v případě nadnárodních projektu je na toto nutné tomuto věnovat zvýšenou pozornost. S tímto souvisí i pracovní režim obvyklý v dané zemi nebo náboženské svátky.

Mezi další významné vlivy patří hospodářské a tržní prostředí, na kterých může záviset například chování konkurence.

Dále projekt mohou ovlivňovat faktory závisející na geografické lokaci. Jedná se například o počasí, nadmořskou výšku, pravděpodobnost výskytu záplav atd. Se změnou prostředí je třeba přehodnotit zejména rizika, tj. přepracovat analýzu rizik.

## 2.4 Úkol

Úkol je základní pracovní jednotka projektu. Jde o činnost v projektu, která má stanoven začátek a konec. Úkol obvykle charakterizuje název, doba trvání, práce, náklady a přiřazené zdroje.

Souhrnný úkol je takový úkol, který má podřízené úkoly. Sumarizuje informace svých podřízených úkolů. Umožňuje hierarchickou strukturalizaci projektu do logických celků. Opakovaný úkol – jedná se o periodicky se opakující se činnost. Periodou rozumíme časový interval mezi dvěma úkoly (den, týden atd.).

### 2.4.1 Vazby mezi úkoly

Při vytváření projektu není vhodné mít pouze seznam činností, ale je dobré mít zaznamenány i logické návaznosti. Úkol, který je nutné dokončit před započetím daného úkolu nazýváme předchůdcem. Úkol, který je třeba realizovat až po skončení daného úkolu nazýváme následník.

Obecně rozlišujeme tyto čtyři typy závislostí:

- Dokončení-Zahájení (FS) – úkol musí být zahájen nejdříve po dokončení předchůdce.
- Zahájení-Zahájení (SS) – úkol bude začínat ve stejnou dobu jako jeho předchůdce.
- Dokončení-Dokončení (FF) – úkol bude končit ve stejnou dobu jako předchůdce.
- Zahájení-Dokončení (SF) – úkol bude dokončen až po době zahájení předchůdce.

Nejpoužívanějším typem závislostí je Dokončení-Zahájení (FS). Zvolené vazby mají přímý vliv na délku projektu, protože nám udávají omezení z hlediska paralelizace jednotlivých činností.

## 2.5 Zdroj

Zdroje obecně rozlišujeme na hmotné (materiální) či lidské (zaměstnanci). Úkolem manažera je tyto zdroje efektivně využívat. Zdroje jsou nutné k tomu, aby vůbec mohla proběhnout činnost definovaná úkolem. Množství a výkonnost zdrojů velmi ovlivňuje zejména dobu pro vykonání činnosti.

Obecně můžeme rozlišit materiálové a pracovní zdroje. Pracovní zdroje jsou přímo potřeba pro provedení úkolu (typicky lidé a zařízení), materiálové přímo nesouvisí s danou činností, ale používají se během projektu (například papír, palivo atd.). Jako základní parametry zdroje obvykle uvádíme: název zdroje, maximální počet jednotek, cena zdroje, kalendář zdroje (v případě lidí se jedná o pracovní hodiny).

## 2.6 Náklady

Každý projekt má obvykle stanoven limit pro čerpání nákladů. Náklady se obvykle odvozují z předpokládaného rozsahu, využití materiálů, technologií, externích služeb atd.

Rozpočet projektu je základní součástí plánu. Můžeme rozlišit dva typy nákladů – fixní a variabilní. Příklad fixního nákladu by mohl být materiál (např. dlaždičky), příkladem variabilního nákladu pracující člověk (např. pokladač). Odhad fixního nákladu v případě materiálu bývá jednodušší, neboť nezávisí na čase (je to jednorázový náklad), v případě variabilního nákladu je třeba odhadnout délku trvání, neboť náklady se budou odvíjet od délky trvání. V projektu se také mohou objevovat náklady na použití. Typickým příkladem tohoto nákladu je pronájem.

## 2.7 Plánování rozpočtu

Množství finančních prostředků ovlivňuje prakticky celý projekt nejvíce. Neboť výší peněz lze ovlivnit jak čas dokončení (můžeme maximalizovat paralelizaci), tak rozsah projektu. Pro realizaci projektu je klíčovou otázkou, kolik bude stát jeho provedení. Musíme tedy odhadnout cenu projektu ještě před jeho vlastní realizací. Tento odhad je velmi důležitý. V případě, že odhadneme náklady příliš vysoké, nemusí k realizaci dojít, protože se to nevyplatí. Pokud odhadneme náklady příliš nízké, projekt sice bude zrealizován, ale nepřinese požadovaný zisk.

Kvalifikovaný odhad lze provést na základě zkušeností manažera s podobnými projekty. V případě, že nemáme k dispozici zkušeného manažera, můžeme povolat expertního pracovníka, aby nám se nám s tímto odhadem pomohl. Vždy je lepší použití interních

zdrojů. Interní pracovník bude levnější (minimalizace nákladů) a také bude mít lepší znalost okolí projektu, proto se dá očekávat, že jeho odhad bude reálnější. Další výhodou je lepší ověřitelnost zkušeností a kvality experta.

Odhad lze provádět i na základě analogie s podobným projektem. I když dva projekty nebudou nikdy úplně stejné (jiné okolí projektu, jiní lidé, jiné zdroje atd.), je toto velmi dobrý zdroj pro vytvoření odhadu. Existují i komerční databáze s daty realizovaných projektů.

U některých projektů se cena projektu odvíjí od ceny, kterou je ochoten zákazník zaplatit. Na základě ceny se pak specifikuje rozsah projektu. Uvedeme si menší příklad. Nechť máme na starost organizaci předávání cen se zajištěním rautu a doprovodného programu. Vzhledem k možných nákladům můžeme najmout známější (dražší) kapelu a v případě rautu můžeme cenou ovlivnit množství a kvalitu jídla (např. losos na víně bude výrazně dražší než chlebíčky). Kvalita realizace a rozsah jednotlivých dílčích činností bude tedy záležet na množství prostředků, které budeme mít k dispozici.

## 2.8 Oblasti řízení projektu

Každý projekt se skládá z několika činností. Tyto činnosti jsou běžnou součástí každého projektu a jejich správný průběh je důležitý pro jeho úspěšný běh.

- Definice cílů projektu
- Plánování projektu
- Řízení projektu a řízení lidí
- Sledování postupu prací na projektu
- Vyhodnocení a ukončení projektu

### 2.8.1 Definice cílů

Úkolem Definice cílů projektu je přesné vymezení a analyzování projektu. Tyto cíle obvykle bývají definovány na základě dlouhodobé strategie organizace. Špatné provedení této etapy může mít pro projekt fatální následky.

Definice cílů projektu obsahuje oficiální zadání projektu, včetně definovaných omezení času, nákladů, rozsahu, analýzu aktuální situace a historie projektu, vymezení všech zainteresovaných stran, jejich očekávání a vliv na projekt.

Výsledkem této fáze je co největší porozumění zadání projektu. Dále co se od tohoto projektu očekává a rovněž rozpoznání okolností, které mají na projekt nějaký vliv. Pokud je tato fáze dobře zvládnuta, je to dobrý předpoklad pro úspěšnou realizaci projektu. V případě, že dojde k špatnému stanovení cílů, celý projekt se bude ubírat špatným směrem, neboť na základě cílů se pak stanovují plány či rozpočet.

### 2.8.2 Plánování

Plánování projektu bychom mohli označit za jakousi simulaci toho, jak bude projekt probíhat. Toto plánování obsahuje písemný popis plnění již zmíněného trojimperativu, proto máme plány pro provedení projektu (hierarchická struktura činností), pro čas (časový harmonogram) i náklady (rozpočet projektu).

Plánování je úvodní etapou, tj. kvalita tohoto plánu rozhoduje o tom, zda bude projekt úspěšný či nikoliv. Plánování je obecně opakující se proces. Iterativně upravujeme stávající plán. Tyto změny se dějí zejména z důvodu změn, které se odehrávají při běhu projektu. Již při fázi plánování je důležité stanovit základní kontrolní mechanizmy. Abychom na odchylky plánu od reality zjistili a mohli se je pokusit řešit. V případě, že problém zjistíme dostatečně včas, může změnou plánu problém vyřešit, aniž by to mělo zásadní vliv na projekt jako takový.

Při každé změně v plánu obvykle dojde také změně základních plánovacích dokumentů, časového plánu a rozpočtu. Množství dokumentace se odvíjí od velikosti projektu, u malých projektů by přílišné množství dokumentace znamenalo spíše zátěž než pomoc. Dalo by se také říci, že čím složitější projekt, tím složitější bývá struktura dokumentace. Struktura dokumentace může být také stanovena interními směrnicemi společnosti nezávisle na velikosti projektu.

Abychom mohli lépe plánovat či kontrolovat celý projekt, je vhodné si jej rozdělit na jednotlivé činnosti. Tyto činnosti je možné do sebe zanořovat, čímž vznikne hierarchická struktura činností.

Plánování je obecně organizační proces pro vytváření a správu plánu. Plánování můžeme také označit za psychologický proces přemýšlení o aktivitách potřebných pro uskutečnění záměru dle nějakých měřítek. Je to základní vlastnost inteligentního chování. Tento myšlenkový proces je základem pro vytvoření a zjemňování plánu nebo integrace tohoto plánu s dalšími plány. Plánování kombinuje předvídání vývoje s přípravou scénářů, jak dosáhnout stanoveného cíle. Termín Plánování je často používán k popisu formálních procedur pro tvorbu dokumentů, diagramů, schůzek pro diskuzi klíčových témat, vytyčení cílů, které je třeba realizovat a celkové strategie.

Z praktického hlediska se jedná o nejdůležitější část. V případě, že v této fázi něco opomeneme, musí později dojít k přeplánování. Každé přeplánování může způsobit nepředvídatelné komplikace. Plán, který stanovíme, je závazný. Musíme jej tedy realizovat a každá významnější změna může znamenat i nový proces schvalovaní.

Při plánování musíme uvažovat v kontextu řízení rizik, kvality, lidských zdrojů či konfigurace (spolupráce mezi dříve zmíněnými oblastmi). Plánování se odvíjí od našeho hlavního strategického cíle. Na základě stanovení tohoto cíle tvoříme plán. Strategie se odvíjí na základě dlouhodobých vizích organizace.

Po definici základních částí definujeme WBS, ta nám určuje souhrn činností. Projekt je třeba si rozdělit na skupiny procesů, jednotlivé procesy a poté na jednotlivé dílčí činnosti. Jakmile rozdílíme projekt na několik částí, je vhodné si vytvořit kontrolní body. Ukončení fáze nazýváme milníkem a jedná se významný bod projektu.

Jakmile je vytvořen kompletní plán, můžeme provést specifikaci i kontrolního mechanizmu a reportovaní, abychom byli schopni zhodnotit skutečné plnění vůči plánu. Na stanovení kontrolních dokumentů, pravidelných kontrol a monitorovací řízení bychom měli pamatovat již ve fázi plánování.

Dalším krokem bývá založení projektového týmu. Projektový manažer obvykle zodpovídá za všechny členy týmu, a proto je i tato fáze velmi důležitá. Dále je také třeba určit kvalitativní standardy, které ovšem primárně závisí na zákazníkovi.

### 2.8.3 Řízení projektu a vedení lidí

Tato aktivita se vyskytuje v průběhu celého projektu. Po celou dobu je nutné pracovníky vést, kontrolovat, ale i motivovat. Organizační struktura se může odlišovat dle konkrétní

organizace. Projektový tým může vznikat i napříč organizační strukturou firmy a mohou do něj být zapojeny i externí pracovníci. Může se jednat o subdodavatele, konzultanty, případně zaměstnance našeho zákazníka. Také je zde problém, že každý člen týmu má jinou výkonnost a jiné znalosti vztahující se ke konkrétní činnosti. Proto neplatí, že zvýšíme-li např. desetkrát počet pracovníků, zkrátíme tím čas na desetinu, podrobněji například v knize [5]. Proto odhady netriviální práce lidí patří k nejtěžším a nejvíce rizikovým z hlediska odhadu délky činnosti.

Procesy pro řízení lidí zahrnují zejména nábor pracovníku, jejich pozdější vývoj a rozvoj týmu jako celku. Projekt je své podstaty dočasnou událostí, máme tedy velkou pravděpodobnost, že členy týmu budou pokaždé jiní lidé. Skupiny lidí pracující na projektu také závisí na jeho fázi. Například při tvorbě software v první fázi budeme spíše potřebovat analytiky, v další fázi více programátory a následně testery.

Důležitou otázkou je také stanovení pravomocí jednotlivých pracovníků a náplň jejich práce, aby bylo možno jasně definovat zodpovědnost. Je důležité si zajistit i možnost externích zdrojů, minimálně jako záložní řešení v případě nečekaných událostí. Externí pracovník má nevýhodu neznalosti okolního prostředí, což také může ovlivnit jeho produktivitu práce. Neboť musí dojít k jeho seznámení s prostředím a vyrovnáním s fungujícími procesy, na které před tím nemusel být zvyklý. Pokud najmeme externího zaměstnance musíme ještě obezřetněji vzít v úvahu bezpečnost, tj. co nejvíce omezit přístup ke zdrojům (např. databáze), které pracovník bezprostředně nepotřebuje.

V současné době je stále větší tendence se na lidské zdroje (human resources) dívat spíš jako na kapitál (human capital), který je třeba zhodnocovat. Neboť v případě některých typů organizací jsou schopnosti zaměstnanců to nejcennější, co mají (například u softwarových společností). Proto je třeba dbát i na další rozvoj pracovníků a podporu týmu jako celku.

Zajímavou metodou může být i rotace pracovních pozic mezi jednotlivými zaměstnanci, v případě, že je toto možné z hlediska jejich kvalifikace. Výhodou tohoto přístupu je, že pracovníci si vyzkouší role i jiných členů týmu, což může mít pak vliv na další spolupráci a celkový rozvoj pracovníka, který pak získává částečně i schopnosti svých spolupracovníku, což je velmi výhodné pro případnou zastupitelnost. Dalším důvodem pro tuto rotaci může být menší oblíbenost určitého typu práce nebo stereotyp.

Jednou z dalších problematik, souvisejících se řízením lidských zdrojů, je také vykazovaní činností. Způsob vykazovaní je obvykle stanoven buď směrnicemi organizace nebo je specifikován již ve fázi plánování.

Jednou z podstatných částí je zajištění funkční komunikace. Naším cílem tedy může být, aby se veškeré informace šířily co nejrychleji a byly určeny skutečně pro příjemce. Komunikace řešená pouze formou hromadného rozesílání všem zaměstnancům na jakoukoliv událost v organizaci není ideální řešení, protože je zbytečně spotřebován čas zaměstnanci na vytřízení pro ně nepodstatných zpráv.

