

Metody odhadování pracnosti softwarových projektů

Methods for estimating the time consumption of software projects

Bc. Michal Kozel

Diplomová práce
2011



Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
akademický rok: 2010/2011

ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení: **Bc. Michal KOZEL**
Osobní číslo: **A08769**
Studijní program: **N 3902 Inženýrská informatika**
Studijní obor: **Informační technologie**

Téma práce: **Metody odhadování pracnosti softwarových projektů**

Zásady pro vypracování:

1. Definuje softwarové projekty.
2. Zpracujte rešerši existujících metod pro odhadování pracnosti.
3. Aplikujte vybrané metody na reálný projekt.
4. Vypracujte porovnání metod.
5. Analyzujte způsob využívání metod odhadu v softwarových firmách.

Rozsah diplomové práce:

Rozsah příloh:

Forma zpracování diplomové práce: **tištěná/elektronická**

Seznam odborné literatury:

1. Boehm, B. W., et al.: **Cost Models for future Software Life Cycle Process: COCOMO 2.0.** The Netherlands, Amsterdam, Science Publishers 1995.
2. Boehm, B. W., et al.: **COCOMO II Model Definition Manual.** USA, Los Angeles, University of Southern California 1999.
3. **International Function Points User Group: Function Point Counting Practices Manual: Release 4.1.1999.**
4. Sedláčková, J.: **Cenové odhady softwarových projektů.** Masarykova univerzita v Brně, Fakulta informatiky, Brno, 2005. Diplomová práce.
5. Anda, B. **Comparing Use Case based Estimates with Expert Estimates.** Proc. of Empirical Assessment in Software Engineering (EASE), Keele, United Kingdom, April 8-10, 2002.
6. SMITH, John. **The Estimation of Effort Based on Use Cases [online].** Somrs : IBM, 2003 [cit. 2011-01-24]. Dostupné z WWW: <ftp://ftp.software.ibm.com/software/rational/web/whitepapers/2003/finalTP171.pdf>.

Vedoucí diplomové práce:

Ing. Radek Šilhavý, Ph.D.

Ústav počítačových a komunikačních systémů

Datum zadání diplomové práce:

24. února 2011

Termín odevzdání diplomové práce:

18. května 2011

Ve Zlíně dne 24. února 2011

prof. Ing. Vladimír Vašek, CSc.

děkan



doc. Mgr. Roman Jašek, Ph.D.

ředitel ústavu

ABSTRAKT

Kozel, M. Metody odhadování pracnosti softwarových projektů.

Diplomová práce. Zlín, 2011

Diplomová práce je založena na problematice odhadování pracnosti softwarových projektů. Hlavním cílem je analyzovat vybrané metody pro odhadování pracnosti a zjistit využívání metod pro odhadování pracnosti v softwarových společnostech. Z rešerše existujících metod jsme vybrali Use Case Points a COCOMO II. Srovnání bylo provedeno na reálném projektu. Primárním zdrojem k získání informací o stavu odhadování v softwarových firmách byla využita metoda dotazování v elektronické podobě. Dle získaných výsledků jsme se vyjádřili k stanoveným hypotézám.

Klíčová slova: software, projekt, odhady pracnosti, Use Case Points, COCOMO II, dotazníkové šetření.

ABSTRACT

Kozel, M. Methods for estimating the time consumption of software projects.

Diploma thesis. Zlín, 2011

This thesis is based on the issue of cost estimation of software projects. The main aim of this thesis is to analyze selected methods for cost estimation and to determine the use of methods for cost estimation in software companies. We chose Use Case Points and COCOMO II from literature retrieval. The comparison was carried out on a real project. The primary source for obtaining information about cost estimation in software companies was used questioning method in an electronic form. According to the obtained results, we expressed to the defined hypotheses.

Keywords: software, project, cost estimation, Use Case Points, COCOMO II, survey questionnaire.

Poděkování

Za cenné rady a připomínky během zpracování diplomové práce děkuji svému vedoucímu panu Ing. Radkovi Šilhavému, Ph.D.

Motto

"You can not plan if you can not measure, and if you fail to plan, you have planned to fail."

Prohlašuji, že

- beru na vědomí, že odevzdáním diplomové/bakalářské práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že diplomová/bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk diplomové/bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně a jeden výtisk bude uložen u vedoucího práce;
- byl/a jsem seznámen/a s tím, že na moji diplomovou/bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – diplomovou/bakalářskou práci nebo poskytnout licenci k jejímu využití jen s předchozím písemným souhlasem Univerzity Tomáše Bati ve Zlíně, která je oprávněna v takovém případě ode mne požadovat přiměřený příspěvek na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše);
- beru na vědomí, že pokud bylo k vypracování diplomové/bakalářské práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky diplomové/bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem diplomové/bakalářské práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji,

- že jsem na diplomové práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze diplomové práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně dne 2. května 2011

.....
podpis diplomanta

OBSAH

ÚVOD	9
I TEORETICKÁ ČÁST	10
1 SOFTWAREVÉ PROJEKTY	11
1.1 CO JE SOFTWAREVÝ PROJEKT	11
1.2 METODIKY PROJEKTU	12
1.2.1 Tradiční metodiky	12
1.2.2 Agilní metodiky	16
1.3 RIZIKA.....	19
1.4 SOUČASNÁ SOFTWAREVÁ KRIZE	21
1.5 ODHADY SOFTWAREVÝCH PROJEKTŮ	23
2 REŠERŠE EXISTUJÍCÍCH METOD	25
2.1 ANALOGIE	25
2.2 EXPERTNÍ ODHADY.....	30
2.3 PARAMETRIC ESTIMATING	32
2.3.1 Cost estimating relationship	32
2.4 WIDEBAND DELPHI	34
2.4.1 Delphi proces	35
2.4.2 Výběr týmu.....	35
2.4.3 Kickoff meeting	36
2.4.4 Individuální příprava	36
2.4.5 Diskuze nad odhady	37
2.4.6 Sestavení výsledků	37
2.4.7 Recenze výsledků.....	38
2.5 METODA FUNKČNÍCH BODŮ.....	38
2.6 USE CASE POINTS.....	44
2.7 COCOMO II.....	48
II PRAKTICKÁ ČÁST	52
3 SROVNÁNÍ VYBRANÝCH METOD	53
3.1 USE CASE POINTS.....	53
3.2 COCOMO II.....	56
3.3 VYHODNOCENÍ SROVNÁNÍ.....	58
4 STAV VYUŽÍVÁNÍ METOD ODHADU V SOFTWAREVÝCH FIRMÁCH	60
4.1 ZPRACOVÁNÍ DOTAZNÍKU (SEZNÁMENÍ S DOTAZNÍKEM)	60
4.2 STANOVENÍ HYPOTÉZ	65
4.3 VYHODNOCENÍ DOTAZNÍKU	65
4.4 PLATNOST NEBO ZAMÍTNUTÍ HYPOTÉZ	71
4.5 SOUČASNÝ STAV V SOFTWAREVÝCH FIRMÁCH.....	72
ZÁVĚR	74
ZÁVĚR V ANGLIČTINĚ	76
SEZNAM POUŽITÉ LITERATURY	78

SEZNAM OBRÁZKŮ	81
SEZNAM TABULEK.....	82
SEZNAM GRAFŮ	83
SEZNAM PŘÍLOH.....	84

ÚVOD

Vývoj software je velmi složitá činnost, proto se téměř osmdesát procent projektů potýká s nějakým druhem problému. Navzdory tomu, že je projekt nakonec úspěšně dokončen, tak časové zpoždění a překročený rozpočet můžou znamenat nemalou finanční ztrátu. Dle statistického průzkumu bylo zjištěno, že počet neúspěchů a množství zbytečně vynaložených peněz je v softwarovém odvětví daleko vyšší než v jiných oblastech.

Nepřesný odhad pracnosti vývoje software je jednou z nejčastějších příčin, která vede k selhání projektu. Zatímco příliš nízký odhad může vést k problémům s řízením projektu, časovému zpoždění, překročení rozpočtu a nízké kvalitě software, tak na druhé straně příliš vysoký odhad může vést ke ztrátě obchodních příležitostí a neefektivnímu využívání zdrojů. Zmíněné příčiny motivují softwarové firmy k provádění průzkumů s jednoznačným cílem zpřesnit odhad pracnosti software.

Cílem práce je seznámení čtenáře s podstatou samotného softwarového projektu, jak správně skloubit jeho obchodní a technické cíle. Práce se zabývá existujícími procesy, které se používají při vývoji počítačových aplikací. Představeny budou tradiční vývojové způsoby, které byly vytvořeny před několika desetiletími, tak i moderní agilní metody staré pouze několik let.

V rešerši metod pro odhad pracnosti je uvedeno sedm nejčastěji používaných zástupců. Některé z postupů jsou čistě závislé na lidském faktoru, další skupina jsou parametrické matematické modely a zbývající skupina je kombinace obou předchozích, kdy výstup člověka se stává vstupem pro matematický model.

Z přehledu metod byly vybrány Use Case Points a COCOMO II, které byly použity na reálném projektu. Záměrně byly vybrány právě tyto dvě metody z důvodu dostupných všech informací pro provedení odhadů pracnosti.

Závěrem bude zpracován anonymní dotazník, jehož účelem bude zjistit, v jakém rozsahu jsou softwarovými společnostmi využívány metody pro odhad pracnosti software. Na základě výsledku dotazníku budou také ponechány v platnosti nebo vyvráceny stanovené hypotézy.

I. TEORETICKÁ ČÁST

1 SOFTWAREVÉ PROJEKTY

1.1 Co je softwarový projekt

Dobrá definice projektu je klíčem k jeho úspěchu. Jedná se o základ, na kterém by měl být postaven každý projekt. Pokud definice není dostatečná, může se stát, že projekt bude neúspěšný.

Jak se dostat z myšlenek nebo obchodních potřeb k řádně definovanému a plánovanému projektu se schváleným obchodním případem? Níže je uveden postup v sedmi hlavních etapách. [1]

1. Definice fáze plánování a dohody: vytvořit plán pro jednotlivé fáze projektu.
2. Vyhodnotit obchodní potřeby: jsou pro tento projekt vůbec potřebné?
3. Nalézt nejlepší řešení: nerealizovat první myšlenku, která nastane.
4. Prozkoumat náklady a přínosy: vytvořit finanční obchodní případy.
5. Definovat projekt: role, rizika, zdroje atd.
6. Odhadnout a naplánovat první etapu: naplánovat detailně první etapu, následující pouze obrysově.
7. Rozhodnout se: jít nebo nejít.

Softwarový projekt lze definovat jako dočasnou činnost, která má stanovené počáteční datum, konkrétní cíl, podmínky, rozpočet, plán, fixní konečné datum a je v něm zainteresována spousta osob (částečně nebo na plný úvazek). Všichni vědí, co mají dělat a jakmile svůj úkol splní, tak je jejich práce končí.

Projekt je soubor činností, které mají cíl. Zároveň by měla existovat odpověď na každou otázku typu „proč“. Níže je uvedeno několik základních otázek, na které je vhodné nalézt odpovědi na počátku projektu.

- Jaké strategické rozhodnutí vedlo k projektu?
- Jak souvisí projekt s obchodním plánem společnosti?
- Navazuje tento projekt na jiný projekt?
- Jak bude naloženo s výsledky projektu?
- Co se stane, pokud projekt skončí nezdarem?
- Vztahuje se tento projekt k jinému projektu, který probíhá ve stejném časovém úseku?

- Proč nebyl projekt proveden dříve?

1.2 Metodiky projektu

Metodika softwarového projektu je rámec, který je používán k definování struktury, plánování a kontrole vývojového procesu. Zahrnuje definici specifických výstupů a artefaktů, které jsou vytvářeny a dokončovány projektovým týmem při vytváření nebo údržbě softwarové aplikace. [2]

Během minulých let bylo vytvořeno mnoho takových rámců, kdy každý disponuje svými silnými a slabými stránkami, proto jeden vývojový rámec není vhodný pro všechny projekty. Pro různé druhy projektů, které jsou založeny na rozmanitých technických, organizačních a týmových okolnostech se hodí jiný metodický rámec. [2]

Vývojové rámce jsou často vázány na nějakou organizaci, která dále rozvíjí, podporuje využití a propaguje metodiky daného rámce.

Rámec je de facto kostra nebo obal pro metody, který bývá parametrizován nebo rozšířen před použitím v konkrétním projektu.

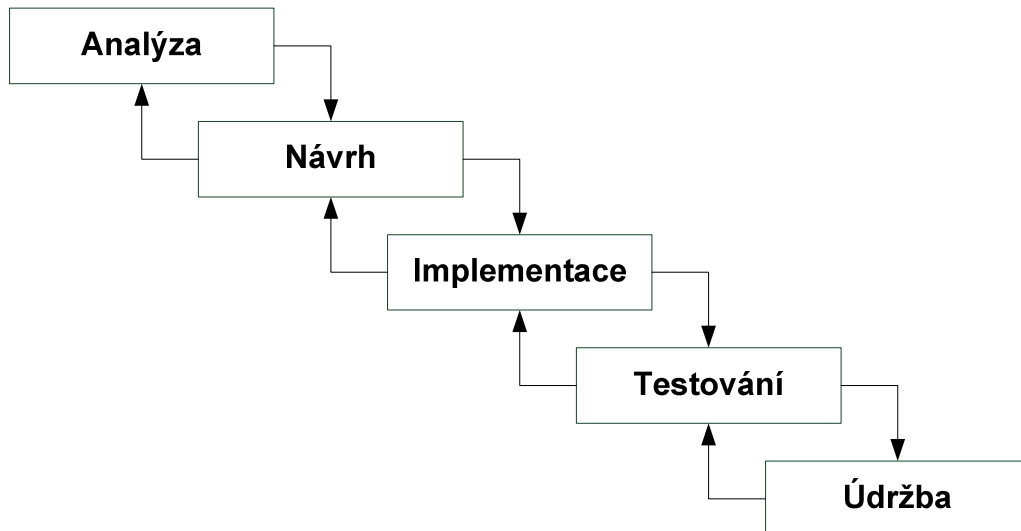
1.2.1 Tradiční metodiky

Vodopádový model

Klasický vodopádový přístup byl představen v roce 1970. Takto je pojmenován, protože může být graficky zachycen jako kaskáda s jednotlivými stupni. Jsou to analýza požadavků, návrh, implementace, testování a předání zákazníkovi. V té době se jednalo o významný krok kupředu v disciplínách zabývajících se vývojem software. [3] Obrázek (Obr. 1) zachycuje jednoúrovňovou zpětnou vazbu, která však nebyla součástí původního modelu, ale byla přidána v průběhu let pro jeho zlepšení. Původní vodopádový přístup měl malou nebo vůbec žádnou zpětnou vazbu mezi jednotlivými etapami, stejně tak jako v kaskádě nemůže téct voda zpět do kopce, protože je přitahována gravitací. [4] Tento model samozřejmě funguje pouze v tom případě, pokud požadavky jsou perfektně specifikovány, než se přejde k návrhu, pokud návrh je dokonalý, než se přejde k implementaci atd. a samozřejmě pokud zákazník nemění své požadavky. Bohužel, žádná z těchto věcí nikdy nebude pravda. Základními principy vodopádového modelu tedy jsou:

- Projekt je rozdělen do sekvenčních fází s určitým stupněm překrývání.

- Důraz je kladen na plánování, časový harmonogram, termíny, rozpočet a dodání celého systému v jednom čase.
- Kontrola projektu je prováděna ve všech jeho fázích na základě analýz a především před přechodem do další fáze.



Obr. 1. Vodopádový model se zpětnou vazbou

Prototypový model

Jedná se o vývojový přístup, jehož cílem je vytvořit prototyp, tzn. nekompletní verzi software, která disponuje základními nebo typickými funkcemi pro finální produkt.

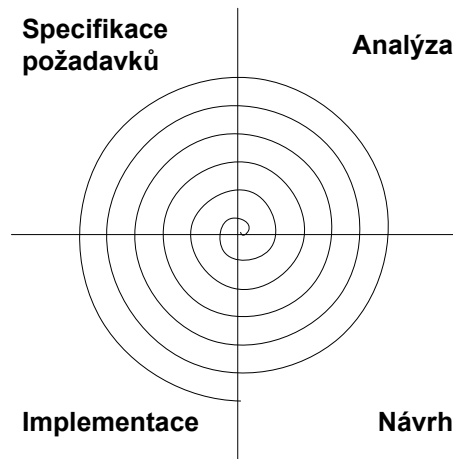
Nejedná se o samostatný vývojový model, ale spíše je součástí rozsáhlejšího tradičního přístupu, např. přírůstkového nebo spirálového. Jeho cílem je snížit rizika projektu tak, že jej rozdělí do menších segmentů a umožňuje provádět jednodušší změny ve vývojovém procesu. Konečný uživatel nebo zákazník je zahrnut do vývoje produktu, což zvyšuje pravděpodobnost, že bude akceptována jeho finální podoba. Opakující se proces změny je prováděn do té doby, než jsou splněny požadavky budoucích uživatelů. [5]

Většina prototypů je vyvíjena s očekáváním, že budou vyřazeny, je však možné, že některé z nich budou zařazeny do finálního systému.