### 2.8.4 Sledování průběhu prací na projektu

Jednou z nejdůležitějších činností manažera je sledování postupu prací na projektu. Pokud správně projekt monitorujeme, můžeme včas zareagovat na vznikající problém už v zárodku a učinit případná opatření, které mohou vliv problému na projekt snížit nebo úplně eliminovat.

Abychom mohli projekt dobře kontrolovat, je vhodné si jej rozdělit do menších celků. Dále je třeba stanovit přesné cíle jednotlivých částí a definovat případnou zodpovědnost

dalších pracovníků. Kontroluje se zejména časový průběh projektu a čerpání rozpočtu.

Kontrolovat můžeme pomocí schůzek či reportů. Je nutná kombinace obou těchto přístupů. Při osobní kontrole máme sice větší jistotu, že věci jsou manažerovi opravdu předkládány, tak jak doopravdy jsou a nejasnosti lze ihned odstranit. Nicméně tento druh kontroly je velmi časově náročný. Jak by měl správný report vypadat, si řeší většinou organizace pomocí vlastních směrnic.

Pokud se zjistí odchylky oproti plánu (například zpoždění úkolu na kritické cestě), je třeba se zamyslet, zda-li není nutné přeplánovat zbytek projektu, abychom, co nejvíce eliminovali potencionální komplikace. Cílem je pak upravit plán projektu tak, abychom dodrželi rozpočet, rozsah a čas projektu.

Manažer po každém přeplánování provádí časovou analýzu projektu. Důležitým výsledkem jsou časové rezervy u jednotlivých činností. Je třeba sledovat činnosti, u kterých jsou nízké časové rezervy, aby jejich následným přečerpáním nedošlo k prodloužení délky projektu.

Další činností je i zdrojová analýza. Postup projektu není závislý pouze na návaznostech činností a délce trvání, ale i na počtu pracovníků, které jsou pro jednotlivé činností nezbytní či jiných speciálních zdrojů. Řešíme tedy problém jak rozvrhnout realizaci projektu při omezených zdrojích, aby skončil v co nejkratším možném čase, případně jak naplánovat realizaci jednotlivých činností, abychom měli požadavky na zdroje přibližně rovnoměrné a zároveň zajistili dodržení termínu ukončení projektu. Při této analýze je tedy posuzována jak délka realizace, tak možnosti využití časových rezerv a případné zajištění dodatečných zdrojů pro zkrácení doby projektu.

Realizace libovolného projektu vyžaduje vynaložení určitých nákladů. Je přirozené tyto náklady co nejvíce minimalizovat. Znamená to rozvrhnout projekt tak, aby spotřeboval co nejméně prostředků. Toto ale bude samozřejmě ovlivňovat délku projektu.Většinou se tedy snažíme nalézt rozumný kompromis mezi co nejkratším časem a náklady.

Obecně můžeme rozlišovat náklady rostoucí se zkracováním činnosti a náklady rostoucí s dobou činnosti.

Při zkracování činnosti může docházet k růstu nákladů kvůli použití lepší technologie, výkonnější techniky či použití kvalifikovanějších pracovníků. Při zvýšení počtu zdrojů dochází ke zvýšení nákladu i kvůli nutností zavedení preciznějšího řízení (vyšší administrativní zátěž), případně zavedení správy a údržby synchronizačních prostředků (např. SVN), rovněž vrůstá nutnost častějších schůzek atd.

S délkou projektu souvisí i náklady, jako jsou pronájem prostor, energie, telefonní tarif či jiné často paušální služby. Tyto náklady musíme platit po celou dobu projektu. Pokud realizujeme více projektu zároveň muže dojít ke sdílení těchto nákladů.

Strategie většiny projektu je dokončit projekt co nejdříve s použitím převážně vlastních zdrojů, aby nedocházelo k přílišnému navýšení.

### 2.8.5 Vyhodnocení projektu

Na konci projektu provedeme jeho vyhodnocení a ukončení.Konkrétní čas a způsob vyhodnocení záleží na konkrétním druhu projektu. Rozdílný způsob vyhodnocení bude například u divadelní hry než při předání informačního systému.

Vyhodnocení divadelní hry lze provést nejdříve po několika představeních například formou anket a přečtení různých recenzí. U informačního systému se budou provádět spíše přejímací testy než vyhodnocovaní anket, jestli je systém správný.

Každý projekt by měl projít analýzou, zda proběhl, tak jak měl, co by se případně

dalo vylepšit a jaké neočekávané události se objevily a proč. Toto je velmi důležité při pozdějším řešení projektů s obdobnou náplní. Přestože je každý projekt unikátní, často se skládá z některých částí, které se mohou často opakovat i v dalších projektech a mohou tak být inspirací pro další plánování.

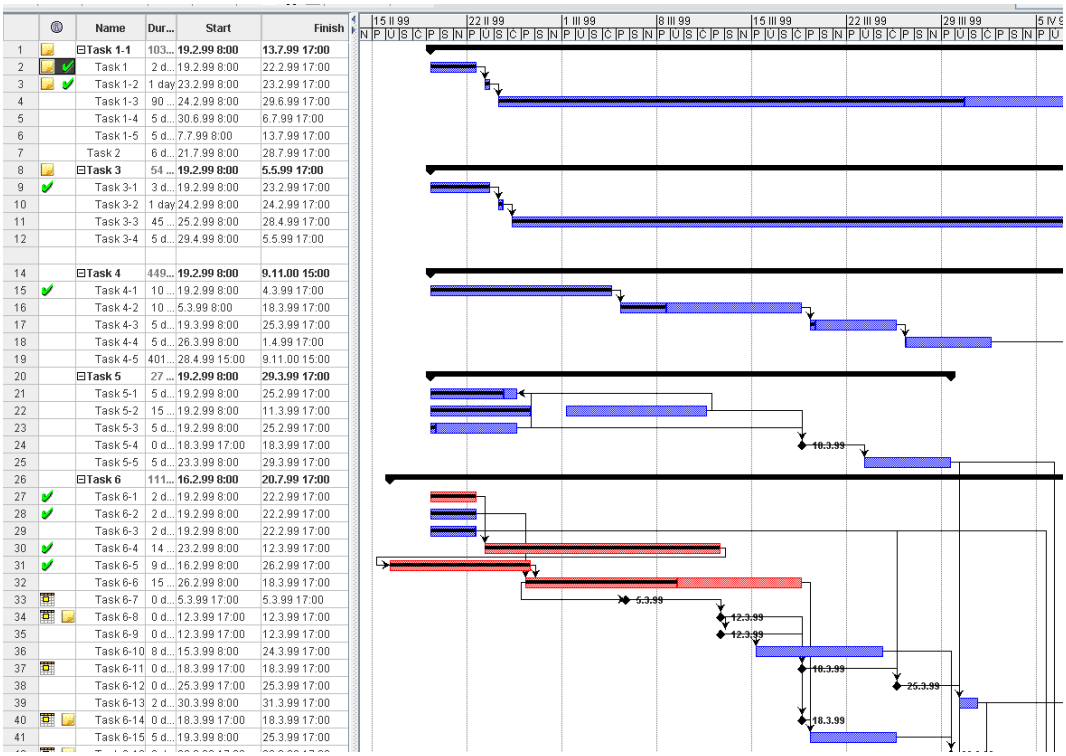
## 2.9 Vizualizace projektu

V projektovém řízení se setkáváme často s vizualizací WBS, či různých závislostí mezi úkoly. Toto bývá zobrazované nejčastěji různými typy grafu. Uvedeme se tedy nejvíce používané grafy, se kterými se můžeme běžně setkat.

### 2.9.1 Ganntův diagram (Gantt Chart)

Tento diagram je primárně určen pro znázorňování síťových grafů (vazby mezi úkoly) a hierarchických struktur (WBS). Ganttův diagram patří mezi nejoblíbenější způsob vizualizace úkolu v projektu. Umožňuje přehledně zobrazit aktuální stav projektu, směrný plán a aktuální plán (dobře viditelný je zejména rozdíl v čase mezi směrným plánem a aktuálním stavem).

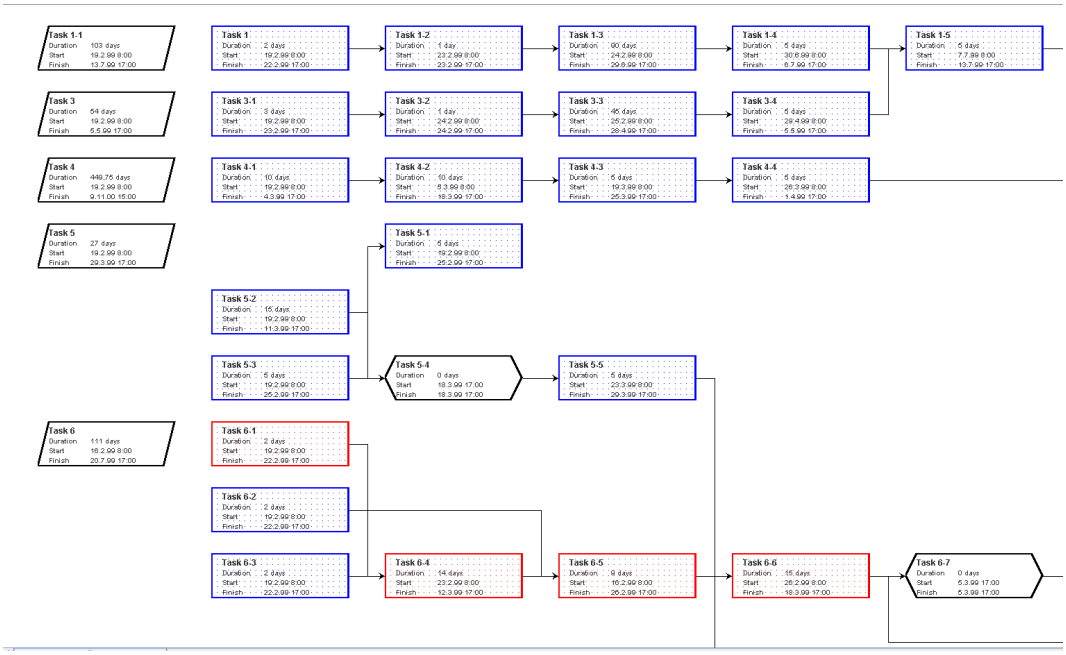
Ganttův diagram je oblíbený zejména pro přehlednost. Ihned jsou vidět závislosti mezi úkoly, či hierarchická struktura.




Obrázek 2.2: Ganttův diagram

### 2.9.2 Síťový diagram (Network Diagram)

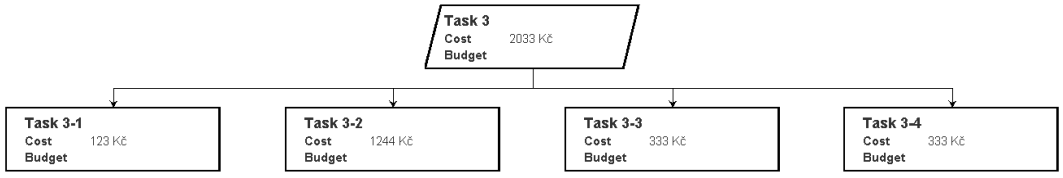
Postupným vkládáním činností do jiných činností vytváříme hierarchickou strukturu. Závislosti mezi činnostmi tvoří obecně grafovou strukturu. Pro přehledné znázornění tohoto grafu bývá poměrně často používám síťový diagram. Uzly tohoto grafu reprezentují činnosti a hrany vztahy mezi těmito činnostmi. Pokud potřebujeme vidět závislostmi mezi jednotlivými činnostmi je tento graf vhodným kandidátem.




Obrázek 2.3: Sítový diagram (Network Diagram)

### 2.9.3 WBS diagram

Hierarchická struktura tvoří strukturu acyklického grafu (stromu). Pokud chceme tento strom zobrazit, můžeme použít pro tyto účely speciálně vytvořen WBS diagram.




Obrázek 2.4: WBS diagram (WBS Diagram)

## 2.10 Management rizik

Neočekávané události se mohou objevit v každém projektu. Určitá míra rizika je přirozená. Je však třeba s riziky počítat a snažit se co nejvíce zmírnit jejich případný dopad.

Reálně řešíme několik základních kategorií rizik.

- Rizika, která se týkají projektového týmu, označujeme jako Interní rizika. Konkrétním případem může být například onemocnění pracovníka.
- Rizika vznikající z důvodu techniky označujeme jako technická. Můžeme mít například problémy s kompatibilitou mezi různými aplikacemi.
- Externí rizika jsou rizika mimo náš tým, například konkurenční firma.

Bohužel, ne všechna rizika lze předvídat, pokud ale provedeme opatření pro námi již předvídatelná rizika, můžeme se tím částečně bránit i proti rizikům, jež jsme si doposud neuvědomili. Například zálohou dat se bráníme nejen živelné pohromě, ale i útoku ze strany zaměstnance.

Abychom mohli rizikům předcházet, musíme je nejdříve identifikovat. Rizik je velmi mnoho a má cenu se zabývat pouze těmi, které jsou nejvíce pravděpodobné. Je tedy dobré si vytvořit seznam rizik. Následně ty méně významná z něj můžeme vyloučit, aby investice spojená s tvorbou opatření nebyla větší než skutečná škoda, která by mohla vzniknout.

### 2.10.1 Analýza rizika

Analýza rizika je velmi důležitou oblastí. Nejdříve identifikujeme rizika, následně se pokusíme navrhnout jak se s těmito riziky vypořádat. Můžeme například vytvořit alternativní plány založené na konkrétních hodnotách. Tento alternativní plán může vypadat přibližně takto:

V případě, že dojde ke zpoždění o 10% procent, přidej do týmu dalšího pracovníka, v případě, že o 20%, povolej externího experta atd.

Můžeme si tedy nastavit jisté limity, které pak spouštějí alternativní plány. Obecně jsou dva přístupy k vypořádání se s rizikem:

- reaktivní,
- proaktivní.

Reaktivní přístup se k této problematice staví tak, že riziko začne řešit, až nastane. Oproti tomu proaktivní se snaží již tvorbou plánu snížit pravděpodobnost výskytu rizika, či minimalizaci v případě, že by riziko mohlo nastat.

Analyzovat rizika můžeme například pomocí SWOT analýzy, tj. určíme silné a slabé stránky, příležitosti a hrozby. Identifikace rizik vychází zejména z předchozích zkušeností. Často se opírá o statistiky (viz. pojišťovnictví), tj. s jakou pravděpodobností se může daná událost stát na základě předchozích poznatků.

### 2.10.2 Krizové plány a eliminace rizika

Další významnou částí je stanovení krizových plánů, tj. co se má stát, pokud daná situace nastane. Důvodem stanovení krizových plánu je zejména snížení času na eliminaci vlivů nečekané události.

Jedním z dalších důležitých bodu je stanovit proces jak včas riziko rozpoznat, tj. například naplánovat pravidelné kontroly, abychom včas zjistili případné zpoždění některého úkolu.

Rizika můžeme eliminovat pomocí využívání ověřených zdrojů, tj. vybíráme si spíše ověřené technologie či pracovníky, zejména pokud jde o kritické části projektu. Pokud

použijeme méně ověřenou technologií (např. z důvodu výrazného snížení nákladů) je vhodné si zajistit alespoň technickou podporu od firmy, která tuto technologii dobře zná.

### 2.10.3 Odhadování a zabudování rezerv

Každý projekt se obecně skládá z několika etap, na každou etapu máme vymezen určitý plánovaný čas, máme tedy určen čas začátku a konce etapy. Důležité je, z čeho se odhad jednotlivých etap skládá. Obecně bychom mohli říci, že se skládá ze složky technologické a složky rizikové.

Technologická složka je reprezentována pouze časem týkající se pouze obsahu činnosti (např. čas pro postavení zdi). Naproti tomu složka riziková je ovlivňována zejména lidských faktorem. Prakticky to pro nás znamená, že je nutné zavést v plánu i určitou rezervu pro tuto činnost. Tato rezerva pak slouží k eliminaci nepředvídaných událostí, jako například nemoc pracovníka či poškození přístroje atd.

Časový odhad se tedy bude skládat z obou složek, tj. součet technologické a rizikové složky. Technologická složka bývá zpravidla nepříliš ovlivnitelná, riziková je ovšem velmi proměnlivá, neboť je velmi obtížné odhadnout práci člověka, kterého neznáme. Dle množství rizika se bude měnit i rozptyl doby potřebný pro vykonání činnosti.

Pro zapracování rezerv můžeme vzít v úvahu několik skutečností. Každá úroveň má vlastní rezervu. Čím více je úrovní, tím více času je vyhrazeno pro rezervy. Je dobré si uvědomit, že velikost rezervy závisí na tom, s jakou pravděpodobností musí být úkol dokončen v termínu. Záleží tedy velmi na vedoucím, který tyto rezervy stanovoval. V každé fázi však ještě přidáváme synchronizační rezervu.

Lidé brání své odhady proti škrtům. Je to přirozené, neboť je vždy lepší mít k dispozici více prostředků či času na realizaci projektu. V případě, že vedení řekne, že je třeba plán optimalizovat např. o 10%, pracovníci budou předkládat příští odhady již navýšené o těchto 10%, aby měli k dispozici dle jejich názoru dostatečný počet prostředků.

### 2.10.4 Parkinosonův projektový zákon

Tento zákon říká, že činnost trvá nejméně tak dlouho, jak dlouhý má přidělený časový interval. Abychom odůvodnili, proč tomu tak opravdu často bývá, uvedeme si menší příklad. Nechť činnost se skládá ze dvou na sebe navazujících úkolů. První činnost se opozdí o např. 2 dny, druhá činnost tedy musí začít s dvou denních zpožděním. Co ale v případě, že činnost ve skutečnosti skončí například o dva dny dříve? Logicky by navazující činnost měla začít o dva dny dříve oproti plánu.

Nicméně reálně se musí počítat s lidských faktorem. Tým se totiž pravděpodobně nebude chlubit s tím, že má již hotovo, aby mohl dostat další práci. Nemá k tomu dostatečnou motivaci. Většinou má stanovenou hodinovou mzdu, tj. čím více hodin stráví na projektu, tím vyšší mzda pro něj.

Další faktorem může být obava z toho, že vedení díky rychlejší realizaci příště upraví čas a náklady potřebné pro projekt směrem k horšímu.

Dalším důvodem může být blokace zdrojů. Dojde ke změně plánu a některé zdroje již mohou být alokovány jiným projektem.

Je tedy spoustu důvodu, proč nehlásit rychlejší realizaci a dochází tak k tomu, že náskok, který by na základě pravděpodobnosti měl nastat, je prakticky nulový, zatímco zpoždění se hromadí.

### 2.10.5 Studentův syndrom

Tento syndrom lze demonstrovat na příkladu. Po obdržení podavků na práci se studenti snaží o oddálení termínu či zmírnění zadání. Pokud se jim tyto požadavky podaří prosadit, získají časovou rezervu, protože odhadují, že prací jsou schopni dokončit v kratším čase. Práci tedy zahájí později o čas, který získali rezervou. V případě, že by odhad byl přesný a nedošlo k nečekaným událostem bylo vše v pořádku. Nicméně může dojít ke komplikacím. Student může onemocnět a problém je tu.

Tento syndrom se samozřejmě netýká pouze ke studentům, ale obecně všech pracovníku, kteří v domnění, že mají ještě rezervu, začnou s prací později než je stanovený termín a tím přijdou o přidělenou rezervu.

## 2.11 Optimalizace projektu

U každého projektu se snažíme, aby proběhl co nejefektivněji. Tento postup je vhodné provádět po každém významnějším přeplánování projektu, tj. nejen při počátečním plánování.

Důležitou vlastností projektů je tkz. kritická cesta, která vyjadřuje minimální čas nutný pro dokončení projektu. Tato cesta se může při přeplánování změnit. Vzhledem k tomu, že kritická cesta má asi nejvyšší vliv na celkovou dobu dokončení projektu, je nutné věnovat plnění úkolu na kritické cestě velkou pozornost, protože každé zpoždění úkolu na kritické cestě bude mít vliv na celkovou dobu trvání projektu.

Dále je nutné se starat o rovnoměrné vytížení zdrojů. Je tedy důležité co nejpřesněji odhadovat přesnou délku jednotlivých úkolu, rovnoměrně přiřazovat pracovníky, kteří jsou schopni vykonávat danou činnost, ke konkrétním úkolů a maximálně využívat interních zdrojů, neboť jsou většinou spolehlivější (máme s nimi předchozí zkušenosti) a levnější.

# Kapitola 3

# Aplikace pro plánování a řízení projektu

V této kapitole bude stručně představeno několik aplikací, které slouží k plánování a řízení projektu. Tyto aplikace obecně šetří spoustu času, zejména při přeplánování nebo při odhadovaní celkového času a potřebných prostředků pro projekt atd.

## 3.1 Microsoft Project

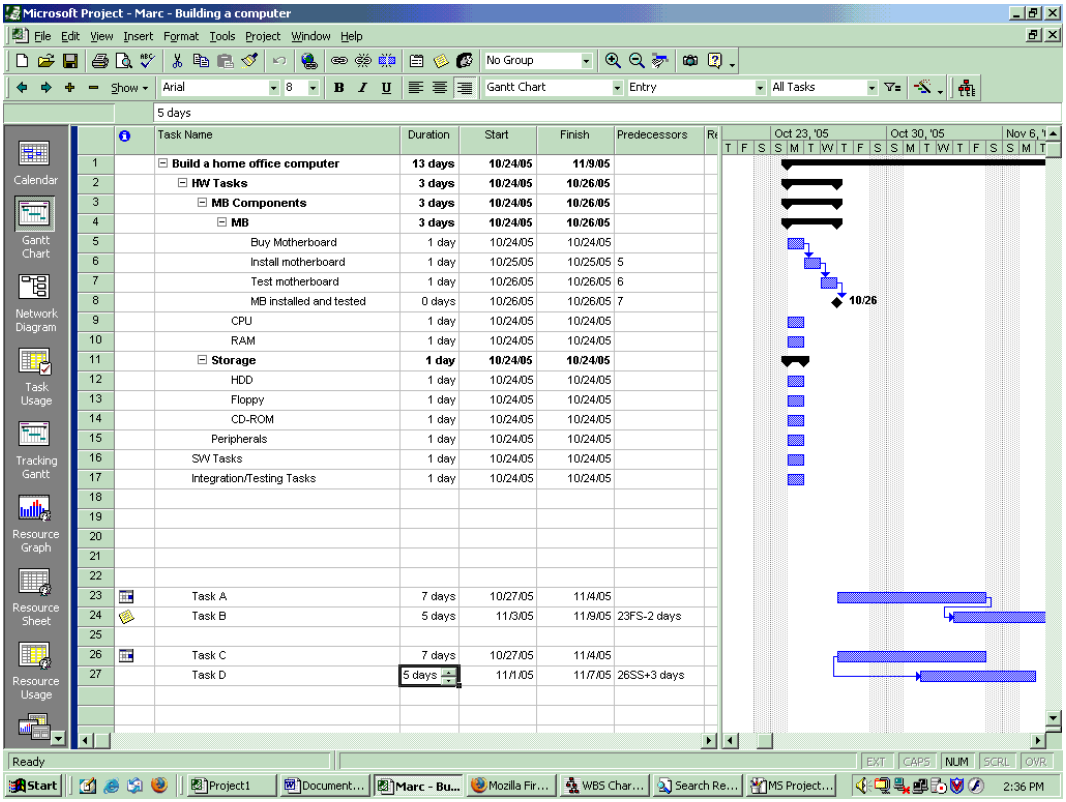
Jedná se o jeden z nejrozšířenějších nástrojů pro plánování na trhu. Pomocí MS Project lze přidávat úkoly, vytvářet mezi nimi závislosti, a vše je pak možno vidět v Ganttově diagramu (3.1). Úkoly pak mohou být přiřazeny zdrojům. Celý průběh projektu může být zaznamenán skrze MS Project, což umožní sledovat celkový pokrok projektu. Toto je dobré zejména ke včasnému přeplánování, pokud nastane neočekávaná událost (obvykle zpoždění, či nemoc pracovníka). V knize [6] lze najít popis funkcí i s příklady.

Tento software je považován za jakýsi standard, mezi jehož hlavní výhody patří zejména rozšíření na trhu a z toho vyplývá i lepší obeznámenost manažerů s tímto produktem. Cena tohoto produktu se v současné době pohybuje lehce přes 30 000 Kč. Jednou z možností, jak si produkt vyzkoušet, je stáhnout trial verzi ze stránek společnosti Microsoft, další může být využít Microsoft Office online (pomocí webového prohlížeče si lze vyzkoušet produkty řady Microsoft Office).

## 3.2 OpenProj

Tento produkt patří do kategorie open source, tj. zdarma máme přístupný nejen samotný produkt, ale dokonce i zdrojový kód. Ovládání je vcelku podobné jako u MS Project. Z obrázku 3.2 můžeme vidět i nápadnou podobnost s aplikací MS Project. Možná bychom mohli s jistou nadsázkou říci, že vztah mezi MS Project a OpenProj je obdobný jako MS Office k OpenOffice.org.

Co se rozsahu funkcí týče, jsou zde prakticky všechny hlavní funkce jako u aplikace MS Project. Další výhodou může být podpora formátů MS Project, ale i platformová nezávislost (program lze používat např. pod systémy Microsoft Windows, Linux, Mac OS X). Podrobnější informace o tomto produktu lze najít na domovské stránce[8]. Zde je možno i program zdarma stáhnout.



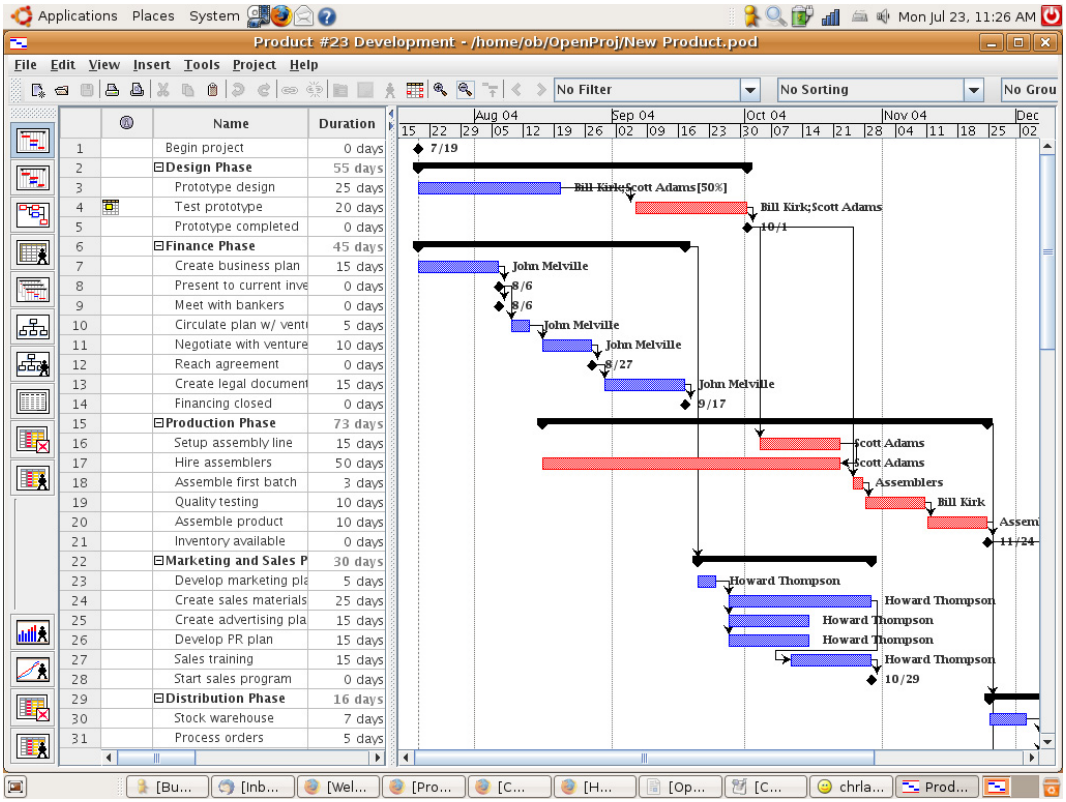

Obrázek 3.1: Snímek obrazovky z aplikace Microsoft Project

## 3.3 GanttProject

GanttProject je další volně dostupnou aplikací pro plánování projektů. Obsahuje typické funkce jako je možnost tvorby hierarchie a závislosti úkolů, Ganttův diagram atd. Je schopen pracovat s formátem souboru aplikace MS Project a generovat výstupy ve formátech *.pdf a *.html. Vzhledem k tomu, že GanttProject je java aplikace, je i platformově nezávislý, tudíž ho můžeme spustit jak na Microsoft Windows či Linuxu, tak i Mac OS X. Více o tomto programu můžete najít [4], kde je i program k dispozici ke stažení.