Spirálový model

Spirálový model vznikl rozšířením vodopádového a prototypového modelu. Jeho charakteristickou vlastností je analýza a řízení rizik v každé vývojové fázi. Spirálový přístup bývá zachycen jako proces, který prochází několikrát čtyřmi kvadranty, kdy každý z nich reprezentuje nějakou aktivitu. Na počátku každé iterace jsou tázány jednotlivé

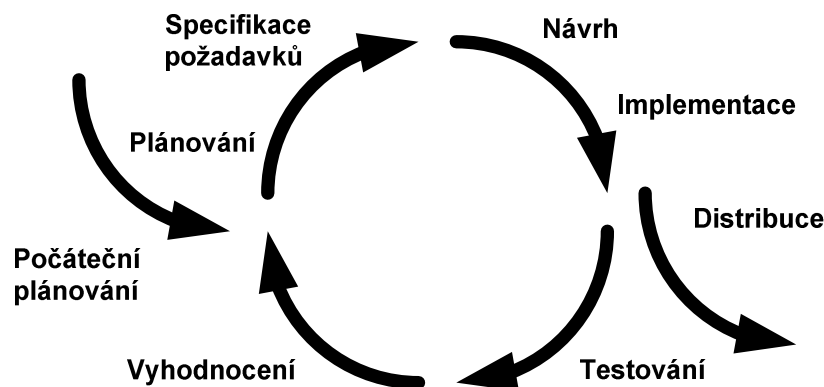
subjekty zapojené do vývoje, zda jsou splněny všechny jejich podmínky pro úspěšné ukončení iterace a na konci každého cyklu je provedena jejich revize. Použití tohoto modelu je vhodné pro vývoj rozsáhlých systémů. [3] [7]



Obr. 2. Spirálový model

Přírůstkový model

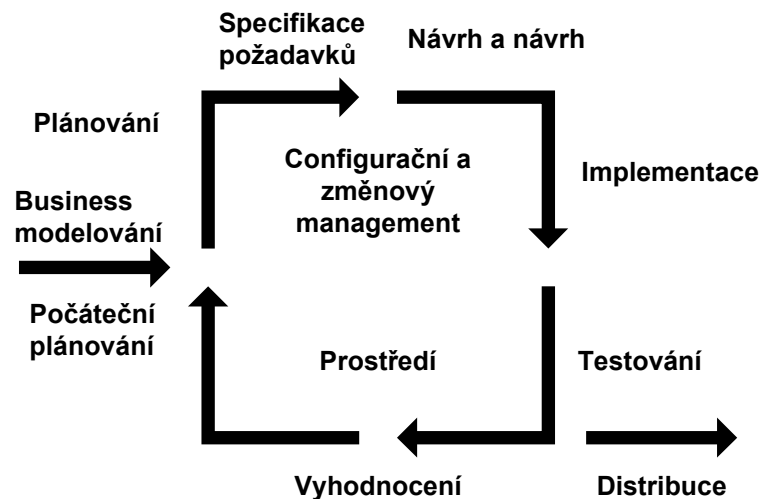
Inkrementální přístup je kombinace sekvenčního a iteračního modelu. Podstatou tohoto modelu je rozdělení projektu do menších segmentů, kdy na každém z těchto segmentů probíhá malý vodopádový přístup a je dokončen před započítáním dalšího přírůstku. V každém přírůstku je cílem vytvořit kvalitní systém, který ale neobsahuje kompletní funkcionalitu. [7] Jednou z největších výhod je jeho flexibilita, protože dokáže pružně reagovat na změny ve specifikaci v průběhu vývoje. [3] Možným rizikem je v tomto modelu situace, kdy může dojít k vyčerpání rozpočtu nebo nedodržení časového harmonogramu.



Obr. 3. Iterační model v rámci modelu přírůstkového

Rational Unified Process

Je možné jej použít pro jakýkoliv rozsah projektu a vzhledem k jeho komplexnosti může být přizpůsoben specifickým požadavkům. Přesto je RUP vhodné použít pro rozsáhlé projekty a větší vývojové týmy, protože klade důraz na analýzu, návrh, plánování, řízení zdrojů a dokumentaci. [8]



Obr. 4. Iterační model v rámci RUP

RUP se skládá ze čtyř základních elementů, na kterých je postaveno celé modelování. [9]

Pracovníci odpovídají na otázku „kdo“. Pracovník definuje chování a odpovědnost jedince nebo skupiny. Chování pracovníka je zachyceno prostřednictvím činností. Jeho odpovědnost je definována k meziproductům, které může vytvářet, modifikovat nebo kontrolovat. Jedná se spíše o roli, která může být přidělena více fyzickým osobám.

Činnosti odpovídají na otázku „jak“. Jedná se o jednotku práce, kterou má provést jedinec nebo skupina a měla by vyústit v konkrétní výsledek. Má jasně definovaný účel, který představuje vytvoření nebo modifikaci meziproductu. Činnost obvykle ovlivňuje pouze několik meziproductů a vztahuje se na jednoho pracovníka. Má definovaný vstupní a výstupní meziproducty.

Meziproducty odpovídají na otázku „co“. Jedná se o část informace, která je produkována, modifikována nebo použita v rámci procesu. Meziproducty jsou hmatatelné výsledky projektu a jsou používány pracovníky jako vstupy a výstupy jednotlivých činností. Za vytvoření a správnost meziproductu zodpovídá definovaný pracovník.

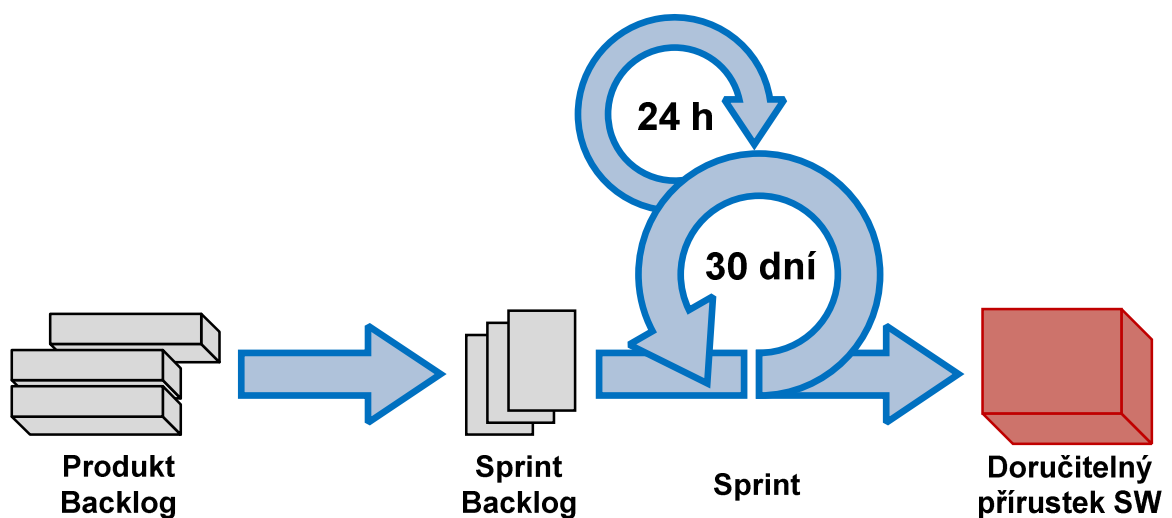
Pracovní procesy odpovídají na otázku „kdy“. Pracovním procesem se myslí smysluplná činnost vedoucí k vytvoření požadovaného výsledku. Pracovní proces může v UML

představovat sekvenční diagram, diagram spolupráce nebo diagram činnosti. Každý proces se skládá z několika činností a zachycuje interakce mezi pracovníky.

1.2.2 Agilní metodiky

SCRUM

Jedná se iterativní přírůstkový vývoj, který se skládá z několika fází. Jednotlivé fáze se nazývají sprinty, které mohou dle potřeb projektu trvat 24 hodin až jeden měsíc. Během každého sprintu tým vytvoří potenciálně doručitelný produkt zákazníkovi. [10] Před započatím dalšího sprintu zákazník připomínkuje danou verzi a stanový seznam požadavků pro další iteraci, které by měly být naplněn. Během sprintu nikomu není dovoleno zasahovat do vytyčených požadavků. Sprint je časový rámeček, který je zakončen předvedením softwaru zákazníkovi. [11]



Obr. 5. SCRUM

Lean Software Development

Metodika vychází z přístupu, používaném v automobilovém průmyslu firmou Toyota „Just in time“. Metoda je složena ze sedmi základních principů, jejichž úkolem je dodat zákazníkovi produkt, který pro něj má přínos.

Plýtvání je odstraněno selekcí opravdu významných funkcí pro systém, je jim přiřazena priorita a zákazníkovi jsou doručovány v malých dodávkách. Klade důraz na rychlost a efektivitu vývoje a spoléhá na rychlou komunikaci mezi vývojáři a zákazníkem. Vývoj produktu se posouvá dál prostřednictvím přání zákazníka. [12]

Rozhodovací pravomoci jsou přiřazeny jednotlivcům nebo malým týmům, což je dle výzkumu pro vývoj více efektivní než klasická hierarchická struktura. Snaží se dokonale využívat týmové zdroje, což znamená, že každý je produktivní tolik času, jak jen to je možné. Je zaměřen na výslednou práci a co nejméně na závislosti v rámci týmu. Tato metodika také doporučuje, aby automatizované unit testy byly napsány ve stejný čas, jako je psán samotný kód. [13]

Feature Driven Development (FDD)

FDD je modelově řízený iterativní přírůstkový proces. Je složen z pěti základních fází. První tři fáze jsou sekvenční (vytvoření modelu, vypracování seznamu vlastností a plánování podle vlastností), zbývající dvě jsou iterativní (návrh podle vlastností a implementace podle vlastností).

Pře začátkem vývoje se vytvoří celkový model systému, který se nezabývá všemi detaily. Jeho cílem je poskytnout globální pohled na systém a stanovit hlavní směr vývoje, kterým se bude projekt ubírat. V iteračních fázích dochází k opakovanému návrhu a implementaci jednotlivých vlastností. Pod vlastností je možné si představit určitou část funkcionality. Jednotlivé části modelu jsou blíže specifikovány. Iterace jsou krátké, s dobou trvání obvykle čtrnáct dní, kdy na konci každé iterace je úkolem zákazníkovi dodat fungující produkt s novými vlastnostmi. Toto dělá metodiku dostatečně flexibilní, protože zákazník může reagovat na nové změny postupně. [14]

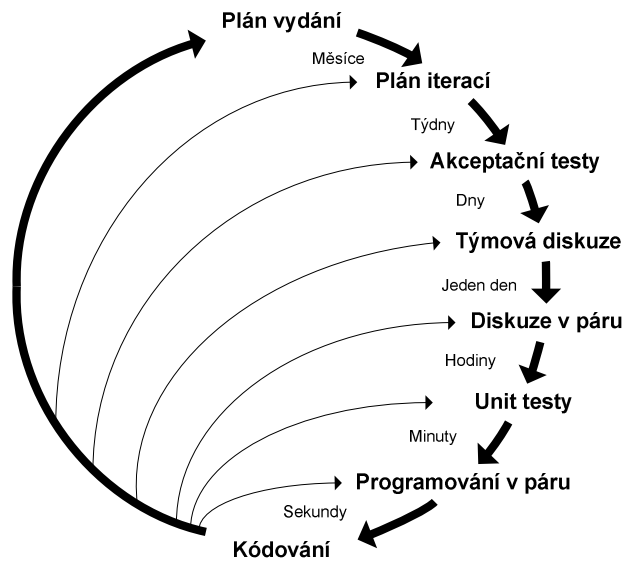
Extrémní programování (XP)

Jedná se o nejhojněji používaného zástupce agilních metodik. Extrémní je nazýváno z toho důvodu, protože veškeré jeho činnosti jsou dotaženy do extrému. Je vhodné pro malé projekty s nejasným nebo rychle se měnícím zadáním. [15]

Mezi všemi zúčastněnými subjekty je udržována kvalitní komunikace, protože extrémní programování ji podporuje a v některých případech přímo vyžaduje.

Snahou je programovat pouze to, co je nutné ke splnění požadavků zákazníka. Zbytečně se nevytváří funkcionality, které by se mohly někdy v budoucnu někomu hodit. Všechny práce vychází z předpokladu, že požadavky se můžou ze dne na den změnit a jakákoliv práce navíc, by tak byla ztracena.

Pro každou část kódu jsou vytvořeny unit testy, které jsou schopny ověřit, zda při změně kódu nedošlo k porušení funkcionality. [11]



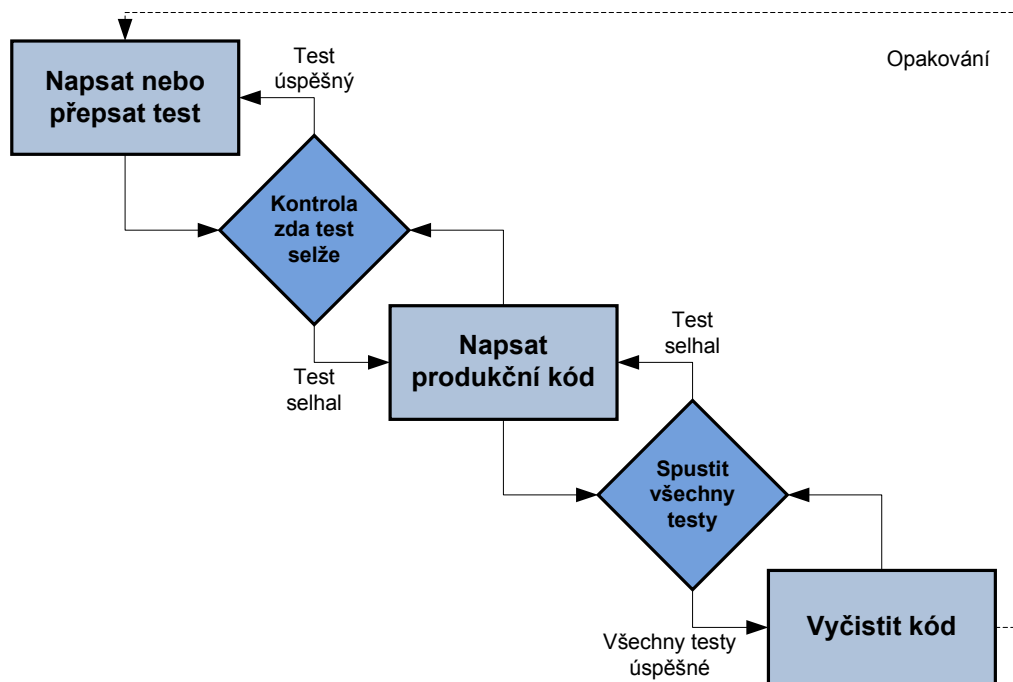
Obr. 6. Plánovací smyčka se zpětnou vazbou

Test-Driven Development (TDD)

Tato metodika obrací klasický vývojový proces a říká, že nejprve se pro danou funkci napíše test a až následně se naprogramuje. Programování funkcionality trvá do té doby, než je schopna projít testem. Následně proběhne optimalizace kódu s odstraněním duplicit a funkce se ještě jednou ověří. Postupuje se ve velmi krátkých iteracích, které mohou trvat pouze i několik minut.

V rámci této metodiky se jedná o testování tzv. jednotek. Obvykle se jedná o malý kousek funkcionality. Testy musí být tedy jednoduché zaměřené pouze na určitou funkci, rychlé, protože je pouštěno více testů naráz, mezi sebou nezávislé a testovací kód musí být stejně tak kvalitní jako produkční kód.

Testovací kód si píše vývojáři sami, protože ti dokážou nejlépe definovat, jaká funkcionality se má testovat. A právě ve fázi psaní testovacích scénářů je skryta fáze návrhu, která není explicitně definována. Pokud má totiž vývojář nejdříve napsat test, musí si dobře promyslet, jak bude daná část kódu fungovat a právě v této fázi je schován vlastní návrh. [16]



Obr. 7. Test Driven Development

1.3 Rizika

V průběhu vývoje software se často vyskytují různé problémy. Systémy vytvářené dle požadavků zákazníka budou na začátku vždy trpět dětskými nemocemi. S touto skutečností je nutné počítat a neopomíjet ji. Podcenění rizik může způsobit, že finální produkt nebude obsahovat požadovanou hodnotu vzhledem k vynaloženým prostředkům. V dnešní době, kdy je kladen důraz na snižování nákladů, je nutné vnímat rizika se zvýšenou pozorností.