## 3.4 KPlato

Dalším volně dostupným programem pro plánování je KPlato. Jedná se o součást známého kancelářského balíku KOffice. Poskytuje základní funkce, jako např. Ganttův diagram, přiřazení zdrojů, WBS(work break down), znázornění kritické cesty, souhrnné úkoly atd. V současné době je tento produkt pouze pro Linux a Mac OS X. Více o tomto programu můžete zjistit na domovkých stránkach[7], kde si můžete tento program i zdarma stáhnout.




Obrázek 3.2: Snímek obrazovky z aplikace OpenProj

## 3.5 Porovnání aplikací pro plánování

MS Project je na tom co se týká požadované funkcionality velmi dobře, další výhodou je vysoké zastoupení na trhu, a to zejména díky tomu, že je vytvořen právě společností Microsoft, jež má obecně velký vliv na světový trh zejména v oblasti software. MS Project je v současné době považován za jakýsi standardní software pro projektové řízení.

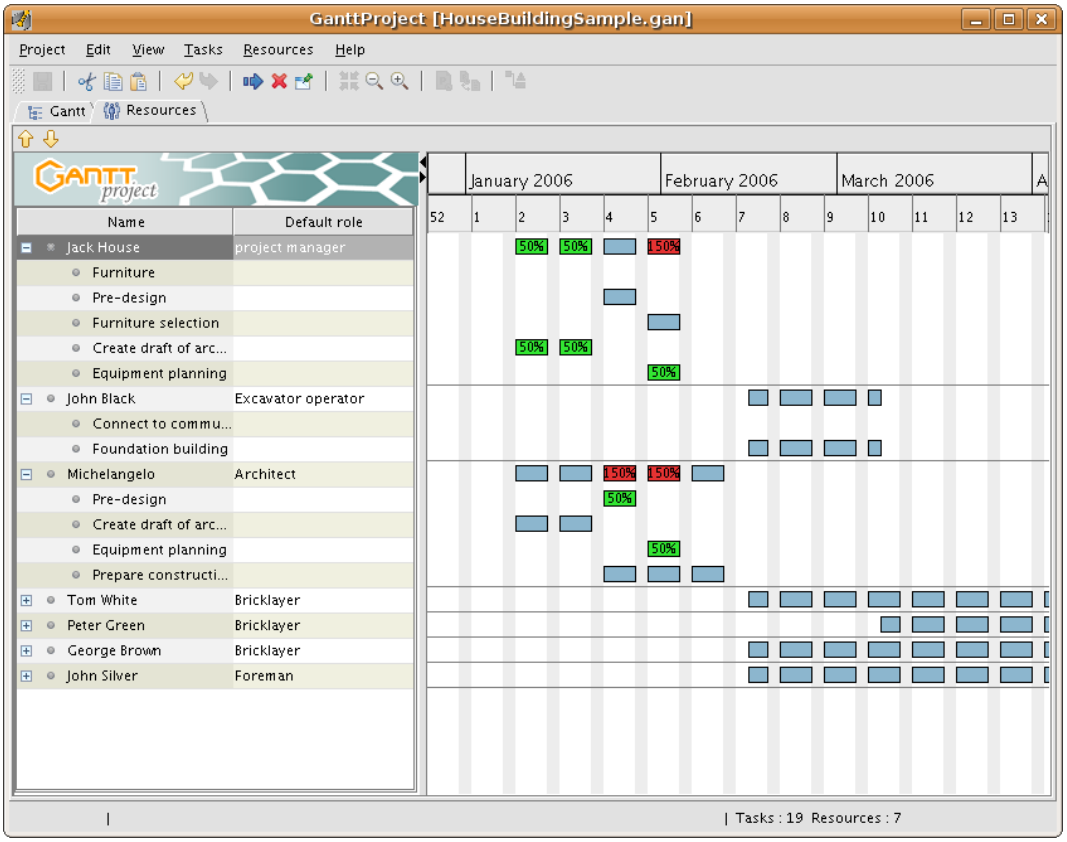
Toto si můžeme ukázat například i na tom, že ostatní zmíněné produkty (kromě KPlato) se snaží o to, aby se souborovým formátem MS Projectu uměly také pracovat. Další výhodou může snadno dostupná knižní dokumentace a dosažitelnost kurzů pro tuto aplikaci.

Také způsob ovládání ve srovnávaných aplikacích je relativně podobný MS Projectu, zejména u OpenProj. Další výhodou MS Projectu je dobrá integrace s ostatními produkty Microsoftu, z čehož ale na druhou stranu vyplývá jakási závislost na dalších nákupech (Outlook, MS Exchange atd.). Z čehož ale vyplývají ještě výrazně vyšší nepřímé náklady, pokud nemáme potřebnou infrastrukturu již vybudovanou.

GanntProject oproti například OpenProj postrádá některé funkce, nicméně ty nejzákladnější zde bez problému najdeme.

Co se hardwarové náročnosti týče, tak OpenProj má vyšší nároky na paměť než například Microsoft Project. Jak GanttProject, tak OpenProj jsou spustitelné na Linuxu, MS Windows i na Mac OS X.

KPlato má o něco více funkcí než například GanntProject. Jeho velkou nevýhodou je, že není momentálně dostupný pod MS Windows, na který je velká část manažerů zvyklá.




Obrázek 3.3: Snímek obrazovky z aplikace GanntProject

## 3.6 Možné nedostatky

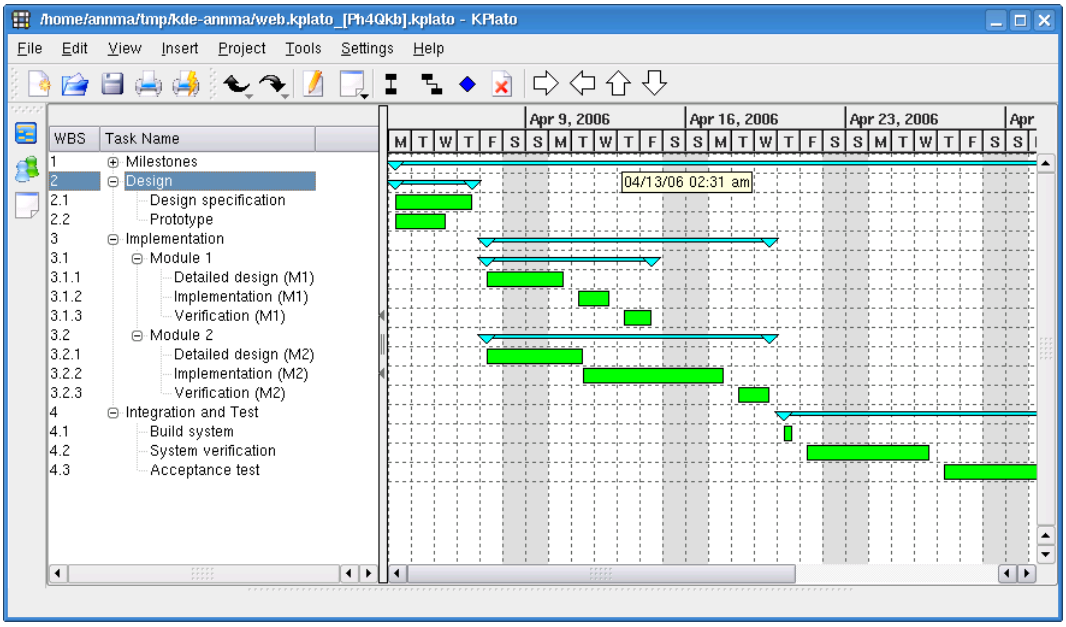
Při analýze jednotlivých funkcí výše zmíněných programů jsem našel několik z mého pohledu nedostatků.

### 3.6.1 Závislost na jedné osobě

Dle mého názoru trpí všechny výše zmíněné programy zejména příliš velkou centralizací, tj. pouze jedna osoba musí vytvořit celý plán projektu. Tato osoba musí znát mít výborné znalosti o celém chodu organizace, protože musí co nejpřesněji odhadovat dobu a náklady trvání jednotlivých úkolu, což je velmi obtížné, ne-li prakticky nemožné. Toto je však bývá částečně řešeno nadstavbových software. V případě MS Projektu se jedná je MS Project Server (klientem je pak MS Project). V případě OpenProj existuje komerční řešení Project-On-Demand (klientem je webový prohlížeč). Tyto produkty umožní sdílení dat, nicméně nedefinují role uživatelů v systému.

### 3.6.2 Komunikace

Dalším problémem by mohla být komunikace, MS Projekt toto řeší zasíláním zpráv pomocí Outlooku, tj. vzniká zde závislost na balíku Microsoft Office a případně Microsoft Exchange Serveru.




Obrázek 3.4: Snímek obrazovky z aplikace KPlato

Dle mého osobního názoru je zasílání emailu použitelné pouze při nižší hustotě zadávání úkolu. Při vyšší hustotě zadávání požadavků může být vhodnější vytvořit uživateli možnost po přihlášení do systému si prohlédnout své úkoly s možností filtrace, u emailu jsme v tomto značně omezeni. Tento systém je výhodnější zejména kvůli rychlejší distribuci dat nebo například změny priority již zadaného úkolu (například při změně kritické cesty, nemoci pracovníka, změně termínu). Tento přístup však vyžaduje častý přístup uživatele do systému, což může být pro uživatele s nižší hustotou zadávaných úkolu nepohodlné. Proto by mělo být možné zvolit si z obou variant v závislosti na typu pracovníka.

Další výhodou řešení komunikace pomocí systému je možnost tvorby různých filtrů, dynamické přesouvání pořadí úkolu podle priority termínů. Systém by tedy například mohl automaticky zvýšit prioritu provedení úkolu v případě, že by tento úkol ocitl na kritické cestě. Hlavní výhodou oproti emailu je zejména přehlednost a možnost systému ovlivňovat již zaslané zprávy (např. možnost změny termínu, priority apod.). Pokud by pracovníkem byl externista, bylo by zde vhodné implementovat i funkci, která umožní tomuto externistovi nepřijmout úkol (nebo spíše nepotvrdit přijetí úkolu). Manažer by v tomto případě mohl být ihned informován a najít si potřebnou náhradu.

### 3.6.3 Nedostatečná podpora sběru dat

Dalším problémem by mohla být nedokonalá znalost prostředí manažera, spočívající například v odhadování doby trvání jednotlivých úkolu. Řešením by mohlo být, aby tuto bázi dat (např. délku trvání úkolu), mohli zadávat i lidé, kteří se lépe orientují v problematice z technického hlediska.

Manažer by si pak mohl pak mohl skládat celý projekt na již existujících základech, což by umožnilo výrazněji zpřesnit odhad času a doby, kterou bude projekt vyžadovat, čímž by se i snížilo riziko neúspěšného dokončení projektu při zachování plné možnosti řízení a korekce manažera.

Manažer musí mít možnost i bázi úkolů upravovat sám, neboť by zde mohlo být riziko, že techničtí vedoucí budou zadávat nepřesné informace, a to zejména zadáním vyššího času a nákladů, než je na projekt nezbytně nutné. Vzhledem k tomu, že manažer uvidí i kdo informaci do systému zadal, bude si moci případně vyžádat konzultaci o důležitých a nákladných částech projektu.

I přesto, že korekce manažera při upravování délek a nákladů na projekt je nezbytně nutná, je tento přístup velmi výhodný zejména při počátečních odhadech. V případě, že by i techničtí vedoucí zadávali do systému vždy pravdivé údaje, k čemuž je bude motivovat možnost kontroly a vyžádaní konzultace managerem, dojde k výraznému zjednodušení řízení částí projektu, se kterými mají někteří lidé ze společnosti již zkušenosti.

Tato báze úkolu by také napomáhala i v začlenění nových řídících pracovníků do společnosti. Neboť asi největším problémem v týmových projektech je decentralizace znalostí jednotlivých účastníků projektu, proto je pro efektivní chod společnosti výhodné, aby v rámci společnosti byly požadované informace co nejlépe dostupné.

### 3.6.4 Lepší podpora vyhledání informací z již zadaných dat

Asi nejlepším zdrojem informací bývá ve většině případů osobní konzultace s pracovníkem, který má zkušenosti s podobným problémem, který právě řešíme.

Tento zdroj informací je velmi drahý, protože vyžaduje zaplatit čas člověka, který tyto informace poskytuje. Nicméně konzultace s touto osobou nám může ušetřit mnoho problémů a snížit, tak konečný čas a náklady na provedení námi zadaného úkolu.

Bylo by tedy dobré, kdyby systém mohl dohledat pracovníky, kteří již byli přiřazeni k úkolu, který nás zajímá. Tento pracovník by si mohl poslat žádost na konzultaci. Vzhledem k tomu, že v systému pro řízení budou již uvedeny tyto informace, neznamená přidání této funkcionality prakticky žádné další požadavky na zadávaní informací.

# Kapitola 4

# Požadavky na systém

Na základě analýzy již dostupných prostředků se budeme snažit navrhnout systém, který si vybere základní část funkcionality z předchozích aplikací s přidáním podpory funkcí, které budou sloužit zejména ke zvýšení rychlosti plánování, vylepšení komunikace a lepšímu sběru dat z předchozích projektu.

## 4.1 Základní funkcionalita

Musí být umožněna také správa lidských či ostatních zdrojů, ale i míst. Systém bude umožňovat vytvářet a rušit projekty. Do těchto projektů bude možno přidávat, ubírat a editovat jednotlivé úkoly. Tyto úkoly budou moci být uspořádány do hierarchie. K úkolům se budou moci přiřadit zdroje, a to jak lidské, tak materiální.

## 4.2 Multiuživatelský systém

Systém bude v maximální možné míře podporovat týmovou spolupráci, základním atributem je tedy zajištění správu účtů. Účty bude možno přiřazovat do skupin s odpovídajícím oprávněním. Dle oprávnění systém nabídne seznam činností, které bude budou moci autentizovaní uživatele používat.

## 4.3 Komunikace

Systém bude umožňovat uživateli při významné události (například přiřazení nového úkolu) informovat uživatele, a to jak pomocí systému, tak emailem. Systém bude podporovat zasílání zpráv mezi uživateli. Dále systém umožní sdílení poznámek mezi uživateli.

## 4.4 Sběr dat

Systém umožní některým vybraným osobám vytvářet jakousi databázi znalostí, ze které pak manažer bude moci vytvářet projekty.

## 4.5 Jednoduchá obsluha

Systém bude mít největší přínos pro organizaci, která jej bude využívat zejména v případě, pokud do něj bude vstupovat maximální množství lidí. Systém tedy musí být maximálně jednoduchý a intuitivní, aby bylo možné je bezproblémově nasadit.

## 4.6 Snadná rozšiřitelnost

Pro fungovaní systému jako takového je velmi důležitá snadná rozšiřitelnost a dostupnost, proto budou moci klienti přistupovat do systému pomocí webové aplikace. Tento přístup znamená téměř nulové náklady pro přidání klienta.

# Kapitola 5

# Návrh aplikace

V této kapitole bude proveden návrh aplikace. Bude zde objasněn základní model architektury, uložení dat (ER diagram) a podrobněji specifikovaná funkcionalita (Diagram případu použití).

## 5.1 Architektura

### 5.1.1 Webová aplikace

Mezi nesporné výhody webové aplikace patří zejména dostupnost. Jedná se o architekturu klient-server. Na serveru běží webový (IIS) a databázový (MSSQL) server. Jako klient je použit klasický webový prohlížeč, který bývá standardní současní operačních systémů. Často jej nalezneme i v PDA, či mobilním telefonu (miniOpera). Dostupnost tedy závisí v podstatě pouze na dostupnosti serveru, neboť klientské programy jsou všude snadno dostupné.

Veškeré aktualizace postačí provádět pouze na serveru. Odpadají tedy problémy s neaktuálností klientů. Všechny tyto výhodou vedou k výrazně nižším nákladům na údržbu a aktualizace. Za tuto flexibilitu však můžeme platit výkonností celého systému. Na tuto aplikaci nejsou kladeny příliš vysoké výkonností nároky. Vyšší nároky na databázový server vznikají pouze jednorázově při provedení určitých operací. Významnější je spíše dostupnost. Z tohoto důvodu byla volena forma webové aplikace.

### 5.1.2 MVC – Model View Controller

Vrstvy sdružují ty části aplikace, které mají podobnou funkcionalitu. Rozdělení systému do vrstev může být výhodné zejména v případě rozsáhlejších aplikací. Mezi hlavní výhodu patří znovupoužitelnost. Díky rozdělení do vrstev nám stačí znát pouze metody okolních vrstev. Tyto metody pak můžeme volat s různými parametry.

Můžeme si uvést malý příklad, na kterém bude ukázána výhoda vrstevnatého modelu. Pokud bychom vytvořili SQL řetězec a poté odeslali databázi a výsledek poté ihned zobrazili, museli bychom tento SQL řetězec vytvářet celý znovu při každém opětovném použití i v případě, že bychom chtěli zpracovat ta samá data (například je můžeme různě zobrazovat nebo je využít jako zdroj dat k různým výpočtům). Tento způsob programovaní není vhodný, zejména pokud vezmeme v potaz budoucí možné změny v databázi, museli bychom v případě změny upravit každý SQL řetězec ve všech jeho výskytech, což je nejenom časové

náročné, ale je to i potencionální zdroj chyb. Další výhodou přístupu založeném na vrstvách je celkové přehlednější kód a tím i menší náchylnost na tvorbu chyb.

Dále je umožněna lepší spolupráce více lidí. Programátorovi pro řídící vrstvu stačí znát funkce a datové objekty, které může použít a již nemusí znát fyzickou strukturu databáze. Designérovi grafického uživatelského rozhraní stačí pouze říci jaké obrazovky s jakými grafickými prvky má vytvořit. Každý člen týmu si tedy může tvořit pouze ve své vrstvě a navzájem si pouze zveřejňovat pouze názvy objektů a metod. Tento přístup tedy například umožňuje, aby designér grafického rozhraní nemusel znát SQL jazyk. A naopak databázový specialista nemusí znát CSS, či HTML nebo ovládat různé grafické programy.

Celý systém je implementován ve třech základních vrstvách:

- Databázová vrstva
- Řídící vrstva
- Prezentační vrstva

Úkolem databázové vrstvy je zpřístupnit data z databáze řídící vrstvě, která pak může přistupovat k datům pomocí objektového modelu. Databázové operace (vkládání, mazání, editace) jsou pak metody těchto objektů.

Řídící vrstva zabezpečuje celkovou logiku aplikace, stará se o obsluhu událostí, které vznikají při použití systému. Ovlivňuje obsah nikoliv však formu zobrazení na základě logiky aplikace. Stará se o udržování kontextu uživatele. Spolupracuje s databázovou (vydává pokyny k získání a úpravě dat) i prezentační vrstvou, tj. řídí jaké informace mají být předány prezentační vrstvě k zobrazení.

Tvorba výsledného grafického uživatelského rozhraní má na starosti prezentační vrstva. V této vrstvě je deklarativně popsán vzhled jednotlivých komponent a rozmístění na stránce. K těmto komponentám jsou pomocí řídící vrstvy přiřazena data. Na základě událostí od uživatele se stránka různě dynamicky mění.

## 5.2 Uživatelský model systému

Systém je navržen tak, aby jej mohlo používat několik základních skupin lidí. Tyto skupiny se přirozeně vyskytují v každém větším projektu. Každý projekt má obvykle nějaké vedoucí, kteří řídí projekt jako takový, v dalším textu označováni jako projektoví manažeři.

Dále se ve skupině lidí realizující projekt obvykle vyskytují osoby, které rozumí jistých částem projektu spíše z oborově specifické hlediska. Tito lidé mají obvykle přesnou představu o tom, co bude pro daný úkol potřeba zajistit, jak dlouho asi bude trvat a kolik to tak asi bude stát.

K tomu, abychom mohli vykonat úkol, potřebujeme obvykle lidi s určitými schopnosti (např. lektora), případně speciální prostory (učebna s počítači), případně další zdroj (např. 3D scanner). Náš specialista tedy zná požadavky na lidi, zdroje a místa. Nicméně nezná detailně všechny zaměstnance a proto vstupuje do hry i pracovník, který bude znát informaci, který konkrétní pracovník dokáže splnit požadavky, například který pracovník je lektor. Tuto informaci bude znát asi člověk, který ověřil kvalifikaci pracovníka firmy, v následujícím textu označován jako personalista. a konečně zde máme samotné pracovníky.

Vznikly nám tedy čtyři skupiny zaměstnanců:

- Projektoví manažeři

- Specialisté na úkoly
- Personalisté ověřují kvalifikaci zaměstnanců
- Pracovníci.

### 5.2.1 Jak je tato skutečnost modelována v systému?

Skutečnost, že některý pracovník splňuje konkrétní požadavky, je dána jeho příslušností do určité skupiny. Personalista tedy bude mít za úkol spravovat členství pracovníků v těchto skupinách. Obdobně je tomu i u místa a ostatních zdrojů.

Skutečnost, že k realizaci úkolu jsou potřeba lidé, zdroje, místa modeluje Specialista na úkoly tím, že určí, které skupiny jsou potřeba k provedení tohoto úkolu.

Tím, že Specialisté na úkoly a Personalisté zanáší své znalosti do systému, vzniká jakási báze dat, která pak může být využita manažerem.

Manažer projektu pak sestavuje konkrétní plán projektu, kde již využívá toho, že specialista mu určil typem úkolu skupiny lidí, míst a ostatních zdrojů, které potřebuje. Stačí mu tedy jen vybrat zvolené pracovníka z této skupiny, tak aby nedocházelo k blokaci. Systém před potvrzením ukáže blokace v předvoleném termínu příslušného pracovníka, místa, či zdroje. Pracovníci jsou modelovány jako objekty, kterým jsou pomocí systému doručeny pokyny.

### 5.2.2 Nevýhody jednouživatelskému přístupu

Pokud bychom použili například standalone aplikaci MS Project, musela jedna osoba (projektový manažer) mít detailní informace o celém chodu organizace. Znát schopnosti všech pracovníku v organizaci, znát detailně všechny činnosti, které organizace dělá. Nároky na projektového manažera jsou pak obrovské ne-li nereálné. Tento člověk je pak klíčovou osobou a dalo by se říci, že i velmi zranitelné místo celé organizace, protože jedině on má globální znalosti, které jsou potřeba pro řízení projektů v organizaci.

Vzhledem k tomu, že pouze jediná osoba tvoří plán projektu, trvá tento plán se sestavit poměrně dlouho dobu. Důsledkem toho může pak organizace přijít o netrpělivého klienta. Neboť získání potřebných informací pro plánování, pokud je projektový manažer nezná, je velmi časově náročné (musí zjistit, kdo tyto informace ví a následně danou osobu kontaktovat).

### 5.2.3 Výhody výše uvedeného modelu

Personalisté a Specialisté velmi usnadňují práci projektovému manažerovi, neboť díky nim pak může rychleji a přesněji tvořit plány konkrétních projektů. Personalisté a Specialisté průběžně spravují tuto bázi dat, tudíž je stále aktuální. Projektový manažer může v různých projektech tuto bázi znalostí opakovaně používat. Protože báze dat se vlastně tvoří po celou dobu chodu organizace a ještě navíc paralelně, je okamžitě připravena pro použití v rámci tvorby unikátního projektu, což celý proces plánování významně urychlí.

## 5.3 Komunikace

Předávání a sdílení informací je velmi důležité pro úspěšný chod celého projektu. Rychlá distribuce informací může mít velký vliv zejména při přeplánování projektu v důsledku

nečekaných událostí, neboť pracovník může začít pracovat teprve poté, co dostane pokyn. Systém se v tomto ohledu snaží pomoci integrováním některých podpůrných funkcí.

- Automatické generování zpráv
- Uživatelské komunikace v rámci systému
- Vkládání poznámek

Systém je schopen automaticky generovat zprávy na základě určitých událostí (například přidání pracovníka ke konkrétnímu termínu daného úkolu). Tyto zprávy pak rozešle uloží do systému a v případě nastavení emailem. Touto funkcí se opět šetří čas a náklady, které by musely být vynaloženy na komunikaci.

Pro předávání sdílených informací byla vytvořena možnost vkládání libovolného množství poznámek k různým typům entit (typy úkolu, konkrétní realizace úkolu, uživatelé atd.).

## 5.4 Odhady systémy

Systém se snaží pomáhat tím, že vyvozuje další užitečné informace z již zadaných dat. Tyto informace mohou pomoci při plánování (odhad nákladů a času), případně při řešení nějakého problému (vyhledání vhodné osoby pro konzultaci). Nezanedbatelnou výhodou je, že získaní této informace uživatele nestojí žádné úsilí z hlediska zadávání dat.

Systém je schopen na základě opakovaní realizovaní určitého typu úkolů odhadnout z již zadaných dat, jak dlouho a kolik finančních prostředků je potřeba ve skutečnosti na tento úkol. Tuto informací získá z konkrétních realizací tohoto typů úkolu. Dále se pokouší odhadnout vhodné nejzkušenější pracovníky, na základě množství jejich předešlých realizací konkrétního typu úkolu.

## 5.5 ER Diagram

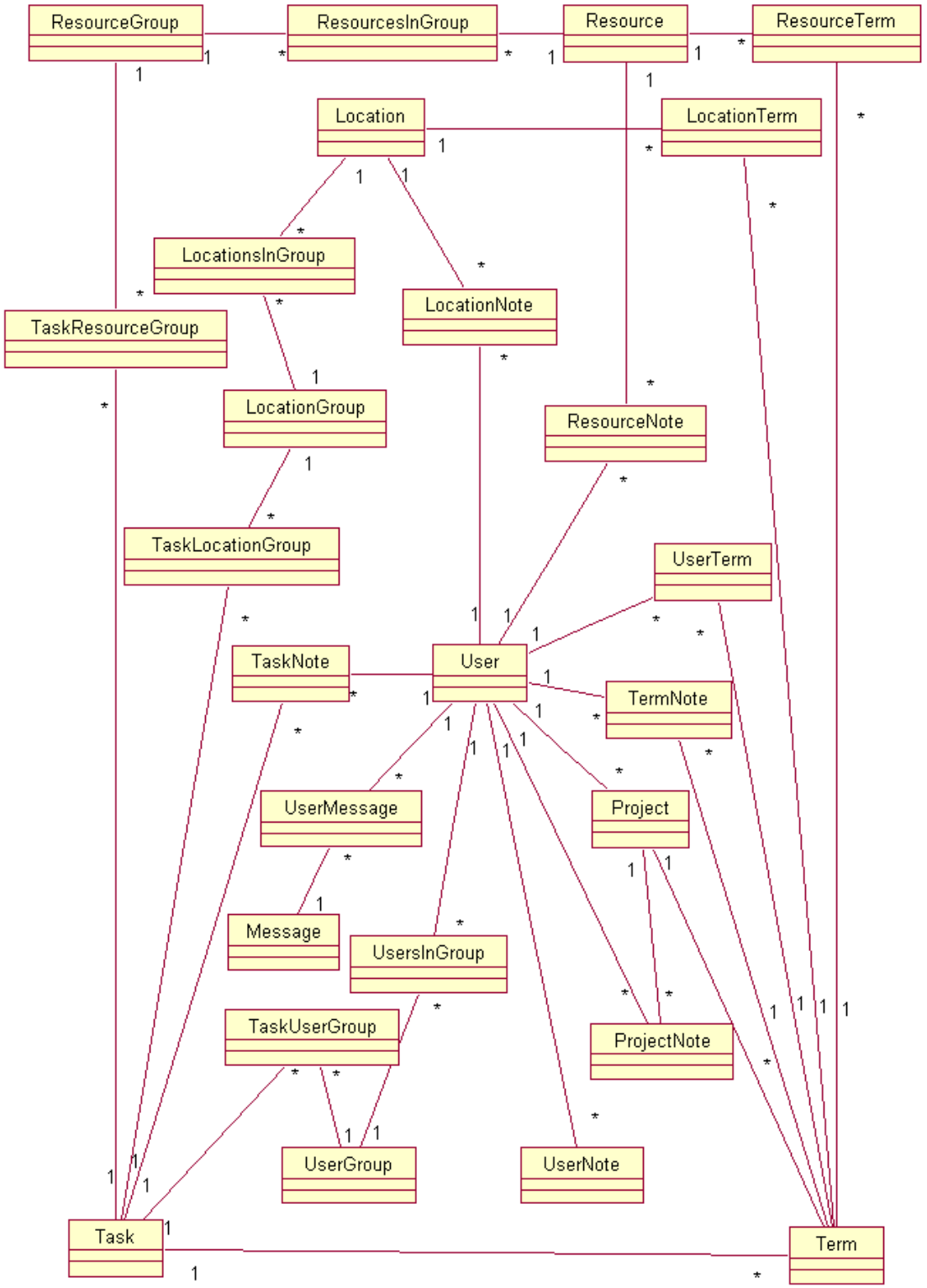
ER Diagram neboli Entity Relationship diagram zachycuje strukturu databáze. Jednotlivé entity jsou zachyceny pomocí tabulek, mezi entitami jsou vazby, které jsou zaznamenány pomocí čar, které mají na svých koncích určenou kardinalitu. Pro realizaci databázové struktury se ER-diagram transformuje na tabulky. Každý vztah má svou kardinalitu a je typu 1:1, 1:N, M:N. Pokud bychom měli mezi entitami Adresa a Organizace vztah 1:1, znamená to, že k jedné organizaci patří právě jedna adresa. Pokud je vazba mezi entitami Task (Word) a Term (Word 20.9.2008-21.9.2008) 1:N, bude to konkrétně znamenat, že se může odkazovat N položek z tabulky Term na 1 položku v tabulce Task. Toto N se zvětší na N+1 v případě, že bude vypsán nový termín pro úkol stejného typu. Pokud je například vazba typu M:N mezi entitami UserGroup a User, může to znamenat, že uživatel může patřit do více skupin a zároveň v jedné skupině může být více uživatelů.