Jednou z nejvýznamnějších částí projektu je definování požadavků na software, především je důležité odlišit klíčové požadavky od těch méně významných, nebo dokonce zcela nepodstatných. Podobně jako v ekonomii je možné aplikovat Paretovo pravidlo 80/20, dvacet procent důležitých požadavků pokrývá osmdesát procent skutečných potřeb zadavatele. Těmto významným požadavkům by měli být přiřazeny odpovídající priority. Pokud se projekt dostane do časových nebo rozpočtových problémů, je nutné se zaměřit hlavně na prioritní požadavky a nezabývat se tím, co je ve skutečnosti méně důležité. Jinak hrozí zvýšení nákladů nebo snížení kvality. [17]

Složným prvkem ve vývoji software je převedení požadavků z reálného světa do oblasti informačních technologií. Obvyklou chybou je otrocká migrace procesů a postupů do vytvářeného informačního systému. Tento přístup obvykle vede ke zbytečné složitosti

systemu. Proces v reálném světě je obvykle možné ad-hoc modifikovat konkrétním pracovníkem. V IT oblasti ale taková řešení nejsou vůbec možná. Je tedy nutné projít veškeré procesy a rozhodnout, které ponechat, které od základu změnit a které do informačního systému vůbec nepřevádět. V případě nepřesného přenesení procesů do systémů může dojít k tomu, že složitost a byrokracie systému překročí únosnou mez a ten přinese více nových problémů, než aby ty stávající řešil.

Další důležitou oblastí při tvorbě software je výběr vhodné platformy a technologie. Zde dodavatel samozřejmě propaguje svoje řešení, které může, ale nemusí být v souladu s potřebami zákazníka. Výběr vhodné technologie je důležité pro budoucí rozšíření a kompatibilitu. Stejně tak přechod ke konkurenci by mohl zvýšit provozní náklady a zdražit případné budoucí požadavky.

Důležitou fází ve vývoji software je testování finální aplikace. Nejde pouze o klasické vyzkoušení, ale nalezení odpovědí na otázky typu „co se stane, když“. Během testování je nutné se zaměřit na funkční testy, bezpečnost, zátěžové testy, skryté kritické chyby, které mohou vést k poškození provozních dat atd.

Výše zmíněná rizika můžeme definovat jako šestici nejčastějších problémů softwarových projektů. [18]

Zpoždění

Jedná se o nejčastější problém softwarových projektů, což je zpoždění proti původnímu harmonogramu. Může se jednat o projekty, na jejichž dokončení jsou navázány další činnosti a jejich zpoždění je tedy nepřijatelné. Ve většině případů bývá čas velmi důležitý, mnohdy je ale lepší datum dokončení odložit a dotáhnout projekt do konce. Na zpoždění si nikdo časem nevzpomene, ale k nekvalitnímu software se už nikdo nikdy nevrátí.

Vysoká chybovost

Dalším významným problémem softwarových projektů je vysoká chybovost. Chyby se v průběhu vývoje vyskytují vždy, je nutné je ale najít a opravit. Odladění chyb v téměř hotovém programu bývá časově náročné. Přestože se podaří chyby odstranit, může dojít k zpoždění projektového plánu.

Vzhledem k těmto skutečnostem je kladen důraz na programátory, aby psali čitelný kód a zároveň byl produkt dostatečně otestovaný. Na druhou stranu malé množství drobných chyb je ze strany zákazníka akceptovatelné pokud mu ušetří velké množství práce. Dle

prováděných výzkumů se dokonce zjistilo, že lidé dávají přednost aplikaci se zajímavými vlastnostmi, i když obsahuje drobné chyby.

Nesplnění požadované funkčnosti

Hlavní příčinou tohoto problému je chyba v komunikaci, která způsobí, že program nedělá přesně to, co od něj bylo očekáváno. Špatná vzájemná komunikace může vzniknout mezi zadavatelem a vývojovým týmem, nebo také uvnitř týmu.

Nedostatečná výkonnost

Další problém, s kterým je možné se setkat, je nedostatečný výkon systému. V některých případech pomůže výkonnější hardware, ale jindy má aplikace výkonnostní limity, které není možné zvýšením hardwaru odstranit. Takové problémy jsou způsobeny špatnou organizací dat nebo vzájemných blokováním programových vláken.

Složité uživatelské rozhraní

Lidé, kteří vyvíjí programy, mají k počítačům blíže než budoucí uživatelé těchto systémů, což může vést k tomu, že uživatelské rozhraní se stane příliš složité, nepřehledné nebo nepochopitelné. Uživatel se tak použití systému vyhýbá, jak jen může.

Obtížná údržba programu

Po akceptování a úspěšném nasazení aplikace nastává období údržby a nezbytných úprav pro zajištění bezproblémového chodu. Pokud v průběhu vývoje byla zanedbána nebo zcela chybí dokumentace, programátoři tvořili těžko čitelný kód, každá změna v takovém případě bude nesmírně náročná a zároveň finančně nákladná.

1.4 Současná softwarová krize

Termín softwarová krize se poprvé začal používat v 60. letech 20. století. Přibližně pouze 30 % vyvíjených aplikací bylo vytvořeno bez problémů, zbývajících 70 % narazilo v průběhu vývoje na nějaké problémy nebo selhaly úplně. Charakteristickými znaky bylo neúnosné prodlužování a prodražování projektů, nízká kvalita programu, nesnadnost či nemožnost údržby a inovace, špatná produktivita práce programátorů, neefektivita vývoje, nejistota výsledku atd. Příčinou krize byl vždy nesoulad mezi složitostí vytvářeného produktu a nezkušeností softwarové profese. [19]

Jednotlivé příčiny softwarové krize:

- Špatná komunikace mezi zainteresovanými osobami do projektu a mezi vývojáři a zákazníkem.
- Nedostatečné plánování celého projektu. Všichni si myslí, že to nějak stihnou a nikdo neví, v jakém stavu se projekt opravdu nachází.
- Nesprávné odhady trvání vývoje, nákladů, rozsahu.
- Nízká produktivita práce. Programátoři se nezabývají klíčovými částmi projektu, místo toho řeší malichernosti.
- Podcenění hrozeb a rizik. Málo jsou sledovány hrozby a místo snadného předcházení přerůstají ve velké problémy.
- Nezládnuté technologie. Falešná představa, že po zavedení nové technologie se potíže samy vyřeší.

Při pohledu na předchozí body je možné konstatovat, že softwarová krize neskončila, ale stále trvá.

Další často opomíjená příčina je narůstající výkon hardware. Počítače začaly vykonávat několik úloh současně a rychleji. Do doby, dokud počítače nepřesáhly určitý rozměr, bylo možné se spolehnout na geniální členy týmu. Po této době jakkoli geniální člen může obsáhnout je určitou část problému. Řešení velkých projektů není možné nechat pouze na nich, proto je velké problémy nutné rozdělit na menší.

V současné době působí v oblasti informačních technologií krize, která ale vznikla z jiného důvodu. Světová ekonomická krize se z oblasti financí přelila na trh informačních technologií. Zpřísnění úvěrování bank zhoršilo finanční situaci podniků a domácností, které omezily výdaje plynoucí do technologií. Firmy šetří na výdajích do IT, lidé méně kupují nové mobilní telefony a počítače. [20]

Krize však nezasáhla všechny oblasti informačních technologií stejně. Softwarové firmy postihla krize méně než výrobce a prodejce hardware. Přestože se vývojových center úspory nedotkly v takové míře, došlo v některých společnostech k ozdravné kúře formou restrukturalizace a některé firmy skončily úplně, především menšího rozsahu s lokálním kapitálem.

1.5 Odhady softwarových projektů

Mnoho lidí si pod pojmem odhad pracnosti nebo odhadování představí „černou magii“. Při letném pohledu je možné nabýt dojmu, že se jedná o velmi subjektivní proces, protože jednomu může zabrat den provést nějaký úkol, jinému to zabere pouze několik hodin. Výsledkem toho je, že pokud jsou lidé tázáni, jako dlouho jim bude trvat vykonání konkrétního úkolu, tak jejich odpovědi jsou diametrálně odlišné. Skutečné množství vynaloženého času, je tedy zjištěno až po provedení úkolu. Jakýkoliv odhad, který se nepřiblížil skutečnému času, je proto považován za nepřesný.

Lidem, kteří nikdy neprováděli odhad prostřednictvím strukturovaných metod, se může zdát, že se nejedná o nic víc, než je pouhé předpovídání budoucnosti. Tento dojem je ještě více posílen, pokud odhad je nepřesný a projekt není dodán včas. Ale formální odhadní proces, který pomůže celému týmu najít shodu ve výsledném odhadu, může zlepšit přesnost těchto odhadů a je mnohem více pravděpodobné, že cíl projektu bude splněn včas.

Nedostatečný nebo v horším případě žádný odhad pracnosti a rozsahu jsou jedním z hlavních důvodů selhání projektu. Odhad je kritickým faktorem při definování nákladů, časového harmonogramu a vynaloženého úsilí. Důsledkem nepřesného odhadu je, že dojde k překročení počátečního rozpočtu a projekt je dodán se zpožděním, což ihned na začátku spolupráce podkopává důvěru ve vytvářený produkt. Určení pracnosti je komplikovaná činnost, jejíž výsledek musí být neustále aktualizován v průběhu životního cyklu. Měřítkem velikosti a složitosti projektu mohou být například řádky zdrojového kódu, požadované funkce, požadované vlastnosti atd. Pracnost je funkcí velikosti projektu, což velmi zvyšuje dopad chyb v návrhu a skrytých vad, které nakonec vyústí v problémy s kvalitou, překročení nákladů a skluzu v časovém harmonogramu. Pracnost musí být neustále měřena, sledována a kontrolována. Dalším faktorem, který často způsobuje nepřesnost v odhadu je specifikace požadavků, které musí být jasně stanoveny a svědomitě kontrolovány.

Jakákoliv lidská činnost, která postrádá kontrolu, se v průběhu času zhoršuje. Toto vyžaduje konstantní pozornost a disciplinovanost, aby docházelo k přírůstkům v softwaru a vývojový proces se posouval dále. Pokud není prováděno žádné měření, neexistuje žádný způsob, kterým zjistit, zda se projekt vyvíjí správným směrem nebo zda dochází k jeho zlepšování. Tato hodnocení musí být aktualizována, aby odrážela současný stav programu a ten mohl být porovnán s projektovým plánem. Po vyhodnocení současného stavu a

zjištění, že projekt se nachází v potížích, je možné provést smysluplná, efektivní nápravná opatření.

Dle výše uvedeného je možné konstatovat, že dobrý způsob měření a reportování stavu systému, umožňuje včasnou detekci problémů a poskytuje kvantitativní vysvětlení kritických otázek rozvoje. Používání těchto metrik dává schopnost identifikovat, řešit a omezit rizika problémů ještě před tím, než vyplují na povrch. Sledování stavu systému nesmí být cílem samo o sobě, vždy musí být integrováno do celkového životního cyklu softwaru. Aby byla metrika efektivní, nestačí, aby byly informace pouze sbírány, musí být používány.

2 REŠERŠE EXISTUJÍCÍCH METOD

2.1 Analogie

Odhad pracnosti software založený na analogii vychází z principu, že dosavadní získané hodnoty v rámci společnosti z dřívějších a podobných projektů jsou lepší indikátory a jsou schopny predikovat budoucí vykonávané práce s mnohem větší přesností, než kdyby byly odhady tvořeny nanovo. Tato skutečnost také usnadňuje přenos organizačních zkušeností na nové projekty.

Pro použití této metody je nutné, aby organizace splňovala následující prerekvizity.

Společnost již musela realizovat nějaké projekty v minulosti a pečlivě musí být vedeny záznamy těchto projektů. Pokud došlo k neúspěšnému dokončení projektu, musí být zanalyzovány příčiny, na základě kterých se tak stalo. Je třeba dbát na to, aby na základě chybné analýzy nedocházelo k negativnímu ovlivnění budoucích projektů. Firma by měla mít dobře organizovanou znalostní bázi, v které je možné vyhledat podobné projekty z minulosti a použít platná projektová data. Lidé provádějící odhad pracnosti by měli mít přístup do znalostní báze, měli by v ní umět velmi dobře vyhledávat, ověřovat vyhledaná data a vhodně je aplikovat na současný projekt. [21]

Jakmile jsou výše uvedené prerekvizity splněny, je možné tuto metodu velmi efektivně aplikovat v dané organizaci.

Výběr podobných projektů

Jedná se o velmi důležitý krok v rámci této metody, jehož výsledkem je seznam podobných projektů. Prvotním kritériem samozřejmě je, o jaký typ projektů se jedná.

1. Projekt s plným životním vývojovým cyklem – klasický vývojový projekt.
2. Implementační projekt – ERP, CRM, logistika apod.
3. Konverzní projekt – přizpůsobení aplikace nově vzniklým okolnostem, např. přechod na měnu euro, Y2K projekt apod.
4. Změna platformy – přesun z jedné softwarové platformy na jinou, jako je např. přesun ze staré verze Unixu na novou.
5. Migrační projekt – migrace aplikace nebo dat z jedné hardwarové platformy na jinou.

Dalším velmi důležitým hlediskem je výběr skupiny minulých projektů. Při výběru by měly být porovnávány následující parametry.

1. Oblast aplikace – je jednou z nejdůležitějších vlastností, které jsou srovnávány. Vždy by měly být srovnávány aplikace působící ve stejné oblasti.
2. Velikost organizace budoucího klienta – rozsah funkcionalit bude rozdílný pro dvě různě velké společnosti, i když působí ve stejné doménové oblasti. Měli by se tedy vybírat minulé projekty velikostně srovnatelné se stávajícím projektem.
3. Počet poboček budoucího klienta – funkcionality budou rozdílné pro organizaci působící na jednom místě a pro společnost v které by měl být systém implementován na více místech.
4. Vlastnosti modulů v aplikaci – projekt z minulosti musí zahrnovat většinu modulů současného projektu. Je možné extrapolovat pro jeden nebo dva moduly, ale v žádném případě pro většinu, když nebyla obsažena v minulém projektu.

Dle použité platformy je nutné zohlednit následující kritéria.

1. Počet vrstev aplikace – dvouvrstvá architektura se liší od třívrstvé architektury.
2. Backend – v současné době jsou téměř všechny aplikace postaveny na relačním databázovém modelu. Pokud se v obou projektech zpracovávají a ukládají data do relační databáze, můžeme je považovat za rovnocenné.
3. Webový server – rozdílné webové servery znamenají rozdílné množství práce. Jedná se o důležité kritérium především u webových aplikací.
4. Middleware – použití rozdílného middleware má dopad na množství vynaložené práce.
5. Programovací jazyk – pracnost projektu je také ovlivněna použitým programovacím jazykem, v kterém je aplikace vyvíjena. Použití rozdílného programovacího jazyka v minulých projektech, než v aktuálním projektu musí být zohledněno v odhadu pracnosti.
6. Vývojové prostředí – použité nástroje pro editaci, debugging a kompilování programu mají také podstatný vliv na produktivitu programátorů. Z toho důvodu je důležité vybrat projekty, které byly vytvořeny ve stejném nebo podobném vývojovém prostředí nebo frameworku.

Seznam minulých projektů

Po zohlednění všech kritérií je možné sestavit seznam podobných projektů, které spadají do stejné aplikační oblasti.

Kritéria	Současný projekt	Projekt 1	Projekt 2	Projekt 3	Projekt n
Typ projektu	Vývojový	Vývojový	Vývojový	Vývojový	Vývojový
Oblast aplikace	CRM	CRM	CRM	CRM	CRM
Velikost organizace	Střední	Velká	Velká	Střední	Malá
Počet poboček	2	6	8	3	1
Moduly	Nákup, Sklad, Logistika	Nákup, Sklad, Logistika, Doprava, Zásoby	Nákup, Sklad, Logistika, Doprava, Zásoby	Nákup, Sklad, Logistika, Zásoby	Nákup, Sklad, Zásoby
Aplikační vrstvy	3	4	4	2	1
Backend	RDBMS	RDBMS	RDBMS	RDBMS	Soubory
Webový server	IIS	Web Sphere	Web Sphere	-	-
Middleware	Nil	Web Logic	Web Logic	-	-
Programovací jazyk	J2EE	Dot Net 03	Dot Net 05	VB 6	COBOL
Vývojové prostředí	J2EE	Dot Net 03	Dot Net 05	VB 6	COBOL

Tab. 1. Seznam minulých projektů

Výběr projektů uvedených v tabulce (Tab. 1) může být případně prováděn automaticky, stejně tak jako počet může být zúžen na dva nebo tři projekty. Člověk provádějící odhad by měl vybrat jeden projekt, který se stane základem pro současný projekt. Tento krok by neměl být zautomatizován. Přítomnost lidského rozhodnutí je důležitá k výběru projektu, který:

1. Je nejvíce podobný současnému projektu, dle výše uvedených parametrů.
2. Umožňuje nejmenší možné přiblížení k obdržení nového odhadu.

Po výběru projektu, jsou provedeny následující kroky, sloužící k přípravě nového odhadu.