Pro přehlednost zobrazení 5.5 nejsou atributy entit viditelné.

### 5.5.1 Popis jednotlivých entit ER Diagram

**Project** charakterizuje projekt. Tento projekt sdružuje několik konkrétních úkolů.

**Task** znázorňuje typ úkolu. Jedná se druh činnosti. Vazba mezi UserGroup a Task vyjadřuje nutnost použití této skupiny pro provedení daného úkol. Vazba mezi LocationGroup a Task značí nutnost konání se úkolu na místě, které je v této skupině.




Obrázek 5.1: Entity Relationship Diagram aplikace bez viditelných atributů

Vazba mezi ResourceGroup a Task značí potřebu speciální skupiny zdrojů pro prove-

dení úkolu.

**Term** vyjadřuje úkol zadaný v konkrétním čase. Druh činnosti je vyvozen z vazby na typ úkolu (Task). Na tento termín úkolu mohou být přiřazeni uživatele, různé zdroje a může se konat na různých místech. Hierarchické struktury je zde dosaženo pomocí položky parrentTerm, ve které je uložen identifikátor nadřazeného úkolu. V případě nejvyšší úrovně (neexistuje nadřazený úkol) je tato hodnota null.

**User** vyjadřuje uživatele, který vstupuje do systému, tento uživatel je zároveň i pracovník, který je schopen plnit určité typy úkolů a taky může být přiřazován k jednotlivých úkolům (Term).

**Resource** představuje specifické zdroje, které mohou být přiřazeny k úkolům (Term) nebo ke konkrétním termínům daného úkolu (Term).

**Location** vyjadřuje místo, kde mohou probíhat jednotlivé úkoly (Term) naplánované v konkrétním daném čase.

**UserGroup** umožňuje sdružovat uživatele (User) do skupin. Tyto skupiny pak mohou být přiřazovány k typu úkolu (Task).

**ResourceGroup** umožňuje sdružovat zdroje (Resource) do skupin. Tyto skupiny pak mohou být přiřazovány k typu úkolu (Task).

**LocationGroup** umožňuje sdružovat místa (Location) do skupin. Tyto skupiny pak mohou být přiřazovány k typu úkolu (Task).

**TaskLocationGroup** vyjadřuje nutnost použití nějakého člena ze skupiny míst (LocationGroup) pro provedení daného úkolu.

**TaskResourceGroup** vyjadřuje nutnost použití nějakého člena ze skupiny zdrojů (LocationGroup) pro provedení daného úkolu.

**TaskUserGroup** vyjadřuje nutnost použití nějakého člena ze skupiny uživatelů (LocationGroup) pro provedení daného úkolu.

**UserTerm** vyjadřuje přiřazení uživatele ke konkrétnímu úkolu.

**ResourceTerm** vyjadřuje přiřazení zdroje ke konkrétnímu úkolu.

**LocactionTerm** vyjadřuje přiřazení místa ke konkrétnímu úkolu.

**UsersInGroup** vyjadřuje členství uživatele ve skupině.

**LocationsInGroup** vyjadřuje příslušnost místa do skupiny.

**ResourcesInGroup** vyjadřuje členství zdroje ve skupině.

**ProjectNote** slouží k uchování poznámek k projektu (Project).

**LocationNote** slouží k uchování poznámek k místu (Location).

**ResourceNote** slouží k uchování poznámek ke zdroji (Resource).

**UserNote** slouží k uchování poznámek k uživateli (User).

**TaskNote** slouží k uchování poznámek k typu úkolu (Task).

**TermNote** slouží k uchování poznámek ke konkrétnímu úkolu (Term).

**Message** slouží pro uchování obsahu zpráv.

**UserMessage** vyjadřuje vztah mezi zprávou a příjemcem této zprávy.

## 5.6 Diagram případů užití

Tento diagram zachycuje funkcionalitu skupin osob, které vstupují do systému. Na základě příslušnosti ke skupině má uživatel v tomto systému jistá oprávnění.

### 5.6.1 Slovní popis Diagramu případu užití

Systém bude využívat několik rolí uživatelů. Každá role má v systému svůj sémantický význam a systém, každému nabídne jiné možnosti dle jeho role.

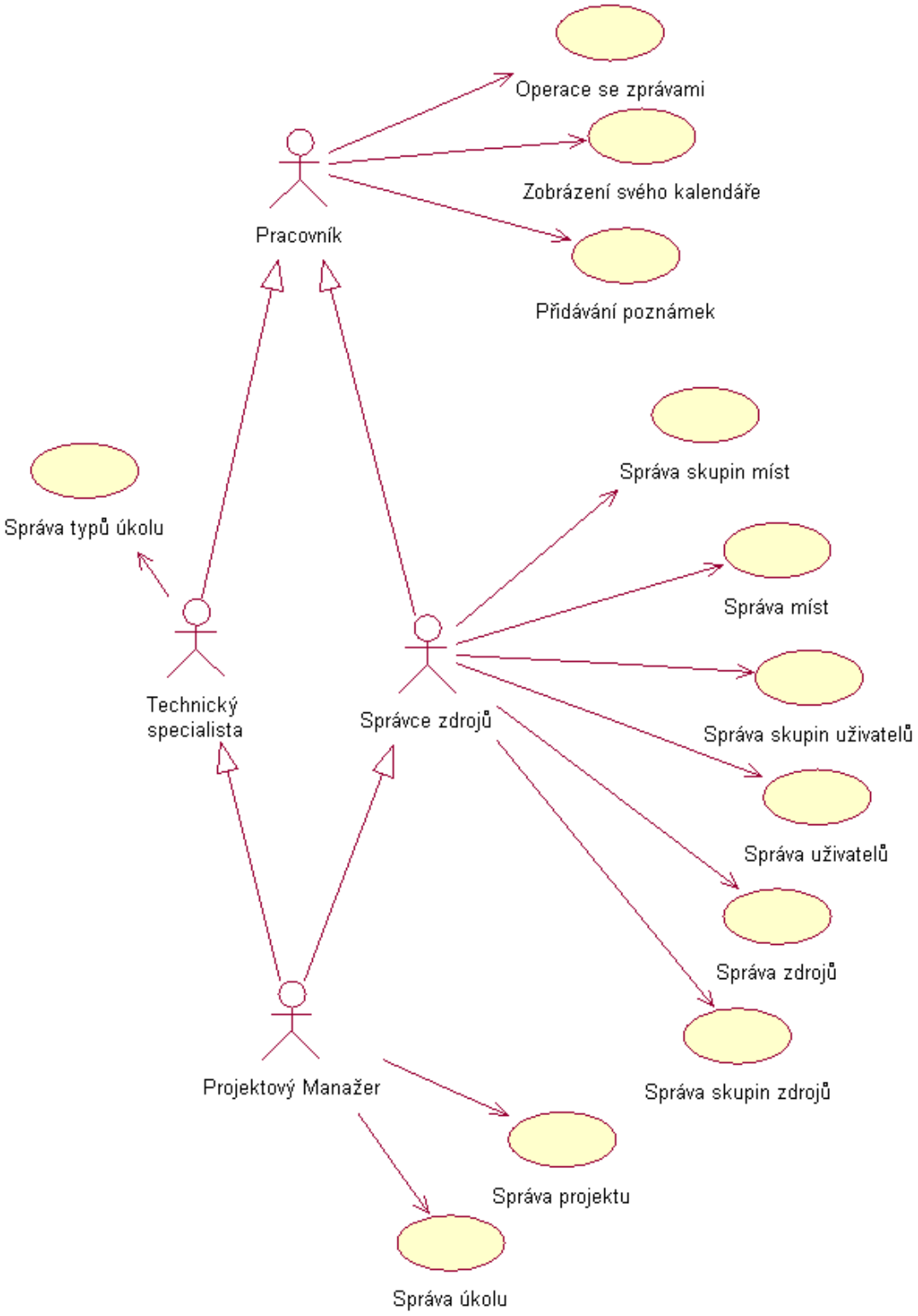
Systém bude rozlišovat následující typy rolí:

- Projektový manažer
- Technický specialista
- Správce zdrojů
- Pracovník

Hlavní úlohou Projektové manažera je sestavit projekt jako takový. Bude moci využívat informací zadané Technickými specialisty a Správci zdrojů. Projektový manažer jako zastřešující osoba však bude mít pravomoci i těchto dvou rolí, aby mohl provádět případné korekce. Pracovník je pomocí systému informován o tom, co má dělat.

Projektový manažer tedy bude moci v systému provádět následující činnosti:

- Správa uživatelů – Přidání, mazání, editace účtů, které jsou nutné pro vstup do systému
- Správa skupin uživatelů – Přidání, mazání, editace skupin, určování členství v těchto skupinách. Členství ve skupině znamená určitou způsobilost. Člen skupiny má tedy schopnost, která umožňuje vykonávat některé typy úloh.
- Správa míst – Přidání, mazání, editace míst, tato místa pak mohou být přiřazeny k úkolu.
- Správa skupin míst – Přidání, mazání, editace skupin, určování členství v těchto skupinách, členství ve skupině znamená určitou způsobilost. Člen skupiny má tedy speciální vlastnosti, které umožňují vykonávat určité typy úkolů právě na tomto typu míst.
- Správa zdrojů – Přidání, mazání, editace ostatních zdrojů, tyto zdroje zahrnují všechny zdroje kromě zdrojů lidských (například automobil), mohou pak být přiřazeny k úkolu.




Obrázek 5.2: Diagram případu užití aplikace

- Správa skupin zdrojů – Přidání, mazání, editace skupin, určování členství v těchto skupinách, členství ve skupině znamená určitou způsobilost. Člen skupiny má tedy speciální vlastnosti, které umožňuje vykonávat určité typy úloh právě pomocí tohoto zdroje.
- Správa typů úkolů – Typy úkolů slouží ke stanovení parametrů činnosti. K těmto typům úkolů jsou přiřazovány skupiny lidí, míst, zdrojů, které jsou potřebné pro úkol. Typům úkolů pak následně může být přiřazeno konkrétní datum a konkrétní lidé, místa a zdroje.
- Správa projektu – Přidání, mazání editace projektů, volba aktuálního projektu pro práci.
- Správa úkolů – Přidání, mazání editace úkolu. V úkolu se určuje začátek a konce trvání. Úkol je konkrétní realizace typu úkolu. Na základě volby typu úkolu se zobrazí skupiny lidí, míst, zdrojů, které byly přiřazeny k typy úkolu. Výběrem konkrétní skupiny zobrazíme členy této skupiny. Výběrem člena skupiny, zobrazíme jeho blokací v naplánovaném termínu. Následně můžeme provést přiřazení. V úkolech lze budovat hierarchickou strukturu úkolu.
- Operace se zprávami – Posílání, čtení přijatých zpráv zaslaných v rámci systému.
- Přidávání poznámek – Tyto poznámky lze přidávat k uživatelům, místům, zdrojům, úkolům, typům úkolů.
- Zobrazení svého kalendáře – Pomocí tohoto kalendáře se zobrazí termíny na základě přiřazení k úkolům ve všech projektech.

Technický specialista tedy bude moci v systému provádět následující činnosti:

- Správa typů úkolů – Typy úkolů slouží ke stanovení parametrů činnosti. K těmto typům úkolů jsou přiřazovány skupiny lidí, míst, zdrojů, které jsou potřebné pro úkol. Typům úkolů pak následně může být přiřazeno konkrétní datum a konkrétní lidé, místa a zdroje.
- Operace se zprávami – Posílání, čtení přijatých zpráv zaslané v rámci systému.
- Přidávání poznámek – Tyto poznámky lze přidávat k úkolům.
- Zobrazení svého kalendáře – Pomocí tohoto kalendáře se zobrazí termíny na základě přiřazení k úkolům ve všech projektech.

Správce zdrojů tedy bude moci v systému provádět následující činnosti:

- Správa uživatelů – Přidání, mazání, editace účtů, které jsou nutné pro tvorbu systému.
- Správa skupin uživatelů – Přidání, mazání, editace skupin, určování členství v těchto skupinách, členství ve skupině znamená určitou způsobilost. Člen skupiny má tedy schopnost, která umožňuje vykonávat některé typy úkolů.
- Správa míst – Přidání, mazání, editace míst, tato místa pak mohou být přiřazeny k úkolu.

- Správa skupin míst – Přidání, mazání, editace skupin, určování členství v těchto skupinách, členství ve skupině znamená určitou způsobilost. Člen skupiny má tedy speciální vlastnosti, které umožňují vykonávat určité typy úkolů právě na tomto typu míst.
- Správa zdrojů – Přidání, mazání, editace ostatních zdrojů, tyto zdroje zahrnují všechny zdroje kromě zdrojů lidských (například automobil), mohou pak být přiřazeny k úkolu.
- Správa skupin zdrojů – Přidání, mazání, editace skupin, určování členství v těchto skupinách, členství ve skupině znamená určitou způsobilost. Člen skupiny má tedy speciální vlastnosti, které umožňuje vykonávat určité typy úloh právě pomocí tohoto zdroje.
- Operace se zprávami – Posílání, čtení přijatých zpráv zaslané v rámci systému.
- Přidávání poznámek – Tyto poznámky lze přidávat k uživatelům, místům a zdrojům.
- Zobrazení svého kalendáře – Pomocí tohoto kalendáře se zobrazí termíny na základě přiřazení k úkolům ve všech projektech.

Pracovník tedy bude moci v systému provádět následující činnosti:

- Operace se zprávami – Posílání, čtení přijatých zpráv zaslané v rámci systému.
- Přidávání poznámek – Tyto poznámky lze přidávat k úkolům.
- Zobrazení svého kalendáře – Pomocí tohoto kalendáře se zobrazí termíny na základě přiřazení k úkolům ve všech projektech.