1. Vytvoření srovnávacího přehledu všech parametrů mezi minulým a současným projektem.
2. Přizpůsobení pro rozdílné parametry, pokud je to nutné. Existují dvě možnosti:
 - a. Je k dispozici detailní odhad minulého projektu – v tomto případě je možné vzít starý odhad a upravit ho dle nového projektu.
 - b. Je k dispozici pouze hrubý odhad minulého projektu – v tomto případě, je nutné přizpůsobit hrubý odhad.
3. Dokončení nového odhadu a umožnit provedení zpětné vazby, pokud je nutná.

4. Uspořádat manažerské zhodnocení odhadu a případně zapracovat připomínky.
5. Vhodnou formou prezentovat odhad zadavateli.

Pokud je dostupný detailní odhad za vybraný minulý projekt, je nutné provést následující body pro vytvoření nového odhadu.

1. Uložit starý odhad jako nový odhad.
2. Provéřit každý detail starého odhadu a
 - a. Pokud je detail identický se současným projektem, tak jej ponechat.
 - b. Pokud je detail naprosto mimo rozsah současného projektu, tak jej vymazat.
 - c. Pokud je detail částečně aplikovatelný pro současný projekt, přizpůsobit jej tak, že bude reflektovat aktuální projekt.
 - d. Provést výše uvedené kroky pro každý detail a formulovat nový odhad
3. Přezkoumat, zda neexistuje nějaký detail, který se vztahuje k aktuálnímu projektu, ale zcela chybí v odhadu na základě starého projektu. Pokud ano, zahrnout jej do odhadu, aby byl odhad kompletní.
4. Následně je potřeba prověřit aspekty prostředí, jako je vývojová platforma, programovací jazyk, počet vrstev aplikace, middleware atd. a udělat vhodné přizpůsobení novému odhadu.

Nyní je odhad připraven k přezkoumání a pro zpětnou vazbu. Je uspořádáno manažerské hodnocení odhadu, na základě kterého je vydáno souhlasné stanovisko.

Naopak pokud je dostupný pouze hrubý odhad minulého projektu, je nutné provést následující body pro vytvoření nového odhadu.

1. Provéřit každý detail starého odhadu.
2. Přiřadit váhu každému parametru
 - a. Váha nula (0) znamená, že tento parametr není vůbec aplikovatelný pro současný projekt.
 - b. Váha jedna (1) znamená, že tento parametr je identický se současným projektem.
 - c. Váha větší než nula a menší než jedna signalizuje, že tento parametr je použitelný, ale ne 100%.
 - d. Váha větší než jedna znamená, že tento parametr je aplikovatelný a jeho použitelnost je vyšší než 100 %.

3. Následně pro obdržení tzv. Compose Weight Factor (CWF) je suma všech vah vydělena počtem všech parametrů.
4. Vynásobením hrubého odhadu minulého projektu s CWF se získá nový odhad pro aktuální projekt.

Nyní je připraven odhad pro kontrolu. Na základně zpětné vazby je možné zahrnout ještě nějaké odhady. Následně dojde k manažerskému zhodnocení a ke schválení odhadu. Níže uvedená tabulka zachycuje samotnou proceduru.

Předpokládejme, že vybraný minulý projekt představuje.

1. Jeho rozsah je 500 funkčních bodů.
2. Požadavky na vývoj o rozsahu 5000 člověkohodin.

Kritéria	Současný projekt	Vybraný projekt	Váhy
Typ projektu	Vývojový	Vývojový	1
Oblast aplikace	CRM	CRM	1
Velikost organizace	Střední	Velká	0,4
Počet poboček	2	6	0,4
Moduly	Nákup, Sklad, Logistika	Nákup, Sklad, Logistika, Doprava, Zásoby	0,4
Aplikační vrstvy	3	4	7
Backend	RDBMS	RDBMS	1
Webový server	IIS	Web Sphere	0,8
Middleware	Nil	Web Logic	0
Programovací jazyk	J2EE	Dot Net 03	1
Vývojové prostředí	J2EE	Dot Net 03	1
Composite Weight Factor (CWF)			1,27

Tab. 2. Současný a vybraný projekt s hodnotou CWF

Na základě tabulky (Tab. 2) je CWF 1,27. Nyní jsou hodnoty minulého projektu vynásobeny tímto parametrem, aby bylo dosaženo aktuálního odhadu.

1. *Rozsah* = 500 * 1,27 což je 635 funkčních bodů.
2. *Požadavky na vývoj* = 5000 * 1,27 což je 6350 člověkohodin.

Výhody použití analogie pro odhady pracnosti

Metoda je založena na hodnotách dosažených společnostmi v dřívějších projektech a je více spolehlivá než ostatní odhadní metody. Jednoduše aplikovatelná s velmi rychlým dosažením výsledku. Nově založené společnosti si můžou koupit zdrojová data od

organizací jako je např. ISBSG (International Software Benchmarking Standards Group). Přestože nejsou tyto odhady prováděny uvnitř organizace, poskytují spousty odhadů a stávají se startovním bodem pro nové organizace. Tato metoda umožňuje použití odborných znalostí a zkušeností z vnitřku organizace, jako žádná jiná metoda.

Nevýhody použití analogie pro odhady pracnosti

Seznam dřívějších projektů a výběr konečného projektu vyžaduje pečlivé vedení záznamů a nástroje pro zpracování těchto záznamů, jakákoliv laxnost v tomto kroku může mít vážné následky pro budoucí odhad. Organizace musí vést dobře navrženou znalostní bázi. Může nastat situace, že aktuální projekt nemá žádný odpovídající projekt v rámci společnosti. Složitěji implementovatelné pro nové organizace.

2.2 Expertní odhady

Expertní odhady představují konzultaci s jedním nebo více odborníky, což zahrnuje jejich zkušenosti a znalosti k právě odhadovanému projektu. Často jsou používány jako doplňková metoda k aritmetickým modelům. Expertní odhad je proces, který je prováděn odborníkem na základě jeho zkušeností s minulými podobnými projekty. Odborník poskytuje informace založené na jeho znalostech a zkušenostech. Expertní odhady mohou být relativně přesné, pokud má odborník důležité nedávné zkušenosti s vývojem a plánováním podobného softwarového projektu. Za daných okolností je tato metoda schopna poskytnout docela věrohodné výsledky. Vzhledem k tomu, že lidské zkušenosti a názory jsou nedílnou součástí expertních odhadů, nelze je aplikovat samostatně. Lidský faktor zde hraje vždy určitou roli. Za účelem eliminace lidského faktoru bývá tato metoda často kombinována s jinou metodou pro odhadování pracnosti. [22]

Samotný proces odhadování probíhá následovně. Vybere se odborník, který se snaží odhadnout na základě seznamu funkcí a vlastností projektu, jak nákladný bude daný projekt. Většinu času se tyto odhady velmi liší od skutečnosti, ale při provedení několika elementárních kroků může být docíleno přesnějšího odhadu.

Prvním krokem, na který by měl být kladen důraz, je správný výběr experta. Osoba by měla splňovat následující předpoklady. Expert by měl být odborníkem v dané oblasti, což představuje, že by měl rozumět obchodní a technické stránce problému. Expert by měl být také odborníkem v technické oblasti a měl by tedy do hloubky rozumět systému, který

zahrnuje daný projekt. V minulosti by se měl podílet na několika projektech z této oblasti. Dále musí znát nástroje a technologie, které budou použity při tvorbě projektu.

Dalším krokem je, že pro každou vlastnost nebo oblast systému by mělo být vytvořeno více scénářů. Nejlepší případ představuje, pokud vše půjde dle předpokladů, jak rychle by mohla být vytvořena funkce nebo vlastnost systému. Nejhorší případ symbolizuje, pokud se vyskytnou všechny možné komplikace, kterých jsou si členové týmu vědomi, jak dlouho zabere vytvoření funkce nebo vlastnosti systému. Nejpravděpodobnější případ je navržen na základě dvou předchozích bodů a zkušeností z minulosti a zachycuje, jak dlouho s největší pravděpodobností bude trvat vytvoření funkce nebo systému.

Na základě těchto tří výstupů, lze například při použití metody Program Evaluation Review Technique (PERT) vypočítat očekávaný odhad. Metoda PERT přiřadí každému ze tří hodnot jinou váhu a očekávaný odhad je možné vypočítat na základě následujícího vzorce.

$$\text{Ocekavany odhad} = (\text{nejlepsi pripad} + (3 * \text{nejpravdepodobnejsi pripad}) + (2 * \text{nejhorsi pripad}))/6$$

Sečtením očekávaného časového odhadu za každou část projektu (implementace, testování atd.) je stanoven celkový časový odhad za celý projekt.

Při použití této metody je odhad závislý pouze na jedné osobě. Velmi často se stává, že nelze najít osobu, která by měla dostatečnou kvalifikaci, která je nutná k provedení odhadu. Výsledek této techniky může být zlepšen tak, že odhad bude nezávisle provádět více osob a každá přijde s jeho vlastním odhadem. Následně se všichni experti setkají a dohromady diskutují nad provedenými odhady. Debata se týká důvodů, proč mezi jejich odhady došlo k rozdílům a měly by společně najít určitou shodu v jejich tvrzeních. Čísla ze vzájemné shody, jsou aplikována do výpočetního vzorce uvedeného výše.

Expertní odhad a odhad provedený na základě názoru více expertů je vhodné použít v rané fázi projektu, kdy požadované vlastnosti jsou známy, ale ještě není přesně jasné, jakým způsobem budou řešeny.

2.3 Parametric Estimating

Metoda parametrického odhadu spočívá ve sběru odpovídajících historických dat, obvykle na určitém stupni detailu, vztahujících se k produktu, který bude odhadován na základě matematických metod. Vzhledem k tomu, že parametrická metoda ve většině případů sbírá data na vyšším stupni rozlišení, méně detailní informace nejsou tolik potřeba jako v jiných metodách. Základním kamenem této metody je vztah odhadovaných nákladů (cost estimating relationship - CER). [23]

CER vyjadřuje vztah nákladů jako závislou proměnou k jedné nebo více nezávislých proměnných. CER definuje náklady jako funkci jednoho nebo více technických parametrů, což mohou být fyzické nebo funkční vlastnosti. Vztah je vyjádřen na základě matematické rovnice, a jakmile je tento vztah stanoven, tak je relativně jednoduché tuto metodu použít. Vztah odhadovaných nákladů může porovnávat finanční k finančním nákladům nebo finanční k nefinančním nákladům. Například finanční k finančním vyjadřuje vztah hodin potřebných na vývoj (nezávislá proměnná) k odhadovaným hodinám na zajištění kvality (závislá proměnná). Tato metoda umožňuje odhadci provést rychlý odhad bez detailního seznámení se všemi informacemi. Při vyjadřování vztahu CER, by měl být kladen důraz na jednotlivé aktivity objektu. Aktivitami jsou myšleny vlastnosti produktu, které mají významný dopad na jeho náklady. Vlastnosti mohou být fyzické, vizuální, funkční nebo jakákoliv jiná identifikovatelná vlastnost produktu.

2.3.1 Cost estimating relationship

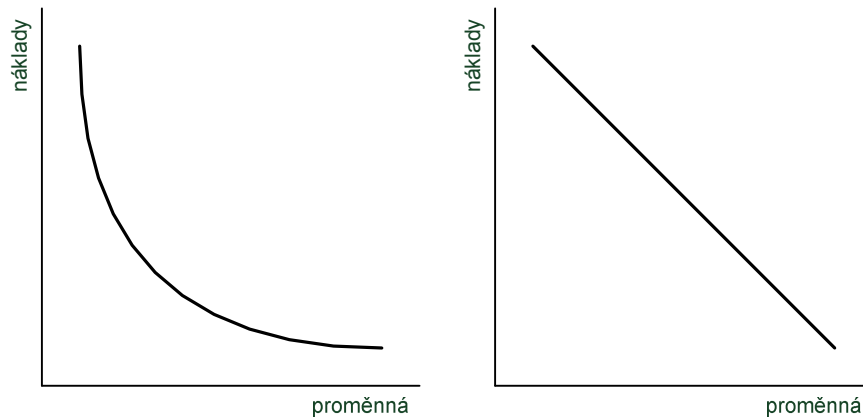
Při zpracování dat z předchozích projektů musí být kladen důraz na dva základní body. Vztah mezi jednotlivými proměnnými musí dávat smysl a musí mít prediktivní hodnotu.

Přestože není možné vyřešit všechny nejasnosti týkající se dostupných informací, osoby provádějící odhad by si měly být dostatečně jisti tím, že vztah odhadovaných parametrů nevykazuje žádné systémové nedostatky. Odůvodněnost použití jednotlivých proměnných může být testována rozmanitými způsoby – pravidelnými kontrolami, prostřednictvím grafů, komplikovanými technikami, které zahrnují testování každé proměnné napříč oborem možných hodnot.

Odhad vztahu mezi náklady může mít omezený rozsah platnosti. Při provádění specializovaných odhadů v rámci úzkého oboru hodnot, nastává situace, že mimo rozsah

těchto hodnot je výsledkem nedostatečná prediktivní hodnota. Nedostatečná prediktivní hodnota vede ve většině případů k chybnému odhadu.

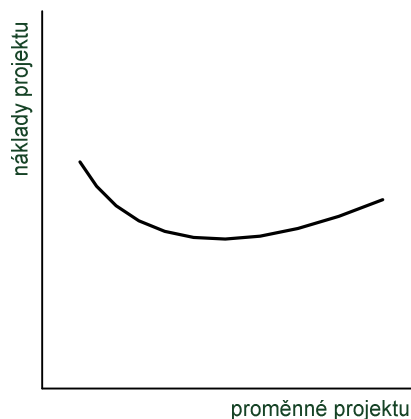
Běžný způsob, který se používá, pro zkoumání důsledku vztahu hodnot mimo daný rozsah je zachycení závislosti křivkou v grafu. Teorie říká, že pokud dojde ke zvýšení hodnot proměnné, hodnota závislých nákladů se zvýší nebo sníží předvídatelným způsobem. Křivka vyznačující závislost graficky může mít lineární nebo logaritmický průběh.



Obr. 8. Lineární a logaritmický průběh závislosti nákladů

Sklon křivky na obrázku (Obr. 8) je poměrně strmý. Pokud by došlo k rozšíření průběhu křivky směrem doprava, dalo by se očekávat její vyrovnání. Nakonec může být křivka horizontální zcela úplně, ale je nepravděpodobné, že by měla pozitivní růst

Na obrázku (Obr. 9) jsou zachyceny celkové náklady projektu ve vztahu k rozsahu projektu.



Obr. 9. Náklady projektu

Dva rozdíly jsou na první pohled ihned viditelné. První část křivky je neobvykle prudká. Druhá část křivky v její poslední části je pozitivní. V některých případech můžou nastávat

problémy se zpracováním projektu v takovém rozsahu, což způsobuje její pozitivní sklon. Při analýze dílčích částí projektu tato situace nenastává, ale při aplikaci na celý projekt, se tato situace může objevit, pokud se bude chtít využít úspor z rozsahu.

2.4 Wideband Delphi

Základy metody Wideband Delphi byly vytvořeny v roce 1940 v americké neziskové organizaci RAND (Research ANd Development), která se zabývá předpovídáním a odhadováním na základě výzkumů a analýz. V těchto letech byla specializována pouze na vojenské účely. V 70. letech byla rozšířena, z čehož plyne doplnění slova „wideband“ do jejího názvu. Od této doby byla aplikována v mnoha průmyslových odvětvích, zahrnující statistický sběr dat až po stanovení prodejních a marketingových prognóz. Ukázalo se, že je to velmi efektivní metoda pro tvorbu odhadů, proto začala být aplikována i v softwarových projektech.

Wideband Delphi je velmi užitečná především pro projektové manažery, protože jejím výstupem je několik podstatných prvků projektového plánu. Nejvýznamnějším výstupem je skupina provedených odhadů, na jejichž základě je vytvořen plán projektu. Vedle toho je definována tzv. WBS (Work Breakdown Structure), což je základním prvkem plánu. Dalšími výstupy, které jsou generovány v průběhu této metody, je soubor jednotlivých předpokladů, na základě kterých jsou vytvořeny vize projektu.

Týmová diskuze je prováděna mezi dvěma sezeními, kde první iniciální se nazývá „kickoff meeting“ a v dalším už dochází ke tvorbě odhadů. Diskuze slouží k stanovení priorit, předpokladů a úkolů, které nebyly doposud definovány. Tým je také mnohem lépe seznámen s tím, co vše je součástí projektu.

Wideband Delphi funguje především díky tomu, že vyžaduje názory všech členů týmu, čehož důsledkem je, že jednotliví členové se korigují navzájem a lépe se vyhnou chybám a mylným odhadům. Přesnost odhadů projektů je založena na zkušenosti osoby provádějící odhad, avšak nejčastější problém těchto odhadů je ten, že osoba provádějící odhad nedostatečně rozumí tomu, co je předmětem odhadu. Může se jednat o zkušeného projektového manažera, ale pokud dostatečně neprozkoumá všechny předpoklady týkající se odhadu, tak jeho výsledek bude nepřesný. Cílem Delphi procesu je tedy diskuze formou předpokladů a prognóz a stanovení vzájemné shody mezi všemi členy týmu. [24]

2.4.1 Delphi proces

Projektový manažer vybere moderátora a odhadní tým o počtu tří až sedmi osob. Na první „kickoff meeting“ schůzce je definována WBS a projednávány předpoklady projektu. Po schůzce každý z členů vytvoří odhad pro úkoly, které mu byly svěřeny. Na druhém sezení jsou revidovány odhady z pohledu celé skupiny a snaží se najít vzájemnou shodu. Na základě této schůzky projektový manažer vytvoří projektový plán, s nímž následně seznámí všechny členy týmu.

2.4.2 Výběr týmu

Aby byl odhad, co nejpřesnější musí být vybrán dostatečně kvalifikovaný tým. Každý člen týmu by měl provádět odhady jednotlivých úkolů čestně a měl být nápomocný při spolupráci se zbytkem týmu. Projektový manažer musí zajistit volný tok informací a vybrat takovou skupinu lidí, kteří se budou vzájemně doplňovat a respektovat. Účastníci by měli dostatečně znát potřeby a filosofii společnosti a v minulosti zpracovávané projekty podobného typu.

Moderátor by měl být obeznámen s Delphi metodou, i když není to nezbytně nutné. Projektoví manažeři mají obvykle pokušení plnit roli moderátora, což není vhodné, protože projektový manažer by měl být součástí odhadního týmu. Projektový manažer hraje důležitou roli při definování předpokladů projektu, protože pouze on má většinou přehled o prioritách projektu, které na počátku nemusí všichni jednoznačně identifikovat.

Rolí moderátora je naslouchat diskuzi, klást otevřené otázky, vyzívat tým k řešení problému a zajistit, aby všichni přispívali do diskuze. Moderátor se může také zúčastnit vlastního procesu odhadování, ale nesmí dojít ke zkreslení odhadů prováděných ostatními členy týmu. Dobře vybraný tým umožňuje moderátorovi zastat neutrální postoj k celé diskuzi.

Projektový manažer by měl vybrat takové členy týmu, kteří budou reprezentovat všechny funkce v budoucím projektu, např. manažeři, analytici, vývojáři, testeři atd. A co je nejdůležitější, každý z členů týmu by měl mít zájem na tvorbě plánu projektu, aby se s ním v budoucnu mohl bez problémů ztotožnit. Delphi proces slouží také k výběru členů technické podpory odhadu softwarového projektu, což každému dává dobrý pocit, že výsledný odhad je založen i na jeho dílčích odhadech.

Nakonec by měl být také vybrán jeden nebo více pozorovatelů, což mohou být investoři, manažeři nebo uživatelé. Pozorovatelé nechápou všechny technické podklady týkající se odhadu, což jim umožňuje přemýšlet o projektu z jiného úhlu pohledu. Účast pozorovatelů také slouží k upevnění vztahů mezi členy týmu a netechnickými lidmi v organizaci. Přestože se přihlížející neúčastní přímo číselných odhadů, stejně jako u ostatních členů týmů nabývají pocit, že byly nedílnou součástí výsledného odhadu. Když se netechničtí členové podílí na definování předpokladů projektu a vidí, jak tým dospěje k jednotlivým odhadům, udělají si představu, jak techničtí pracovníci vytvářejí takový odhad a jak přesně bude software vyvíjen.

2.4.3 Kickoff meeting

Cílem kickoff meeting je připravit se na následující schůzku, která se bude týkat vlastních odhadů. Každý z členů dostane před začátkem sezení podkladové dokumenty, aby se seznámil s projektem, který bude odhadovat. Předpokládá se, že členové týmu si přečtou dokumenty před začátkem schůzky.

Moderátor vede jednání, na kterém vysvětlí wideband delphi metodu nových odhadcům. Pokud někteří z členů týmu doposud nečetli podkladové dokumenty, moderátor zkráceně představí co je jejich obsahem. Moderátor s členy zhodnotí a zreviduje vlastní cíl odhadování a zkontroluje, že každý z členů je dostatečně seznámen s problematikou, aby mohl následně přispívat do diskuze. Členové týmu probírají produkt, který se bude v budoucnu vyvíjet, a navrhují předpoklady. Výsledkem sezení je soubor 10-20 významných úkolů. Tyto úkoly reprezentují nejvyšší úroveň WBS. Další detaily mohou být odvozeny později z těchto úkolů. Soubor úkolů je podkladem pro tvorbu budoucích odhadů. Tým se dohodne na základních jednotkách odhadu (dny, týdny, stránky atd.).

Tým musí zaujmout kolektivní shodu, co bude cílem odhadování při příštím sezení. Cíl je ve většině případů jednoznačný, ale je možné, že všichni členové s ním nemusí úplně souhlasit. Nesouhlas může být zaměřen na to, zda se odhady budou týkat i technické dokumentace, podpory produktu atd.

2.4.4 Individuální příprava

Po skončení kickoff meetingu moderátor sepíše všechny úkoly a předpoklady a distribuuje je týmu. Nezávisle si každý člen připraví podklady, které obsahují odhad pro každý z úkolů, podklady na základě kterých byly odhady vytvořeny a další úkoly, které by měly

být zahrnuty ve WBS a tým je vynechal při kickoff meetingu. Odhad pro každý úkol by měl být doplněn do části „Úkoly k dosažení cíle“ a sloupec „Čas“ by měl obsahovat odhadovaný čas pro každý z těchto úkolů. Čas každého úkolu by měl být vyplněn jako čas potřebný pro jeho splnění. Pokud například je standardní jednotka jeden den a je potřeba, aby první osoba pracovala 10 dní a druhá 6 dní, tak celkem je to 16 člověkodní.

Málokdy se stává, že všechny úkoly se perfektně překrývají, proto je nutné také zahrnout časové prodlevy, když se například čeká na nějaký výstup třetí strany (hardware, licenční povolení atd.). Časové prodlevy jsou evidovány zcela odděleně stejně tak jako čas nutný pro schůzky, reporty, dovolenou atd.

Při odhadování každého z úkolů si člověk uvědomí, že ve většině případů je nutný nějaký předpoklad pro jeho splnění. I tyto předpoklady musí být začleněny jako samostatné úkoly.

2.4.5 Diskuze nad odhady

Před tím než dojde k samotnému vyplnění formulářů, moderátor se zeptá účastníků, zda narazily na nějaké nové úkoly během jejich individuální přípravy. Každý z úkolů, na kterém se tým shodne je zařazen do WBS. Odhady pro tyto úkoly jsou řešeny později v rámci tohoto sezení.

Účastníkům jsou rozdány prázdné formuláře, kde každý vyplní úkoly a odhady, které si nachystal v individuální přípravě. V průběhu diskuze jsou účastníky upravovány časy jednotlivých odhadů na tomto formuláři.

Postupně se probírají jednotlivé úkoly, kdy moderátor u každého z úkolů posbírání údaje o odhadu účastníků a prezentuje je na viditelném místě. Každý úkol je diskutován v několika cyklech, kdy postupně dochází ke konvergenci časů jednotlivých odhadů. Moderátor by se měl také zaměřit na nejnižší a nejvyšší odhad, zda nedošlo k zanedbání některých předpokladů nebo zda by naopak úkol neměl být rozdělen na dílčí.

2.4.6 Sestavení výsledků

Po ukončení schůze nad odhady, spolupracuje moderátor s projektovým manažerem, aby shromáždili výsledky z dílčích příprav a samotného jednání. Manažer odebere nadbytečnosti a řeší zbývající drobné odchylky mezi jednotlivými odhady.

Je vytvořen podobný formulář, jaký měl k dispozici každý z účastníků v průběhu diskuze. U každého z úkolu jsou zachyceny odhady jednotlivých členů a je zvýrazněn nejnižší a

nejvyšší odhad. Úkol s velkým rozdílem mezi nejnižším a nejvyšším odhadem by měl být označen a podroben budoucí diskuzi. Zároveň je zachycena průměrná hodnota, u každého z úkolů. Při výpočtu průměrné hodnoty by měl projektový manažer vypustit nejvyšší a nejnižší hodnoty.

Může nastat i situace, že některý z členů stále nesouhlasí s odhadem pro konkrétní úkol. Projektový manažer musí být zvláště opatrný, pokud se jedná o úkol, který by měl zpracovávat tento člověk. Důležitou součástí této metody je zajištění shody mezi všemi účastníky na konečném harmonogramu. Názory všech členů by měly být respektovány ve stejném rozsahu, aby někdo nemohl nabyt dojmu, že jeho názory jsou ignorovány. Projektový manažer by měl předejít této situaci svoláním dodatečné diskuze, kde by měl být vyslyšeny argumenty osoby, která se odlišuje od společného odhadu. I přestože zbytek týmu nadále trvá na svém odhadu, konkrétní člověk alespoň nabude dojmu, že jeho námítky byly vyslyšeny a nebyly ihned odmítnuty.

2.4.7 Recenze výsledků

Projektový manažer svolá finální schůzku, jakmile jsou výsledky k dispozici, aby s nimi seznámil tým. Cílem schůzky je zjistit, zda výsledky sezení jsou dostatečné pro další plánování. Tým by měl určit, jestli odhady dávají smysl a zda je rozsah akceptovatelný. Dále by mělo dojít k ověření finálního listu úkolů, zda je kompletní. Stále se mohou vyskytovat úkoly, které je potřeba upřesnit. Tým se tedy domluví, že uspořádá další sezení, při kterém rozdělí tyto úkoly na dílčí úkoly. Toto je vhodné také udělat u úkolů, které mají velké rozdíly mezi jednotlivými odhady

2.5 Metoda funkčních bodů

Pro analýzu funkčních bodů jsou specifické následující kroky.

1. Odhaduje pracnost software kvantifikováním požadované funkcionality, která je nabízena zákazníkovi.
2. Odhaduje pracnost vývoje a údržby software nezávisle na technologii použité pro implementaci.
3. Odhaduje pracnost vývoje a údržby software důsledně napříč všemi projekty a organizacemi.

Při zpracování 2. a 3. bodu spousta společností disponuje rozsáhlým úložištěm funkčních bodů napříč projekty, technologiemi a organizacemi. Toto úložiště je bezvýznamné, pokud

je prováděn první odhad pracnosti, který je možné srovnat pouze s podobnými projekty, které byly zpracovány v jiných společnostech. [25]

Funkční body

Jednoduše řečeno funkční body jsou standardní jednotky funkcionalit, které dohromady představují funkcionalitu celé aplikace. Stejně tak, jako například dům je měřen ve čtverečních metrech, tak software je měřen počtem funkčních bodů, které jsou dodány zákazníkovi.

Před samotnou aplikací této metody je nutné reflektovat několik podmínek. Definování funkčních bodů je prováděno z pohledu zákazníka. Vše je tedy založeno na tom, že uživatel jakoby pracuje se systémem, kde jsou definovány vstupy a po zpracování těchto vstupů obdrží nějaké výstupy. Ve výsledku se jedná o pochopení toho, co musí být aplikací uloženo a zpracováno. Jak již bylo zmíněno, funkční body jsou technologicky neutrální. Nezáleží tedy, zda se jedná o webovou aplikaci vytvořenou v např. Java, PHP, .Net apod. Historicky je prokázáno, že zahrnutím této metody dojde ke zvýšení celkových nákladů na projekt pouze o 1 %, což znamená, že se jedná o nízkonákladovou metodu. Opakovatelnost je velmi důležitá, protože bez ní není možné důvěřovat datům, která jsou shromažďována v úložišti.

Pět standardních funkcí

Během počítání funkčních bodů se pracuje s pěti standardními funkcemi. První dvě se nazývají datové funkce a zbylé tři se nazývají transakční funkce. Jména těchto funkcí jsou uvedeny níže.

1. Datové funkce:
 - a. Interní logické soubory
 - b. Externí soubory rozhraní
2. Transakční funkce:
 - a. Externí vstupy
 - b. Externí výstupy
 - c. Externí požadavky

Před vlastním detailním popisem jednotlivých funkcí je nutné porozumět několika podmínkám a definicím.

1. Definované požadavky pro procesy nebo data jsou odsouhlaseny, jak budoucími uživateli, tak vývojáři systému.
2. Data ovlivňující systém a základní procesy musí být detailně zmapovány. Specifikuje se co, kdy nebo jaká data jsou zpracována.
3. Základní proces je nejmenší jednotka symbolizující aktivitu, která má pro uživatele smysl.
4. Data Element Type neboli DET je jedinečná, uživatelsky rozpoznatelná, neopakující se oblast. Tato definice je aplikována na analýzu datových i transakčních funkcí.
5. Record Element Type neboli RET je uživatelsky rozpoznatelná podskupina datových elementů s Interním logickým souborem a Externím souborem rozhraní.

Datová funkce – Interní logické soubory (ILFs)

Představují data, která jsou ukládána a udržována uvnitř aplikace, za kterou je sledován počet funkčních bodů.

Příklady, které mohou být zahrnuty do Interních logických souborů.

- Tabulky relační databáze.
- Textové a binární databázové soubory.
- Kontrolní informace aplikace, případně uživatelské předvolby, které jsou systémem ukládány.
- LDAP ukládání dat.

Není možné říct, že všechny tyto věci jsou automaticky zahrnuty do ILFs, pouze mohou být.

V průběhu počítání ILFs se vychází ze dvou tabulek. Smyslem první tabulky je určení, zda právě sledovaná ILFs má stupeň složitosti Low (L), Average (A) nebo High (H). Při pohledu do tabulky a stanovení počtu DETs a RETs, které jsou zahrnuty v ILF, je možné získat danou složitost.

Například při 5 DETs a 1 RET se jedná o složitost se stupněm Low. Naopak při 21 DETs a 2 RETs se jedná o složitost se stupněm Average.

RETs	Data Element Types (DETs)		
	1-19	20-50	51+
1	L	L	A
2 - 5	L	A	H
6 a více	A	H	H

Tab. 3. ILFs stupně složitosti

Na základě spočítané složitosti jsou jednotlivým složitostem přiřazeny funkční body. Matice složitosti je uvedena v tabulce (Tab. 4).

Složitost	Funkční body
Low	7
Average	10
High	15

Tab. 4. ILFs matice složitosti

Datová funkce – Externí soubory rozhraní (EIFs)

Představují data, která aplikace používá nebo se na ně odkazuje, ale nejsou v ní přímo uložena.

Příklady jsou obdobné jako v ILFs, ale opět je důležité si uvědomit, že hlavní rozdíl mezi nimi je, že data v EIFs nejsou uložena a udržována přímo v aplikaci, za kterou je sledován počet funkčních bodů.

Pro určení počtu funkčních bodů v EIFs se postupuje stejně jako u ILFs. Nejprve je stanoven počet DETs a RETs. Dle jejich počtu je určena složitost.

RETs	Data Element Types (DETs)		
	1-19	20-50	51+
1	L	L	A
2 - 5	L	A	H
6 a více	A	H	H

Tab. 5. EIFs stupně složitosti

Matice složitosti pro EIFs vypadá následovně.

Složitost	Funkční body
Low	5
Average	7
High	10

Tab. 6. EIFs matice složitosti

Transakční funkce – Externí vstupy (EIs)

Jedná se o základní procesy, které zpracovávají data nebo kontrolují informace přicházející do systému z jeho okolí. Hlavním úkolem EIs je udržovat jeden nebo více ILFs a/nebo měnit chování systému.

Příklady, které mohou být zahrnuty do Externích vstupů.

- Vstupní data zadaná uživateli.
- Data nebo soubory poskytované externími aplikacemi.

Přiřazení funkčních bodů probíhá obdobně jako pro ILFs a EIFs. RET je však nahrazeno tzv. FTR (File Type Referenced), což může být ILF nebo EIF.

V konkrétním případě může proces obsahovat 5 DETs, které odkazují na EIF nazvané Uživatelé a ILF nazvané Procesy. Dle níže uvedené tabulky připadá 5 DETs a 2 FTRs pro EIs složitost stupně Average.

FTR	Data Element Types (DETs)		
	1-4	5-15	16+
0-1	L	L	A
2	L	A	H
3 a více	A	H	H

Tab. 7. EIs stupně složitosti

Tabulka složitosti pro EI.

Složitost	Funkční body
Low	3
Average	4
High	6

Tab. 8. EIs matice složitosti

Transakční funkce – Externí výstupy (EOs)

Jedná se o elementární procesy, které posílají data nebo kontrolní informace mimo hranice aplikace. Hlavním úkolem Externích výstupů je poskytovat informace uživatelům prostřednictvím procesní logiky. Procesní logika musí obsahovat přinejmenším matematický vzorec nebo výpočet, který zpracovává uložená data v jednom nebo více Interním logickém souboru nebo mění chování systému.

Příklady, které mohou být zahrnuty do Externích výstupů.

- Aplikací vytvořený report, jehož zdrojem jsou data v databázi.

Pro konkrétní případ, který má např. 10 DETs a odkazuje na 2 FTRs je možné nalézt složitost v níže uvedených tabulkách.