# Kapitola 6

# Použité technologie

V této kapitole budou stručně představeny technologie, které byly použity při vývoji aplikace. Při výběru technologie pro realizaci aplikace jsem se rozhodoval mezi PHP, JSP s využitím frameworku a ASP.NET. Pro aplikace menšího rozsahu je dle mého názoru nejvýhodnější použít PHP, protože je možné v něm poměrně snadno vytvořit funkční databázovou aplikaci. Navíc je dostupné i velké množství tutoriálů, také je ze všech třech nejlepší podpora dostupnosti hostingu. Jako jedinou nevýhodu vidím v tom, že od začátku tento jazyk nebyl budovám jako objektově orientovaný (k tomu došlo až ve verzi 5) .Další nevýhodou by mohl být výkon, neboť PHP je čistě interpretovaný jazyk (dochází k opakované kompilaci). Výhodou využití Java technologií je zejména v obrovském množství již hotových knihoven, na druhou stranu z mých osobních zkušeností vyplývá i velká vzájemná nekompatibilita. ASP.NET nakonec zvítězilo díky nulovým problémům s kompatibilitou (všechno vytvořila firma Microsoft = nižší riziko nedokončení projektu) a objektově orientovaném přístupu. Další výhodou ASP.NET je i možnost použití hostingu, tudíž by systém mohly použít i menší firmy. Jedinou nevýhodou oproti PHP shledávám ve strmější učící křivce, neboť je potřeba podstatně více znalostí, abychom mohli ASP.NET začít používat. O technologii ASP.NET se můžete dozvědět na [15] či [16], informace o použitém programovacím jazyce C# pak na [14].

## 6.1 ASP.NET

Jedná se o webový framework vyvíjený firmou Microsoft, který se používá pro tvorbu dynamických webových stránek, aplikací či webových služeb. Jedná se o nástupce ASP (Active Server Pages). ASP.NET je postaven na Common Language Runtime, který dovoluje používat libovolný jazyk z rodiny .NET (nejčastěji je to C# a Visual Basic). Tento framework má za úkol dynamicky generovat webové stránky, které obsahují HTML, CSS a JavaScript.

### 6.1.1 Code-behind model

Hlavním účelem tohoto modelu je oddělit skriptovací tagy a výkonný kód. Soubory s tagy mají obvykle příponu aspx, soubory s code-behind mají aspx.cs (vb) v závislosti na zvoleném jazyku. Tímto způsobem dochází k oddělení prezentační vrstvy. V code-behind se však může objektově přistupovat k ASP.NET tagům, které byly vytvořeny v prezentační vrstvě. V code-behind tedy obvykle najdeme zejména obsluhy události.

### 6.1.2 User controls

V ASP.NET můžeme používat velké množství již dodaných komponent (control). Tyto controls však lze i poměrně jednoduše dodělat, popřípadě se jich vyskytuje i velké množství na Internetu.

### 6.1.3 Vypořádání s udržováním kontextu

Obecným problém webových aplikací je, že HTML protokol je bezstavový. Aplikace jsou však typicky stavové (udržují kontext). Tento problém většina jazyku řeší většinou pomocí Session, Cookies či předáváním hodnot parametru pomocí metod GET (obvykle generování řetězců) a POST (obvykle přes odeslaní ve formulářích ). Session je obecně kolekce uživatelsky definovaných proměnných udržovaných na straně serveru, které se na serveru udrží po dobu uživatelské aktivity. Cookies udržují kontext na straně klienta, tj. ukládají si stavové informace do souboru na klientské stanici.

### 6.1.4 ViewState

Tento mechanismus ASP.NET používá k předávání stavu ASP.NET komponent (controls) HTML stránce. Jedná se po skryté pole ve formuláři nazvané jako _VIEWSTATE. Hodnota tohoto pole se mění při aktualizaci stavu ASP.NET komponenty na webové stránce.

### 6.1.5 Výkon

ASP.NET se snaží zlepšit výkon oproti klasickým skriptovacím jazykům kompilací kódu na straně serveru do jednoho nebo více DLL souborů. Tato kompilace se automaticky provede při první žádosti o načtení stránky. Tato vlastnost zachovává výhodu skriptovacích jazyků v jednoduchosti, ale díky předkompilování pomáhá k vyššímu výkonu.

### 6.1.6 Master Page

V ASP 2.0 se objevil koncept Master Page, který umožňuje tvorbu stránky pomocí šablony. Master Page obsahuje komponentu ContentPlaceHolders, do kterého pak stránky vkládají vlastní dynamicky obsah.

## 6.2 C#

Jedná se o objektově orientovaný programovací jazyk vyvinutý firmou Microsoft. Spolu s jazykem Visual Basic patří k nejpoužívanějším jazykům pro .NET framework. Syntax tohoto jazyka je založena na C++, ale je ovlivněna i celou řadou dalších jazyku (zejména Javou a Delphi). Programy napsané v C#, jako ostatně všechny jazyky pro .NET, potřebují ke svému běhu .NET Framework (podobně jako Java JRE), což je činí z principu pomalejšími oproti programům, které přistupují k prostředkům přímo (např. jazyk C).

C# má silnou typovou kontrolu, kontrolu délky polí, detekci použití neinicializovaných proměnných a garbage collector (automatické uvolňování paměti podobně jako u Javy).

C# se odlišuje od C++ v mnoha směrech. Nejsou zde globální proměnné nebo funkce. Všechny metody a proměnné musí být deklarovány v rámci třídy. C# nedovoluje automatickou konverzi na typ boolean, tj. nepřevádí automaticky integer na boolean jako tomu je u jazyka C.

Spravovaná paměť nemůže být explicitně uvolněna, toto má na starosti garbage collector (prevence proti memory leaks – neuvolňování paměti). Není zde povolena několikanásobná dědičnost, ačkoliv třídy mohou implementovat větší množství rozhraní (podobně jako v Javě). C# se snaží eliminovat implicitní přetypování na minimum z důvodu bezpečnosti.

C# má unifikovaný systém typů, který se nazývá Common Type System (CTS). Unifikovaný systém typů znamená, že všechny typy, včetně primitivních datových typu jako jsou integery, jsou podtřídou třídy System.Object .Například každý datový typ má metodu ToString(), kterou zdědil. C# umožňuje definování struktur, které mohou být z hlediska výkonu výhodnější než třídy.

Současná verze jazyka je C#3.0, která je součástí .NET Frameworku 3.5. V této verzi se objevily některé vlastnosti inspirované funkcionálními programovacími jazyky jako je Haskell a HL. Asi hlavní novinkou je vznik a začlenění Language Integreted Query (LINQ) do jazyka C#. Tento jazyk má využívá poměrně dost klíčových slov z jazyka SQL (select, from, where atd.) a dovoluje dotazy tímto jazykem nad SQL, XML, kolekcemi, poli atd.

Systém generování dokumentace je obdobný jako u javadoc a je založený na XML syntaxi.

# Kapitola 7

# Implementace

Při implementaci byly použity zejména technologie zmíněné v předchozí kapitole, tedy ASP.NET, C# a MS SQL. Pro detailní pochopení používání těchto technologií doporučuji: [17], [18], [19]. Vzhledem k rozsahu aplikace (cca 900kB zdrojových kódů) jsou v této kapitole popsány pouze některé zajímavější problémy a nejsou jde uvedeny detailní postupy, ale pouze principy fungování.

## 7.1 Obecný popis

Aplikace využívá MasterPage pro generování výchozího vzhledu. Jednotný vzhled je také zajištěn pomocí definice CSS a skinů, které určují vzhled komponent. Do MasterPage je také umístěn kód, který se opakuje na každé webové stránce. Je zde řešeno přepínaní zdrojových xml souborů, které obsahují informace o struktuře menu na základě role přihlášeného účastníka, či základní zabezpečení, aby se účastník nemohl dostat na stránku nepřihlášen.

K datům se přistupuje pomocí adapterů, které jako zdroj dat používají DataSety, tj. kolekce, které obsahují část dat z databáze. Tuto vrstvu pak lze přímo propojit s komponentami prezentační vrstvy pomocí ObjectDataSource .Také si lze vytvořit mezivrstvu, která tyto data ještě nějakým způsobem může předzpracovat. V souborech *.aspx.cs je kód, který slouží k řízení logiky stránky *.aspx, ve které je deklarativně popsán vzhled stránky (code-behind model).

Ke komponentám na *.aspx stránce je možno přistupovat objektově orientovaných přístupem, což umožňuje poměrně snadnou modifikaci libovolných parametrů komponenty (např. datových zdrojů komponent či skrývání/zobrazení určitých částí stránky atd). Do *.aspx stránky lze začlenit i výkonný C# kód, ale bylo by v rozporu s MVC modelem, neboť logická a prezentační vrstva by neměla být rozdělena. Reakce na chování uživatele v systému je ve většině případů řešena odchytáváním událostí. Je však možné odchytávat i celou řadu dalších událostí, které nemají k uživateli přímý vztah, jedná se například o události vznikající v komponentách. Časté využití těchto událostí v realizovaném systému je zejména při předzpracovaní vkládaných dat z komponent

## 7.2 Zavedení pojmu

Abychom lépe porozuměli následujícímu textu, zavedeme si několik pojmu. Tyto pojmy se týkají zejména stromové strukturu.

Nejdříve si ujasníme rozdíl mezi úkolem a typem úkol:

**Typ úkolu** – činnost, která se může dít opakovaně, není uvedeno datum začátku a konce.

**Úkol** – konkrétní realizace úkolu, který se koná v určitém čase.

Termíny mohou být několika typů:

**Rodič** – úkol, který obsahuje alespoň jednoho potomka. Na 7.3 je to uzel s červenou a černou barvou.

**Potomek** – úkol, který má rodiče.

**List** – úkol, neobsahuje žádné potomky. Jde tedy o nejspecializovanější termín. Na 7.3 je to uzel se zelenou barvou.

**Kořen** – úkol, který nemá rodiče. Jde tedy o nejvíce obecný termín. Na 7.3 je to uzel s černou barvou.

## 7.3 Hierarchie úkolů

Abychom modelovali WBS (WorkBreakdownStructure) neboli členění úkolů na další úkoly, umožníme, aby jeden termín mohl obsahovat i jiné termíny. Každý úkol obsahuje informaci o tom, kdo je jeho rodič. Úkoly jsou tedy částečně uspořádány.

Tato hierarchická struktura představuje souvislý acyklický graf, tedy strom. K úkolům typu list lze pak přiřazovat různé zdroje. Úkol, který je nadřazený ostatním úkolům, pak agreguje informace z nižších úkolů. Tyto informace jsou náklady, počáteční a koncové datum a stav dokončení. Čím delší má termín dobu trvání, tím větší má váhu na stav dokončení.

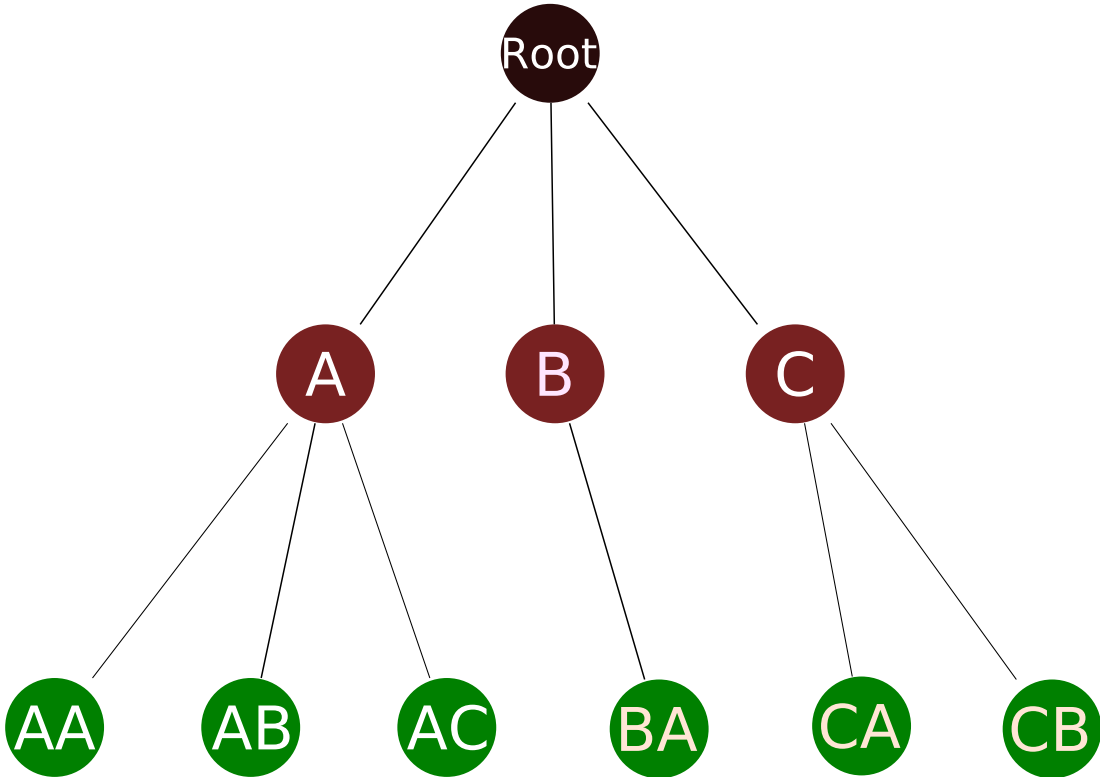
Uvedeme si menší příklad agregace nákladu. Termín A se skládá z termínů AA, AB, AC. Termín AA stojí costAA, termín AB stojí costAB, termín AC stoji costAC. Termín A tedy bude stát costA = costAA + costAB + costAC. Některé hodnoty úkolu jsou tedy závislé na jeho potomcích. Celou situaci můžete vidět znázorněnu na 7.3.

### 7.3.1 Výpočet nákladů

Pro úkoly, které již nemají žádné podúkoly (úkoly typu List), se na základě přiřazených zdrojů vypočítají náklady pro tyto jednotlivé úkoly. Náklady se spočítají z doby trvaní úkolu a nákladů na zdroje, lidi a místa. Následně dojde k vyhledání rodiče právě zaktualizovaného úkolu. Tento rodič si vyhledá všechny své děti, ze kterých získá jejich náklady. Tyto náklady poté agreguje a přepíše si hodnotu ve svém uzlu. Toto se opakuje dokud není dosažen kořen stromu.