FTR	Data Element Types (DETs)		
	1-5	6-19	20+
0-1	L	L	A
2-3	L	A	H
4 a více	A	H	H

Tab. 9. EOs stupně složitosti

Složitost	Funkční body
Low	4
Average	5
High	7

Tab. 10. EOs matice složitosti

Transakční funkce – Externí požadavky (EQs)

Externí dotazování jsou základním procesem, který posílá data nebo kontrolní informace mimo hranice aplikace. Hlavním úkolem externích dotazů je prezentovat informace uživatelům prostřednictvím získávání dat nebo kontrolou informací od Interních a Externích logických souborů. Procesní logika neobsahuje žádný matematický vzorec nebo výpočet a nevytváří žádná sekundární data. Žádný Interní logický soubor není udržován během zpracování a žádné chování systému není změněno.

Příklady, které mohou být zahrnuty do Externích požadavků.

- Report vytvořený aplikací neobsahující žádná odvozená data.

V konkrétním případě, který má 20 DETs a odkazuje na 4 FTRs je možné nalézt složitost v tabulkách (Tab. 11) a (Tab. 12).

FTR	Data Element Types (DETs)		
	1-5	6-19	20+
0-1	L	L	A
2-3	L	A	H
4 a více	A	H	H

Tab. 11. EQs stupně složitosti

Složitost	Funkční body
Low	3
Average	4
High	6

Tab. 12. EQs matice složitosti

Dohromady vzato jedná se o dvě datové a tři transakční funkce, které společně reprezentují pět funkcí, které jsou zahrnuty do funkčních bodů.

Pomocí metody funkčních bodů je možné přesněji odhadnout.

1. Náklady na projekt.
2. Průběh projektu.
3. Optimální počet pracovníků.

Pro zlepšení vývojového procesu je možné vést si níže uvedenou statistiku.

1. Počet člověkohodin na jeden funkční bod.
2. Náklady na jeden funkční bod.
3. Počet funkčních bodů zpracovaných za měsíc, týden nebo den.
4. Počet chyb připadajících na jeden funkční bod.

2.6 Use Case Points

Use case modelování je zobrazení funkční struktury systému z pohledu uživatele. Primárně je určen k definici systému, aniž by odhaloval jeho strukturu. Každý use case obsahuje vlastní scénář, což je posloupnost událostí, které v jeho rámci probíhají, včetně případných variant. Dále je v modelu zachycena interakce (komunikace) mezi uživatelem (aktorem) a systémem.

Obecně je možné říct, že aplikace obsahující velký počet komplikovaných use case, zabere více úsilí pro návrh a implementaci než malá aplikace s méně komplikovanými use case. Kromě toho čas nutný k vytvoření aplikace je ovlivněn následujícími body. [26]

- Počet kroků (událostí) pro dokončení use case.
- Počet a složitost aktorů.
- Technické požadavky na use case, jako je bezpečnost, výkon atd.
- Faktory prostředí, ve kterém je aplikace vytvářena, např. vývojový tým, zkušenosti, znalosti atd.

Use case points (UCP) je metoda pro odhadování velikosti a úsilí pro vytvoření aplikace na základě use case. Analýza zahrnuje aktory, scénáře a různé technické a environmentální faktory na základě kterých je sestavena výsledná rovnice.

Rovnice se skládá ze čtyř proměnných:

1. Technical Complexity Factor (TCF)

2. Environment Complexity Factor (ECF)
3. Unadjusted Use Case Points (UUCP)
4. Productivity Factor (PF)

Každá proměnná je vypočítána samostatně na základě hodnot a konstant. Kompletní rovnice vypadá následovně.

$$UCP = TCF * ECF * UUCP * PF$$

Pro stanovení odhadu založeného na UCP je nutné provést následující kroky.

1. Stanovit a spočítat Technical Complexity Factor
2. Stanovit a spočítat Environment Complexity Factor
3. Spočítat Unadjusted Use Case Points
4. Stanovit Productivity Factor
5. Vypočítat Use case points na základě předchozích čtyř bodů

Technical Complexity Factors

Je definováno třináct faktorů, které ovlivňují dopad technických vlastností na odhad rozsahu aplikace. Každému faktoru je přiřazen význam formou vah. Hodnota 0 signalizuje, že faktor je bezvýznamný a naopak hodnota 5 znamená, že má významný vliv.

	Technical Factor	Váha	Popis
T1	Distributed System	2	Má systém distribuovanou nebo centralizovanou architekturu.
T2	Performance	1	Je odezva systému důležité kritérium.
T3	End user efficiency	1	Efektivita systému pro uživatele.
T4	Complex internal processing	1	Složitost procesů zpracovávaných systémem.
T5	Reusability	1	Je snahou vytvářet znovupoužitelný kód.
T6	Easy to instar	0,5	Jednoduchost klientské instalace.
T7	Easy to use	0,5	Uživatelská přívětivost systému.
T8	Portable	2	Přenositelnost mezi platformami.
T9	Easy to chase	1	Parametrizovatelnost systému.
T10	Concurrent	1	Bude systém používat velké množství uživatelů.
T11	Special security features	1	Je požadováno SSL, kódování apod.
T12	Provides direct access for third parties	1	Spolupracuje systém s produkty třetích stran.
T13	Special user training facilities are required	1	Vyžaduje složitost systému extra školení.

Tab. 13. Technické faktory systému

Pro každý projekt jsou vývojovým týmem jednotlivým faktorům přiřazeny hodnoty v rozmezí od 0 do 5 dle složitosti, jak je vnímána. Například více vláknová aplikace vyžaduje více zkušeností než jedno vláknová aplikace, stejně tak nutnost znovupoužití

aplikace vyžaduje většinu funkcí přiřadit parametry a tím pádem i větší pracnost atd. Pokud je složitosti přiřazena hodnota 0 jedná se téměř o bezvýznamný technický faktor, 3 znamená průměr a 5 znamená velmi velký vliv.

Váha každého faktoru je vynásobena jeho vnímanou složitostí. Výsledkem součtu všech vypočítaných faktorů je Total Factor.

Konečný Technical Complexity Factor (TCF) je vypočten dle vzorce.

$$TCF = 0,6 + (0,01 * Total Factor)$$

Environment Complexity Factor

Je definováno osm faktorů, které ovlivňují dopad vlastností prostředí na odhad rozsahu aplikace. Každému faktoru je přiřazen význam formou vah. Hodnota 0 signalizuje, že faktor je bezvýznamný a naopak hodnota 5 znamená, že má významný vliv.

	Environment Factor	Váha	Popis
E1	Familiarity with project	1,5	Stupeň seznámení pracovníků s detaily projektu.
E2	Application Experience	0,5	Zkušenosti s podobnou aplikací.
E3	Object Oriented Experience	1	Jak je důležité, že pracovníci ovládají OOP.
E4	Lead analyst capability	0,5	Zkušenost pracovníka vedoucí projekt za analytické oddělení.
E5	Motivation	1	Jsou programátoři motivováni při práci na projektu.
E6	Stable Requirements	2	Jak moc je si klient jistý co vlastně chce.
E7	Part-time workers	-1	Podílejí se na projektu lidé se zkráceným pracovním úvazkem.
E8	Difficult Programming language	-1	Složitost programovacího jazyka.

Tab. 14. Environmentální faktory systému

Stejně jako u TCF je váha každého faktoru vynásobena jeho vnímanou složitostí. Výsledkem součtu všech vypočítaných faktorů je Total Factor.

Konečný Environment Complexity Factor (ECF) je vypočten dle vzorce.

$$ECF = 1,4 + (-0,03 * Total Factor)$$

Unadjusted Use Case Points

UUCP jsou získány na základě dvou výpočtů.

1. Unadjusted Use Case Weight (UUCW) – který je založen na celkovém počtu kroků obsažených ve všech use case scénářích.
2. Unadjusted Actor Weight (UAW) – který je založen na složitosti všech aktorů.

Unadjusted Use Case Weight

Jednotlivé use case jsou rozděleny do kategorií Simple, Average a Complex. Každé této kategorii je přiřazena váha dle počtů kroků vč. kroků alternativních.

Typ Use Case	Popis	Váha
Simple	Jednoduché uživatelské rozhraní, které je napojeno na jednu databázovou entitu. Scénář obsahuje maximálně 3 kroky. Implementace zahrnuje maximálně 5 tříd.	5
Average	Složitější uživatelské rozhraní, které je napojeno na 2 a více databázových entit. Scénář obsahuje 4-7 kroků. Implementace zahrnuje 5-10 tříd.	10
Complex	Složitě uživatelské prostředí, které je napojeno na 3 a více databázových entit. Scénář obsahuje více než 7 kroků. Implementace zahrnuje více než 10 tříd.	15

Tab. 15. Rozdělení use case dle složitosti

UUCW je vypočítáno následovně. Nejprve je vynásoben počet všech use case, které spadají do dané kategorie váhou této kategorie. Hodnoty za každou kategorii jsou sečteny, čehož výsledkem je celkové UUCW.

Unadjusted Actor Weight

Podobným způsobem jsou rozděleny do skupin Simple, Average a Complex i aktori.

Typ Aktora	Popis	Váha
Simple	Aktor představující jiný systém s přesně definovaným API.	1
Average	Aktor představující jiný systém komunikující přes protokol, např. TCP/IP.	2
Complex	Aktor je osoba, která ovládá systém přes uživatelské rozhraní.	3

Tab. 16. Rozdělení aktorů dle složitosti

UAW je vypočítáno následovně. Nejprve je vynásoben počet všech aktorů, kteří spadají do dané kategorie váhou této kategorie. Výsledky za každou kategorii jsou sečteny, jehož výsledkem je celkové UAW.

Konečné Unadjusted Use Case Points (UUCP) je vypočítáno součtem UUCW a UAW, tzn. dle vzorce.

$$UUCP = UUCW + UAW$$

Productivity Factor (PF)

Productivity Factor je koeficient, který představuje počet člověkohodin, které je obvykle nutné vynaložit na zpracování jednoho use case. Hodnota je založena na zkušenosti z minulých projektů. Pokud nejsou k dispozici žádná historická data, tak doporučená hodnota je v rozmezí 15-30. Typická hodnota je 20.

Finální výpočet

Use case points je získán vynásobením všech výše vypočítaných proměnných. Hodnota UCP odhad pracnosti projektu v člověkohodinách.

$$UCP = TCF * ECF * UUCP * PF$$

Výhody a nevýhody

Na začátku projektu za účelem zpřesnění odhadu je možné proměnné rovnice upravit a přizpůsobit.

Počet kroků ve scénáři use case ovlivňuje také odhad. Velký počet kroků dělá systém složitější a zvyšuje hodnotu UCP naopak malý počet kroků směřuje k jednoduchosti. Za určitých okolností může dojít ke snížení počtu kroků, ale se zachováním klíčových kroků aplikace.

Pro snížení složitosti je možné use case s vazbou include nebo exclude vypustit, následně tyto use case budou vystupovat pouze jako jeden.

Počet aktorů v use case také ovlivňuje odhad pracnosti. Pokud to situace umožňuje, tak pod více aktory si lze představit jednoho superaktora. Tato operace sníží složitost, ale neovlivňuje samotné use case.

Technické a environmentální faktory musí být neustále přizpůsobovány, dle aktuálně obdržených dat. Přesnou hodnotu Productivity Factor je možné získat až v průběhu času. Je nutné sledovat čas strávený navrhováním a implementací jednotlivých use case a dle získaných hodnot přizpůsobit tento faktor.

2.7 COCOMO II

COCOMO neboli COConstructive COst MOdel je parametrický matematický model, jehož podkladem pro odhad pracnosti projektu je jeho fyzický rozsah vyjádřený jednotlivými řádky kódu (LOC).

Vzhledem k tomu, že formální metody používají jako vstup velikost projektu, je nutné, aby tato predikovaná velikost byla co nejpřesnější.

Velikost aplikace je možné odhadnout následovně. [27]

1. Použité funkce a metody se rozdělí do těchto kategorií.
 - a. Nový návrh a nový kód.

- b. Podobný návrh a nový kód.
 - c. Podobný návrh a částečně znovu použitelný kód.
 - d. Podobný návrh a použitelnost kódu ve velkém rozsahu.
2. Rozsah metod a funkcí je možné také získat následovně.
- a. Analogie a expertní odhady – dle modifikovatelnosti nebo znouvupoužitelnosti je stanoven rozsah každé funkce či metody. Tento odhad může být vytvořen odborným odhadem nebo analogií dle historických projektů. Odborný odhad je založen na zkušenosti s podobnou funkcionalitou, zatímco analogie je založena na podobnostech a rozdílech s minulými projekty.
 - b. Statistický přístup – pro podobné nebo kompletně nové funkce, kde chybí zkušenost a historická data nebo projekty s neúplnými požadavky je nutné provést odhad nejmenší (Least), největší (Most) a nejpravděpodobnější (Likely) velikosti aplikace. Dle metody Program Evaluation Review Technique (PERT) je následně vypočítán očekávaný rozsah

$$Rozsah = (Least + 4 * Likely + Most) / 6$$

Do výpočtu LOC nejsou zahrnuty prázdné řádky a komentáře. Zároveň se vypouští části kódu generované programem, obsaženy jsou pouze zdrojové řádky napsány člověkem. Za jeden LOC se považuje jeden logický řádek kódu, tzn. například příkaz "if-then-else" je počítán jako jeden zdrojový řádek kódu. [28]

Obecný odhad pracnosti projektu v člověkoměsících je vypočten dle následujícího vzorce.

$$Effort = a * EAF * KLOC^b$$

Konstanta „a“ reflektuje organizační a technologické náklady, její hodnota je 2,94. EAF je korekční subjektivní faktor nákladů. KLOC je počet řádků kódu vyjádřený v tisících. Exponent „b“ je měřítkem organizačních okolností projektu, doporučená hodnota je 1,0997 nebo může být vypočítána na základě SF (Scale Factor).

Celkový výpočet je tedy upraven subjektivním faktorem nákladů EAF (Effort Adjustment Factor). Faktor nákladů zahrnuje produktové, hardwarové, personální a projektové atributy. Každému z patnácti atributů je přiřazen rating z šestibodové stupnice (Very low, Low, Nominal, High, Very High, Extra High) dle jeho důležitosti. Na základě ratingu jsou jednotlivým nákladovým atributům přiřazeny koeficienty v typickém rozmezí 0,7–1,66. Nákladové faktory jsou uvedeny v tabulce (Tab. 17).

Nákladový faktor	Rating					
	Very Low	Low	Nominal	High	Very High	Extra High
Produktové atributy						
Požadovaná spolehlivost	0,75	0,88	1,00	1,15	1,40	-
Velikost databáze aplikace		0,94	1,00	1,08	1,16	-
Složitost produktu	0,70	0,85	1,00	1,15	1,30	1,65
Hardwarové atributy						
Výkonové omezení	-	-	1,00	1,11	1,30	1,66
Omezení kapacity pevného disku	-	-	1,00	1,06	1,21	1,56
Kolísavost virtuálních prostředí	-	0,87	1,00	1,15	1,30	-
Požadovaná odezva	-	0,87	1,00	1,07	1,15	-
Personální tributy						
Kapacita analytické oddělení	1,46	1,19	1,00	0,86	0,71	-
Zkušenost s vývojem aplikací	1,29	1,13	1,00	0,91	0,82	-
Kapacita vývojového oddělení	1,42	1,17	1,00	0,86	0,70	-
Zkušenost s virtuálními prostředími	1,21	1,10	1,00	0,90	-	-
Zkušenost s programovacím jazykem	1,14	1,07	1,00	0,95	-	-
Projektové atributy						
Aplikování softwarových metod a procesů	1,24	1,10	1,00	0,91	0,82	-
Používání softwarových utilit	1,24	1,10	1,00	0,91	0,83	-
Dodržování časového plánu	1,23	1,08	1,00	1,04	1,10	-

Tab. 17. Nákladové faktory aplikace

Hodnota EAF je získána vynásobením všech patnácti nákladových faktorů.

Jak bylo zmíněno výše dalšími faktory, které přispívají k průběhu projektu a jeho nákladům jsou Scale Factors. Hodnoty jednotlivých faktorů se stanoví dle tabulky (Tab. 18) z kterých se vypočítá konečná hodnota exponentu „b“. [29]

Scale Factor (W_i)	Rating					
	Very Low	Low	Nominal	High	Very High	Extra High
PREC (Precedentedness)	6,20	4,96	3,72	2,48	1,24	0,00
FLEX (Development Flexibility)	5,07	4,05	3,04	2,03	1,01	0,00
RESL (Architecture / Risk Resolution)	7,07	5,65	4,24	2,83	1,41	0,00
TEAM (Team Cohesion)	5,48	4,38	3,29	2,19	1,10	0,00
PMAT (Process Maturity)	7,80	6,24	4,68	3,12	1,56	0,00

Tab. 18. Scale Factors

PREC (Precedentedness) reflektuje podobnost s projekty vyvíjenými v minulosti. FLEX (Development Flexibility) odráží stupeň flexibility ve vývojovém procesu. RESL (Architecture / Risk Resolution) zachycuje, zda se bere ohled na architekturu systému a zda se zohledňují možná rizika projektu. TEAM (Team Cohesion) reflektuje, jak dobře se

znají jednotlivý členové vývojového týmu a jak spolu spolupracují. PMAT (Process Maturity) představuje model zralosti firmy.

Každému faktoru je opět přiřazen rating z šestibodové stupnice (Very low, Low, Nominal, High, Very High, Extra High) dle jeho významu. Stupeň Very low je nejhorší hodnota, která zvyšuje pracnost projektu, naopak Extra High je nejlepší hodnota, která pracnost projektu nechává bez změny. Na základě ratingu je každému faktoru přidělena specifická hodnota, tzv. váha (W_i). Jednotlivé váhy jsou sečteny a koeficient „b“ je vypočítán dle následujícího vzorce.

$$b = 0,91 + 0,01 * \sum W_i$$

II. PRAKTICKÁ ČÁST

3 SROVNÁNÍ VYBRANÝCH METOD

Pro komparaci metod na reálném projektu jsme zvolili Use Case Points a COCOMO II z důvodu toho, že všechny informace a zdrojová data potřebná pro jejich srovnání jsou v daném projektu k dispozici. Vzhledem k složitosti systému jsme analyzovali pouze jednu z jeho komponent.

Komponenta LBS byla navržena v programu Enterprise Architect společnosti Sparx Systems. Program používá pro návrh objektově orientovaný standardizovaný modelovací jazyk UML (Unified Modeling Language). Informace jsme získávaly především z Use Case Modelu, který zachycuje hranice a interakci mezi částmi systému a jeho uživateli. Samotná aplikace byla vytvořena v Microsoft Visual Studiu 2010 programovacím jazykem C# (C Sharp).

Komponenta aplikace funguje jako Windows Service, což znamená, že se jedná o dlouhodobě spuštěnou část systému, která běží na aplikačním serveru a systému poskytuje určité funkcionality. LBS komponenta komunikuje se čtyřmi aktory. První z nich je další Windows Service, která zpracovává tok dat ve formátu XML a předává je LBS. Druhým aktorem je uživatelské rozhraní v kterém je možné sledovat předávaná data do LBS a ovlivňovat jejich ukládání do databáze. Třetím aktorem je vlastní MS SQL databáze a posledním aktorem je prezentační rozhraní.

3.1 Use Case Points

Abychom mohli získat hodnou UCP v člověkohodinách, tak nejprve jsme museli získat hodnoty jeho čtyř činitelů.

Technical Complexity Factors

Vzhledem k tomu, že webové, aplikační i databázové servery běží v clusteru, tak jsme faktoru T1, přiřadili složitost čtyři. Stejnou složitost mají ještě faktory T2, T3 a T13. Je kladen důraz na výkon a rychlost systému, protože musí obsloužit spousty zákazníků za krátký časový okamžik. Uživatelský komfort a přívětivost komponenty je také velmi důležitá, a pokud je systém zaváděn, tak i přes existující příručky a manuály vyžaduje zaškolení zákazníka do provozu. Na opačnou stranu složitosti jsme přiřadili faktory T8 a T11, protože aplikace jako celek vůbec neřeší přenositelnost mezi jednotlivými platformami, striktně běží pouze na serverových řešeních firmy Microsoft a bezpečnost není předmětem této části systému.

	Technical Factor	Váha	Složitost	Váha * Složitost
T1	Distributed System	2	4	8
T2	Performance	1	4	4
T3	End user efficiency	1	4	4
T4	Complex internal processing	1	3	3
T5	Reusability	1	2	2
T6	Easy to instar	0,5	1	0,5
T7	Easy to use	0,5	3	1,5
T8	Portable	2	0	0
T9	Easy to chase	1	2	2
T10	Concurrent	1	2	2
T11	Special security features	1	0	0
T12	Provides direct access for third parties	1	3	3
T13	Special user training facilities are required	1	4	4
	Total Factor			34

Tab. 19. Technické faktory komponentu LBS

Poté co jsme všem faktorům přiřadili složitosti, tak jsme tyto složitosti vynásobili váhami stanovené pro jednotlivé technické faktory. Vypočtenou hodnotu Total Factor, jsme dosadili do následujícího vzorce.

$$TCF = 0,6 + (0,01 * 34)$$

Vypočtená hodnota Technical Complexity Factors (TCF) pro komponentu LBS je 0,94.

Environment Complexity Factor

Složitosti s největším vlivem jsme přiřadili k faktorům E1, E3 a E4. Dobrá znalost celkové aplikace je vyžadována k tomu, aby se člověk mohl plnohodnotně podílet na jejím vývoji. Zaškolení a proniknutí do dané problematiky trvá několik měsíců. Použití objektově orientovaného přístupu je při vývoji této komponenty naprostou nutností, proto jsme ji přiřadili vnímanou složitost pět. Třetím velmi podstatným faktorem jsou zkušenosti a přehled vedoucího pracovníka za analytické oddělení, aby docházelo k rozvoji produktu správným směrem a dokázal včas upozornit na problémy, které mohou nastat. Nejnižší složitost jsme přiřadili faktoru E7, což představuje zaměstnance pracující ve firmě na zkrácený pracovní úvazek. Po v minulosti provedené restrukturalizaci byl s těmito pracovníky rozvázán pracovní poměr, protože se v důsledku ukázalo, že i když se jedná o levnější pracovní sílu, tak jejich částečná nepřítomnost na pracovišti je pro firmu spíše obtíž než přínosem.

	Environment Factor	Váha	Složitost	Váha * Složitost
E1	Familiarity with project	1,5	4	6
E2	Application Experience	0,5	2	1
E3	Object Oriented Experience	1	5	5
E4	Lead analyst capability	0,5	4	2
E5	Motivation	1	3	3
E6	Stable Requirements	2	2	4
E7	Part-time workers	-1	0	0
E8	Difficult Programming language	-1	3	-3
	Total Factor			18

Tab. 20. Faktory prostředí pro komponentu LBS

Definované vnímané složitosti jsme vynásobili váhami jednotlivých faktorů. Vypočtenou hodnotu Total Factor za ECF, jsme dosadili do následujícího vzorce.

$$ECF = 1,4 + (-0,03 * 18)$$

Vypočtená hodnota Environment Complexity Factor (ECF) pro komponentu LBS je 0,86.

Unadjusted Use Case Points

Činitele UUCP jsme získali na základě součtu dvou dílčích hodnot Unadjusted Use Case Weight (UUCW) a Unadjusted Actor Weight (UAW).

Abychom mohli stanovit hodnotu UUCW, museli jsme nejprve jednotlivé use case na základě kterých byla vytvořena komponenta LBS rozdělit do tří skupin dle počtu kroků ve scénáři, tříd a databázových entit. Celkový počet use case je 41. Use Case Diagram je uveden v příloze P I.

Typ Use Case	Váha	Počet	Váha * Počet
Simple	5	23	115
Average	10	17	170
Complex	15	1	15
UUCW			300

Tab. 21. UUCW pro LBS

Počet use case zařazených do skupin jsme vynásobili váhami přiřazenými každé skupině. Po sečtení těchto hodnot jsme obdrželi UUCW, která je pro komponentu LBS 300.

Podobně jako use case i aktory komunikující s komponentou LBS jsme rozdělili do skupin dle jejich vlastností. Tři přistupují k LBS přes API rozhraní (Windows Service, MS SQL databáze a prezentační rozhraní), proto byly zařazeny do skupiny Simple. Čtvrtým aktorem

je uživatel, který ji ovládá přes uživatelské rozhraní, proto byl zařazen do skupiny Complex.

Typ Aktora	Váha	Počet	Váha * Počet
Simple	1	3	3
Average	2	0	0
Complex	3	1	3
UAW			6

Tab. 22. UAW pro LBS

Počet aktorů zařazených do skupin jsme vynásobili váhami přiřazenými každé skupině. Po sečtení těchto hodnot jsme získali hodnotu UAW, která je pro komponentu LBS 6.

Vypočtené hodnoty UUCW a UAW jsme dosadili do následujícího vzorce.

$$UUCP = 300 + 6$$

Vypočtená hodnota Unadjusted Use Case Points (UUCP) pro komponentu LBS je 306.

Productivity Factor (PF)

Vzhledem k tomu, že daná komponenta byla v rámci firmy vyvíjena poprvé, tak nebyla k dispozici žádná historická data. Pro PF byla tedy zvolena typická hodnota 20.

Finální výpočet

Use case points jsme získali vynásobením všech výše vypočítaných činitelů.

$$UCP = 0,94 * 0,86 * 306 * 20$$

Hodnota Use case points (UCP) v člověkohodinách pro komponentu LBS je 4948,4080. Po převedení na člověkoměsíce jsme dostali hodnotu 30,9213.

3.2 COCOMO II

Při výpočtu odhadu pracnosti dle COCOMO II se vychází z počtu logických řádků kódu. My jsme měli tu výhodu, že projekt již byl realizován v minulosti a mohli jsme využít funkcionality Code Metrics dostupné ve Visual Studiu 2010. Pokud je daný projekt přidán v Solution Exploreru je možné aplikovat tuto funkci a získat tak řádky kódu za daný projekt nebo část projektu. V případě komponenty LBS je to 6239 řádků kódu. Pro použití v této metodě musely být ještě řádky kódu převedeny na tisíce, tzn. 6,239 tis. řádků kódu.

Nejprve jsme spočítali odhad pracnosti použitím níže uvedeného vzorce s doporučenými hodnotami parametrů a koeficientů bez korekčních faktorů.

$$Effort = 2,94 * 6,239^{1,0997}$$

Odhadovaná pracnost v člověkoměsících je v tomto případě 22,0158.

Po zapojení nákladových faktorů EAF jsme každému z nich přiřadili rating, který je uveden v následující tabulce v šedě podbarvených polích.

Nákladový faktor	Rating					
	Very Low	Low	Nominal	High	Very High	Extra High
Produktové atributy						
Požadovaná spolehlivost	0,75	0,88	1	1,15	1,4	-
Velikost databáze aplikace		0,94	1	1,08	1,16	-
Složitost produktu	0,7	0,85	1	1,15	1,3	1,65
Hardwarové atributy						
Výkonové omezení	-	-	1	1,11	1,3	1,66
Omezení kapacity pevného disku	-	-	1	1,06	1,21	1,56
Kolísavost virtuálních prostředí	-	0,87	1	1,15	1,3	-
Požadovaná odezva	-	0,87	1	1,07	1,15	-
Personální tributy						
Kapacita analytické oddělení	1,46	1,19	1	0,86	0,71	-
Zkušenost s vývojem aplikací	1,29	1,13	1	0,91	0,82	-
Kapacita vývojového oddělení	1,42	1,17	1	0,86	0,7	-
Zkušenost s virtuálními prostředími	1,21	1,1	1	0,9	-	-
Zkušenost s programovacím jazykem	1,14	1,07	1	0,95	-	-
Projektové atributy						
Aplikování softwarových metod a procesů	1,24	1,1	1	0,91	0,82	-
Používání softwarových utilit	1,24	1,1	1	0,91	0,83	-
Dodržování časového plánu	1,23	1,08	1	1,04	1,1	-

Tab. 23. Nákladové faktory komponenty LBS

Vynásobením vybraných ratingů obdržíme hodnotu EAF, kterou doplníme do vzorce pro výpočet pracnosti. V druhé situaci je tato hodnota 1,2126.

$$Effort = 2,94 * 1,2126 * 6,239^{1,0997}$$

Odhadovaná pracnost v člověkoměsících je v případě zahrnutí nákladových faktorů do výpočtu 26,6971.

Do posledního výpočtu jsme zapojili i Scale Factors, které nahrazují doporučenou hodnotu exponentu „b“. Z druhého výpočtu zůstala již zahrnuta hodnota EAF. Po zapojení tzv.

nákladů prostředí byl každému z nich přiřazen rating, který zachycuje následující tabulka v šedě podbarvených polích.

Scale Factor (W_i)	Rating					
	Very Low	Low	Nominal	High	Very High	Extra High
PREC (Precedentedness)	6,2	4,96	3,72	2,48	1,24	0
FLEX (Development Flexibility)	5,07	4,05	3,04	2,03	1,01	0
RESL (Architecture / Risk Resolution)	7,07	5,65	4,24	2,83	1,41	0
TEAM (Team Cohesion)	5,48	4,38	3,29	2,19	1,1	0
PMAT (Process Maturity)	7,8	6,24	4,68	3,12	1,56	0

Tab. 24. Scale Factors pro komponentu LBS

Sečtením vybraných ratingů jsme získali hodnotu 17,01, kterou jsme dosadili do následujícího vzorce, abychom vypočítali vlastní exponent „b“.

$$b = 0,91 + 0,01 * 17,01$$

Exponent „b“ má hodnotu 1,0801. Toto číslo dosadíme od vzorce pro výpočet pracnosti.

$$Effort = 2,94 * 1,2126 * 6,239^{1,0801}$$

Odhadovaná pracnost v člověkoměsících je v případě zahrnutí nákladových faktorů i faktorů prostředí do výpočtu 25,7561.

3.3 Vyhodnocení srovnání

V tabulce (Tab. 25) jsou uvedeny odhady pracnosti v člověkoměsících pro komponentu LBS. Jak je vidět množství práce, kterou bude muset vynaložit jedna osoba, se pohybuje v rozmezí 22 až téměř 31 měsíců. Rozpětí mezi nejkratším a nejdelším odhadem je přibližně 9 měsíců, což je hrozně vypadající dlouhá doba.

Metoda	Odhadovaná pracnost (člověkoměsíce)
Use Case Points	30,9213
COCOMO II - standardní parametry	22,0158
COCOMO II - nákladové faktory EAF	26,6971
COCOMO II - nákladové faktory EAF i Scale Factors	25,7561

Tab. 25. Srovnání metod pro odhad pracnosti

Pokud budeme předpokládat, že by na projektu pracovalo např. pět osob, tak odhadovaná pracnost se pohybuje v rozmezí 4,4 až 6,2 měsíce, což na první pohled nemusí vypadat jako velký rozdíl. Z našeho pohledu se ale stále jedná o velkou časovou prodlevu, protože ve většině případů jsou na nesplnění termínů vázány penále nebo další dílčí činnosti.

Nejnižší odhadovaná pracnost byla získána pomocí metody COCOMO II se standardními parametry. Jak je viditelné z tabulky (Tab. 25), tak po zapojení nákladových faktorů a faktorů prostředí odhadovaná pracnost o několik měsíců vzrostla. Zde je jasně viditelné, jak zahrnutí dalších parametrů a změna jejich ratingu může ovlivnit celou odhadovanou pracnost. Metoda Use Case Points se naopak vyznačuje nejvyšší odhadovanou pracností a i COCOMO II s nejnákladnějším odhadem si ji přiblížilo pouze na rozdíl čtyř měsíců. Odhad pracnosti dle Use Case Points v největší míře ovlivňuje samotný návrh vytvořený analytiky. Počet jednotlivých use case, počet kroků v samotných use case, množství vazeb include a extend, to jsou všechno věci, které přímo působí na odhadovanou pracnost projektu.

Nemůžeme tedy jednoznačně říci, která z daných metod je pro dodavatele software lepší. V obou metodách existuje mnoho parametrů, jejichž malá změna ovlivňuje odhadovanou pracnost. Naším doporučením proto je vybrat si jednu z nich a tu postupně na několika projektech přizpůsobit vlastním potřebám.

4 STAV VYUŽÍVÁNÍ METOD ODHADU V SOFTWAREVÝCH FIRMÁCH

Cílem této kapitoly bylo zpracovat dotazník zabývající se situací využívání metod pro odhadování pracnosti software v jednotlivých společnostech. Byly stanoveny dvě hypotézy, které jsme se snažili ponechat v platnosti nebo vyvrátit. Zároveň jsme přispěli s vlastní zkušeností ohledně aplikace přístupů v současných firmách.

4.1 Zpracování dotazníku (seznámení s dotazníkem)

K vytvoření dotazníku vedla snaha o zmapování současného stavu v softwarevých firmách, co se týče používání a aplikace metod pro odhadování pracnosti software. Zajímalo nás to především v době, kdy probíhá celková ekonomická krize, která se promítá i do tvorby softwarevých produktů. O to více je v současné době vyvíjen tlak na snížení rozpočtu vyhrazeného pro tyto produkty. Společnosti proto musí více dbát na dodržení firemních procesů, dodat software ve stanoveném termínu, v odpovídající kvalitě a za předem stanovenou cenu. Z toho vyplývá, že dodavatel již v počáteční fázi musí co nejpřesněji odhadnout celkové náklady související s návrhem, tvorbou a integrací budoucího systému.