### 7.3.2 Získání počátečního a koncového data nadřazeného úkolu

Z úkolu, který je listem, se zjistí jeho rodič, tento rodič si poté vyhledá své potomky. V těchto potomcích se pak snaží najít datum počáteční datum úkolu, který začal nejdříve a koncové datum úkolu, který končí nejpozději. Tyto údaje si uloží do svého uzlu. A následně upozorní rodiče, který si opraví opět údaje o počátečním a koncovém datu. Toto se opakuje dokud není dosaženo kořenového termínu.




Obrázek 7.1: Zobrazení stromové struktury termínů

### 7.3.3 Získání hodnoty stavu dokončení

Tato hodnota se vypočítá na základě zadaných potomku, kde je tato hodnota zadávaná ručně. Úkoly, které trvají déle mají vyšší váhu než úkoly, které trvají kratší dobu. Potomek uvědomí svého rodiče, aby si aktualizoval stav ve svém uzlu. Toto se opakuje dokud rodičovský uzel není kořenem.

### 7.3.4 Rychlost za cenu redudance

Obecně jsou možné dva přístupy, jak získávat údaje z podřízených úkolů. Jedním z nich je projít všechny potomky až do úrovně listů, a získat tak agregovaný údaj. Tento způsob má výhodu, že v databázi nejsou uloženy žádné nadbytečné informace, nevýhodou však je, že při každém zobrazení termínu je třeba projít všechny podřízené úkoly do úrovně listů, tj. v nejhorším případě bychom dosáhli lineární časové složitosti.

Druhý způsob je předpočítat si některé hodnoty, které si průběžně ukládáme do databáze. Výhodou tohoto přístupu je okamžitá dostupnost údajů při čtení tohoto údaje, ovšem vrůstá složitost při zápisu. Při editaci uzlu je třeba projít z listu až ke kořenu a upravit tyto hodnoty, nicméně tato složitost je pouze logaritmická.

V aplikace je implementováno druhé řešení, protože je výrazně rychlejší díky průběžné agregaci mezi výsledku v uzlech. Hodnoty jsou pak rovnou připraveny pro čtení, což má významný výkonností vliv, neboť čtení je prováděno několikanásobně častěji než zápis.

# Kapitola 8

# Ovládání systému

Jedním z důležitých prvku informačního systému pro nasazení do reálného světa je ovladatelnost, systém by měl být maximálně jednoduchý při zachování plné funkčnosti. Ovladatelnost systému často rozhoduje o tom, zda by si jej zákazník zvolil. Také platí, že čím lépe se systém ovládá, tím rychleji se dají zaučit lidé, kteří tento systém budou používat a tím i snížit náklady na školení, čímž vlastně klesne celková cena systému. Není proto divu, že v současné době se vynakládají poměrně velké prostředky na tvorbu uživatelských rozhraní.

## 8.1 Postupné zobrazování prvků

Podle událostí uživatele se dynamicky mění prostředí (např. se objevují další možnosti). Pokud uživatel zadává něco v několika na sebe navazujících krocích, možnosti se postupně objevují dle akce uživatele. Systém se snaží uživatele co nejvíce vést. V případě, že by projektový manažer chtěl přidat pracovníka k úkolu, klikne na skupinu, po té se na stránce objeví členové vybrané skupiny, po výběru člena skupiny se zobrazí blokace a až teprve v této fázi je možné pracovníka přidat.

## 8.2 Hlavní menu

Jedná se o "horní proužek", který slouží k rychlé a přehledné navigaci k tématickým celkům (například Úkoly). Tento proužek je přítomný po celou dobu přihlášení, uživatel tedy nikdy nemá pocit, že se v systému ztratil, neboť kliknutím na ten správný tématický celek se dostane přesně tam, kam chce. Vzhledem k množství položek v menu jsou sémanticky společné části sdružovány, po najetí myši se rozbalí kompletní nabídka. Pod menu se nachází výpis, kde se uživatel zrovna nachází (např. v detailu úkolu), kliknutím je možné se dostat na o úroveň výš (seznam úkolů). Dále zde vidíme přihlášeného uživatele a je možné se i zde odhlásit.

## 8.3 Režim výpis

Po kliknutí na tento tématický celek se zobrazí výpis položek (režim Výpis). Tyto položky lze třídit podle mnoha kriterií kliknutím na název sloupce v záhlaví. Ve spodní časti tabulky jsou nachystaná pole pro vložení záznamu. Záznamy je možno procházet pomocí stránkování ve spodní částí. V levé části se pak nachází operace, v pravé je obvykle odkaz na Detail.


Obrázek 8.1: Snímek z aplikace

## 8.4 Režim Detail

Operace Detail zpřístupňuje detailní rozpis položek, které by se z prostorových důvodu nemusely vejít na obrazovku v režimu Výpis. Zobrazuje se zde celý výpis a je možno zde provádět další operace, které mají souvislosti se zvolenou položkou. Například u úkolu to může být přiřazení veškerých lidí, míst, zdrojů, vkládání poznámek, apod. V režimu Detail také bývají často zobrazeny informace, které mají vztah k vybrané položce zobrazené v Detailu. U typu úkolu to může být například seznam uživatelů, kteří mají s daným úkolem nejvíce zkušeností atd. Dalo by se tedy říct, že v režimu Detail se odehrává většina operací této aplikace.

## 8.5 Navigace v hierarchické struktuře úkolů

Úkoly do sebe mohou být zanořovány. Je tedy nutné řešit hierarchické uspořádání. Aplikace to řeší dvojím způsobem:

- Zooming – V detailu každého úkolu jsou vidět jeho podřízené úkoly, kliknutím na detail podřízeného úkolu si zobrazíme tento úkol. Pokud si budeme přát vidět nadřazený úkol, klikneme na nadřazený úkol. Způsob navigace nám tedy umožňuje se pohybovat po jednotlivých uzlech stromu. Úkoly jsou uzly v tomto stromu, jednotlivé vazby typu nadřízený-podřízený úkol jsou hrany.

  Výhodou tohoto přístupu je přehlednost a snadnost navigace, která nám umožní snadné zanořování a vynořování. Nevýhodou ale může být, že náš pohled vždy směřuje pouze do jednoho uzlu.

- Stromové zobrazení – Toto zobrazení se snaží o zobrazení celé stromové struktury. Je možné zobrazit či skrýt různé části stromové struktury. Tento náhled slouží pro


Obrázek 8.2: Snímek z aplikace

globální pohled, kde jsou rychle vidět souvislosti mezi úkoly.

## 8.6 Osobní kalendář

Aby každý mohl být snadno seznámen se svými povinnostmi, jsou plánované začátky a konce zaznamenány do kalendáře. Uživatel systému tedy pomocí listování kalendářem snadno vidí, kdy se mu blíží termíny. Pokud chce uživatel zobrazit detaily týkající se začátku či konce

úkolu zaznamenaném v kalendáři, kliknutím na tuto položku zobrazí podrobnější informace o tomto úkolu. Kliknutím na konkrétní datum se mu zobrazí seznam úkolu s termíny dokončení, na kterých by měl ten den pracovat, což je velmi výhodné zejména pokud pracovník pracuje v několika projektech zároveň na částečný úvazek. Tento jednoduchý a přehledný způsob tak dokáže pracovníka informovat o jeho náplni práce.

## 8.7 Cíle

Primární snahou byl zejména jednotný vzhled a princip používání, neboť systém si stane přínosný zejména v případě, že do něj bude vstupovat co nejvíce lidí. Celá logika ovládání je založena pouze na několika hlavních principech, takže naučit se pracovat s aplikací ve velmi krátkém čase by neměl být problém.

Každý uživatel uvidí pouze volby, které jsou založeny na jako roli v organizaci, což mu pohyb v systému opět zjednoduší, neboť mu bude do značné míry skryta složitost systému jako celku. Přestože je aplikace realizované přes webové rozhraní, je zde snaha o udržení způsobu ovládání klasické standalone aplikace, aby si uživatel co nejrychleji mohl zvyknout. Všechny tyto vlastnosti by měly přispět ke snadné integraci maximálního počtu uživatelů do systému.

# Kapitola 9

# Závěr

V této práci jsem se zaměřil na analýzu současného stavu aplikací zabývající se podporou projektového řízení. Na základě této analýzy jsem stanovil nejdůležitější společné funkce, které se vyskytovaly ve všech těchto aplikacích. Vybíral jsem si převážně z volně dostupných produktů. Většina těchto produktů se snaží alespoň částečně podobat Microsoft Project.

Dále jsem doplnil funkce, které jsem v těchto aplikacích nenašel a z mého pohledu jsou důležité. Jednalo se zejména o podporu týmové spolupráce, zavedení rolí (projektoví manažeři, techničtí vedoucí, správci zdrojů, běžní pracovníci), možnost tvorby odhadu systémem na základě realizovaných projektů atd.

Cílem této práce bylo navrhnout a zrealizovat aplikaci, která by pomáhala při projektovém řízení nejen při tvorbě plánu, ale i navazujících činnostech (řízení komunikace, distribuce pokynů). Tato aplikace obsahuje základní funkce z již existujících aplikací, které jsem analyzoval. Je zde také předělán zejména uživatelský model, díky němuž je umožněna lepší paralelizace při tvorbě plánu. Techničtí vedoucí a správci zdrojů zadávají informace do systému nezávisle na projektu. Projektový manažer tyto informace pak opakovaně využívá ve svých projektech. Byly doplněny funkce, které by uživatelům systému měly ulehčit komunikaci (automatické generování zpráv, zasílání emailu, vkládání poznámek atd.). Díky zpracování informací z již realizovaných úkolu je také umožněna zpětná vazba pro projektového manažera, díky níž může lépe nastavit parametry úkolu pro další plánování.

Primární požadavky, které jsem na aplikaci kladl, byly zrealizovány. Celá aplikace je poměrně rozsáhlá, zdrojové kódy aplikace mají cca. 950Kb v cca. 70 souborech. Při tomto rozsahu aplikace se již začaly i projevovat výhody MVC ve znovupoužitelnosti funkcí jednotlivých vrstev.

## 9.1 Možnosti rozšíření

Další vývoj tohoto systému by mohl směřovat například k dolování dat. Bylo by například zajímavé zjistit, který manažer má lepší výsledky v jakých úkolech a proč, nebo kteří zaměstnanci jsou pro které úkoly statisticky nejvhodnější (drží termíny). Tato oblast je důležitá pro projekt, neboť opět umožňuje lepší možnosti v plánovaní. V systému se může nacházet celá řada užitečných informací, které ale nejsou přímo viditelné, neboť nevznikají přímým zadáním údajů, ale hromaděných velkého množství dat, které lze následně zpracovat. Systém by se díky těmto informacím mohl učit a nabízet projektovému manažerovi optimální varianty.

Dalším směrem by mohlo být umožnit cestovat manažerovi v čase projektu. Toto by

bylo dobré zejména k analýze, kde se v projektu udělala chyba. Zpětná analýza projektu je velmi důležitá pro budoucí plány. Trochu nadsazeně by se dalo říci, že dobrého manažera dělají zejména velké zkušenosti, které získá realizací projektu.

Projekt by také mohl být rozšířen o celou řadu vizualizačních prvků, zejména Ganttova diagramu, síťové diagramu, apod. Dále grafické upozornění na zvláštní stavy těchto úkolů.

Dalším praktickým rozšířením by mohla být podpora importu a exportu do MS Project formátu.

Systém by také mohl obsahovat systém pro správu souboru SVN, což by bylo velmi výhodné zejména pro softwarové produkty. Aplikace je v současné době tento problém částečně schopna řešit pomocí přidání libovolného množství odkazů na tyto soubory do poznámek.

# Literatura


[1] Baca, C.; Jansen, P.: *PMP – Project Management Professional Workbook*. New York, USA: Sybex, 2003, ISBN 0782142400, 284 s.

[2] Kerzner, H.: *Project Management – A Systems Approach to Planning, Scheduling, and Controlling*. New York, USA: Wiley, 2001, ISBN 0-471-39342-8, 1406 s.

[3] Martin P, Tate K.: *Project Management*. New York, USA: Wiley, 2001, ISBN 0-471-13503-8, 271 s.

[4] Barashev, D.: Domovská stránka GanttProject. [online], [cit.21.12.2007]. Dostupné na URL: http://ganttproject.biz/.

[5] Brooks, F.: *The Mythical Man-Month*. Boston, Massachusetts, USA: Addison-Wesley, 1995, ISBN 0-201-83595-9, 336 s.

[6] Hyndrák, K.: *Vytváříme projekty v programu Microsoft Project 2000*. Praha: Computer Press, 2000, ISBN 80-7226-329-3, 303 s.

[7] KPlato Team: Domovská stránka KPlato. [online], [cit.13.12.2007]. Dostupné na URL: http://www.koffice.org/kplato/.

[8] Projity Incorporated: Domovská stránka OpenProj. [online], [cit.19.12.2007]. Dostupné na URL: http://openproj.org/openproj.

[9] Dr. Ing. Tomáš Šubrt: ZIP Projektové řízení. [online], [cit.28.4.2007]. Dostupné na URL: http://etext.czu.cz/php/skripta/skriptum.php?titul_key=77.

[10] Wikipedia: Project. [online], [cit.21.4.2007]. Dostupné na URL: http://en.wikipedia.org/wiki/Project.

[11] Wikipedia: Project Management. [online], [cit.22.4.2007]. Dostupné na URL: http://en.wikipedia.org/wiki/Project_management.

[12] Wikipedia: Risk Management. [online], [cit.25.4.2007]. Dostupné na URL: http://en.wikipedia.org/wiki/Risk_management.

[13] Staníček Z.:Řízení projektů I. díl – Podstata řízení projektů. [online],[cit.26.4.2007]. Dostupné na URL: http://www.projektoverizeni.cz/pages/rizeni/cteni/RPclanek.pdf.

[14] Wikipedia: C Sharp (programming language). [online], [cit.28.4.2007]. Dostupné na URL: http://en.wikipedia.org/wiki/C_Sharp_%28programming_language%29.

[15] Wikipedia: Asp.NET. [online], [cit.28.4.2007]. Dostupné na URL: http://en.wikipedia.org/wiki/Asp.net.

[16] Microsoft Corporation: ASP.NET. [online], [cit.28.4.2007]. Dostupné na URL: http://asp.net.

[17] Microsoft Corporation: ASP.NET – Data Access Tutorial. [online], [cit.30.4.2007]. Dostupné na URL: http://asp.net/learn/data-access/.

[18] Microsoft Corporation: ASP.NET – QuickStart Tutorial. [online], [cit.1.5.2007]. Dostupné na URL: http://quickstarts.asp.net/QuickStartv20/aspnet/Default.aspx.

[19] Microsoft Corporation: ASP.NET Web Application. [online], [cit.2.5.2007]. Dostupné na URL: http://msdn.microsoft.com/en-us/library/ms644563.aspx.