Záměrně jsme zvolili anonymní dotazník, abychom docílili co největší zpětné vazby od respondentů. Přeci jenom ne každý chce odhalit svoji identitu, když jeho postupy nejsou například 100%. Průzkum byl prováděn přesně 267 hodin (přibližně 11 dní) v období 13.02.2011 – 24.02.2011. Celkem bylo e-mailem osloveno 78 softwarevých firem, jejichž kontakty byly zjištěny prostřednictvím vyhledávače Seznam.

Celkový počet otázek v dotazníku byl 16, což jsou zhruba $\frac{3}{4}$ doporučeného rozsahu dotazníku takového typu. Vzhledem k tomu, že v dotazníku bylo obsaženo větvení, tak v širší větvi respondenti mohli odpovědět na 15 otázek a v užší větvi pouze na 8.

Samotné znění dotazníku vypadá následovně.

Metody odhadování pracnosti softwarevých projektů

1. Počet zaměstnanců ve společnosti? (povinná, forma – seznam, právě jedna správná odpověď)
 - 1-10
 - 11-25
 - 26-50

- 51-100
- 101 a více

Účel otázky: zjistit počet zaměstnanců ve společnosti.

2. Objem zahraničního kapitálu ve společnosti? (povinná, forma – seznam, právě jedna správná odpověď)

- 0-25 %
- 26-50 %
- 51-75 %
- 76-100 %

Účel otázky: zjistit podíl zahraničního kapitálu ve firmě.

3. Počet zpracovávaných projektů ročně? (povinná, forma – seznam, právě jedna správná odpověď)

- 1-5
- 6-10
- 11-20
- 21 a více

Účel otázky: zjistit počet projektů absolvovaných v průběhu kalendářního roku.

4. Používáte metody pro odhadování nákladů projektu? (povinná, forma – seznam, právě jedna správná odpověď – rozdělující)

- Ano (větvení, pokračuje se 5. otázkou)
- Ne (větvení, pokračuje se 6. otázkou)

Účel otázky: otázka v které nastalo větvení a z toho důvodu respondent odpověděl na 8 nebo 15 otázek. K větvení docházelo z důvodu toho, že pokud respondent odpověděl na otázku záporně, nemělo smysl mu pokládat další otázky vztahující se k metodám odhadování pracnosti projektu.

5. Jaké konkrétní metody pro odhadování nákladů používáte? (dále se pokračuje 7. otázkou, povinná, forma – seznam, právě jedna správná odpověď – polouzavřená)

- Ad hoc, tzn. nelze jednoznačně specifikovat
- Neformální analogie, založená na zkušenostech osoby provádějící odhad
- Formální analogie, dle udržované databáze předešlých projektů

- Formální modely jako např. COCOMO
- Jiná odpověď:

Účel otázky: zjistit jaké konkrétní techniky, metody, přístupy používá respondent k odhadu.

6. Nepoužíváte metody pro odhadování nákladů, na základě čeho zpracováváte cenové nabídky? (dále se pokračuje 14. otázkou, povinná, delší text)

- Otevřená odpověď:

Účel otázky: pokud respondent nepoužívá metody pro odhadování pracnosti, tak na základě čeho odhaduje pracnost a tím pádem stanovuje cenu.

7. Jaké vstupy jsou používány pro odhadování nákladů? (povinná, forma – seznam, právě jedna správná odpověď – polouzavřená)

- Neúplný soubor požadavků
- Detailní soubor požadavků
- Detailní uživatelské rozhraní, zahrnující prototypy jednotlivých obrazovek
- Kompletní architektura systému
- Jiná odpověď:

Účel otázky: zjistit co je zdrojem pro tvorbu odhadů.

8. Kolik osob je zapojeno do zpracování odhadu nákladů? (povinná, forma – seznam, právě jedna správná odpověď)

- 1-2
- 3-5
- 6-10
- 11 a více

Účel otázky: zjistit, kolik lidí je tázáno nebo má právo mluvit do odhadů pracnosti.

9. Jaké jsou pozice osob provádějících odhady nákladů? (povinná, forma – seznam, alespoň jedna správná odpověď – polouzavřená)

- Majitel společnosti
- Finanční manažer
- Projektový manažer
- Analytik

- Databázový specialista
- Vývojář
- Tester
- Vlastní odpověď:

Účel otázky: zjistit, jaké jsou pracovní pozice osob, které jsou zapojeny do odhadů.

10. Bývají osoby provádějící odhady nákladů omezeny dostupnými zdroji, např. rozpočet, omezená pracovní síla, čas atd.? (povinná, forma – seznam, právě jedna správná odpověď)

- Ano
- Spíše ano
- Spíše ne
- Ne

Účel otázky: zjistit, zda osoby zapojené do odhadu ihned na začátku disponují omezenými zdroji.

11. U kolika procent projektů bylo nutné zredukovat odhadované náklady, protože výsledná cenová nabídka nebyla konkurenceschopná? (povinná, forma – seznam, právě jedna správná odpověď)

- 0-25 %
- 26-50 %
- 51-75 %
- 76-100 %

Účel otázky: zjistit, zda dochází k situacím, že je nutné snížit odhadovanou pracnost, protože její cena není konkurenceschopná.

12. Jak často dochází ke konfrontaci finančního manažera a projektového manažera ohledně výše odhadovaných nákladů? (povinná, forma – seznam, právě jedna správná odpověď)

- Vždy
- Velmi často
- Často
- Ojedinele
- Vůbec

Účel otázky: zjistit, zda ve společnosti nastávají rozepře ohledně ceny odhadovaných nákladů.

13. Realizujete na počátku každého projektu tzv. "kickoff meeting" a sezení týkající se odhadu pracnosti? (povinná, forma – seznam, právě jedna správná odpověď)

- Ano
- Spíše ano
- Spíše ne
- Ne

Účel otázky: zjistit, zda se na počátku každého projektu pořádá úvodní schůzka se všemi zúčastněnými.

14. Kolik procent projektů nebývá dodáno včas? (povinná, forma – seznam, právě jedna správná odpověď)

- 0-25 %
- 26-50 %
- 51-75 %
- 76-100 %

Účel otázky: zjistit, v kolika případech se projekt dostane do takové fáze, že není dodán včas.

15. U kolika procent projektů dochází k překročení počátečního rozpočtu? (povinná, forma – seznam, právě jedna správná odpověď)

- 0-25 %
- 26-50 %
- 51-75 %
- 76-100 %

Účel otázky: zjistit, v kolika případech došlo k překročení plánovaného rozpočtu.

16. U Kolika procent projektů dochází ke snížení kvality produktu (např. vynechání některých zavedených procesů), aby byl dodržen počáteční rozpočet nebo termín předání? (povinná, forma – seznam, právě jedna správná odpověď)

- 0-25 %
- 26-50 %
- 51-75 %

- 76-100 %

Účel otázky: zjistit, v kolika případech došlo ke snížení kvality produktu.

4.2 Stanovení hypotéz

Dotazník nám měl také pomoci zamítnout nebo ponechat v platnost níže uvedené hypotézy.

Více než 50 % společností aplikující metody pro odhadování pracnosti software, jsou schopné dodat software včas.

U této hypotézy jsme vycházeli z předpokladu, že pokud společnost aplikuje metody pro odhadování pracnosti software, tak jejich systémy jsou dodávány včas nebo s minimálním časovým zpožděním. Použití sofistikovaných metod by se tedy mělo odrazit v schopnosti stanovit si, kolik osob pro daný projekt bude nutné vyčlenit a jak dlouho bude trvat splnění požadavků zákazníka.

Klíčové otázky: 4, 6, 14, 15, 16.

Odhady pracnosti omezené dostupnými zdroji v důsledku snižují kvalitu software.

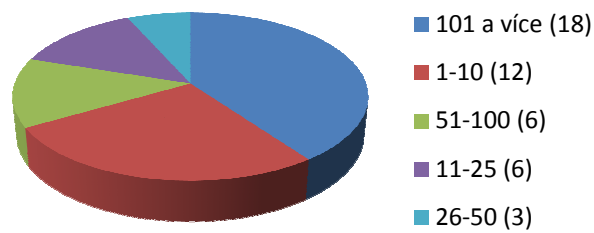
Tato hypotéza nám měla pomoci odhalit, zda dochází k negativním dopadům na kvalitu dodávané aplikace, pokud daný projekt disponuje omezenými zdroji, jako jsou například rozpočet, pracovní síla nebo čas.

Klíčové otázky: 10, 11, 12, 14, 15, 16.

4.3 Vyhodnocení dotazníku

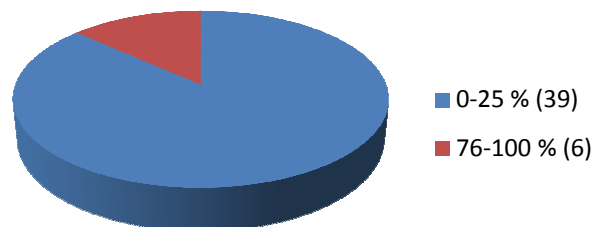
Dotazník úspěšně vyplnilo v daném časovém rozmezí 45 ze 78 oslovených firem. Vzhledem k tomu, že se jednalo o anonymní dotazník, nebyli jsme schopni určit, o které konkrétní firmy se jednalo. Výsledek dotazníku vč. grafů je uveden v příloze, v této kapitole se budeme zabývat nejdůležitějšími a nejzajímavějšími skutečnostmi z provedeného průzkumu.

Více než 1/3 firem zaměstnává 101 a více zaměstnanců, v absolutní hodnotě to je 18 společností. Na 2. místě se s absolutním počtem 12 umístilo rozmezí s počtem zaměstnanců 1-10.



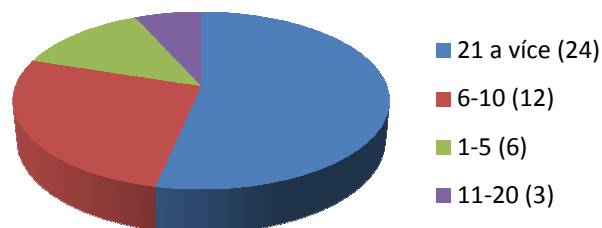
Graf 1. Počet zaměstnanců ve společnosti

Objem zahraničního kapitálu se u většiny oslovených firem pohybuje v rozmezí 0-25 %, z čehož je možné usuzovat, že se jedná o ryze české firmy.



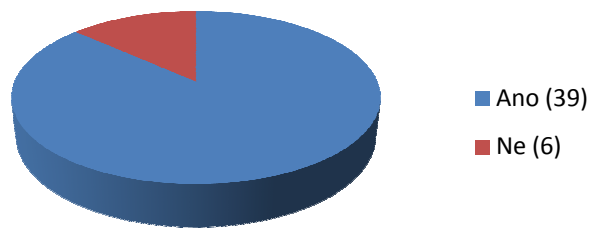
Graf 2. Objem zahraničního kapitálu ve společnosti

Dalším zajímavým zjištěním je, že nadpoloviční většina firem zpracovává 21 a více projektů za rok. Pravděpodobně se jedná o projekty menšího rozsahu s nižší složitostí.



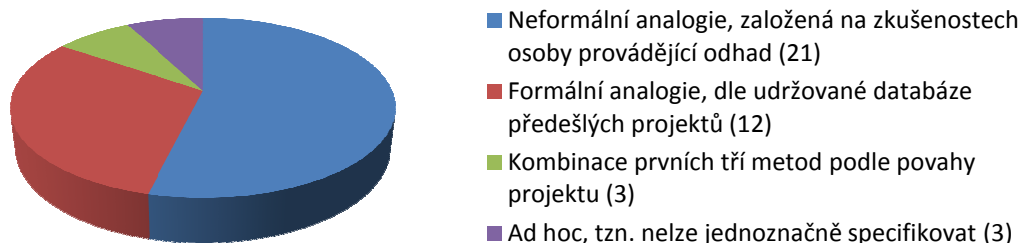
Graf 3. Počet zpracovávaných projektů za rok

Téměř 90 % (39) respondentů odpovědělo, že používá metody pro odhadování pracnosti software.



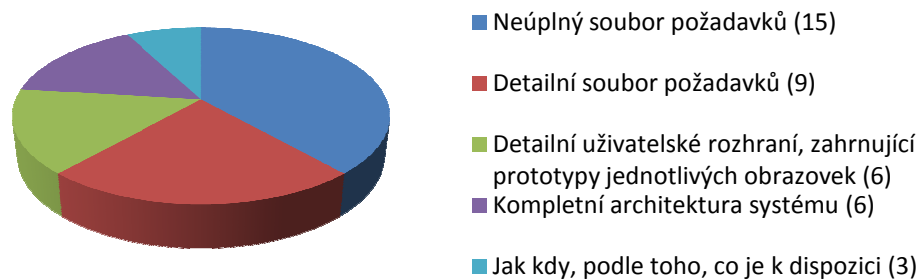
Graf 4. Počet firem používajících metody pro odhadování pracovního zatížení

U 90 % (39) firem, které používají metody pro odhadování pracovního zatížení, jsou nejčastěji aplikovány expertní odhady a analogie na základě minulých projektů. Ani jeden respondent neodpověděl, že by využíval nějaký formální matematický model, jako je například COCOMO. U šesti respondentů, kteří odpověděli, že nepoužívají žádnou metodu pro odhad pracovního zatížení, jsme se ptali, na základě čeho tedy stanovují náklady projektů například pro cenové nabídky. Objevily se zde pouze dvě odpovědi a to „kvalifikovaný odhad“ a „podle zkušeností“. Je vidět, že některé společnosti si neuvědomují, že se jedná o neformální metody, na základě kterých provádí odhady.



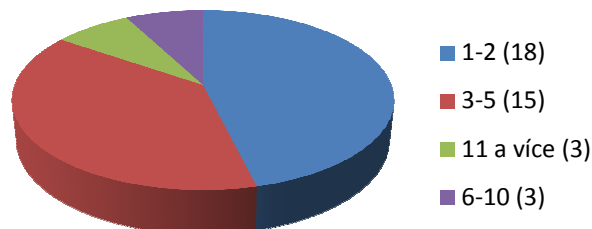
Graf 5. Aplikované metody pro odhad pracovního zatížení

Téměř polovina firem ze zmíněných 39, které provádějí odhady, odpověděla, že disponuje pouze neúplnými požadavky od zadavatele. Minimum respondentů má k dispozici detailní specifikaci požadavků nebo popis kompletní architektury budoucího systému.



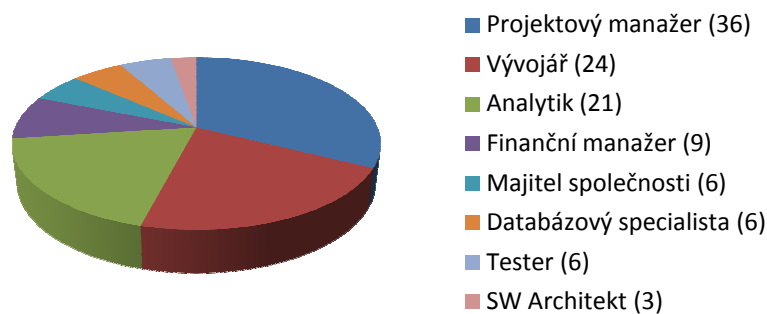
Graf 6. Druhy informací pro provedení odhadu

Na základě průzkumu bylo zjištěno, že do odhadu je zapojeno minimum osob ve společnosti, naprostá většina odpověděla v rozmezí 1-5 lidí. Je možné usuzovat, že se jedná o projektového manažera a několik dalších vedoucích pozic.



Graf 7. Počet osob zapojených do odhadů pracnosti

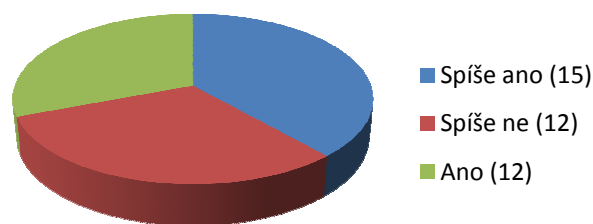
Přesnou odpověď jsme našli v další otázce. Zjišťovali jsme, jaké jsou pozice osob zapojených do odhadů. Bylo možné vybrat více z nabízených možností. Nebylo žádným překvapením, že projektový manažer figuroval až na tři společnosti v každé odpovědi. Na dalších dvou pozicích se s mírným odstupem umístil vývojář a analytik. Finanční manažer byl do tvorby odhadů zapojen v minimálním rozsahu.



Graf 8. Pozice osob provádějící odhady

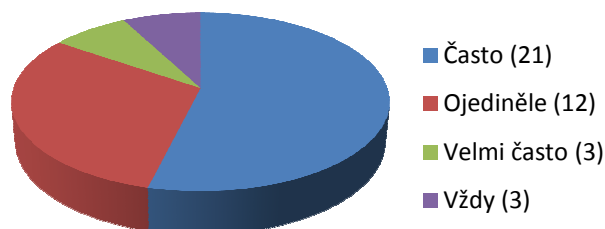
2/3 respondentů opovědělo „ano“ nebo „spíše ano“ na otázku, zda jsou během provádění odhadů omezení přidělenými zdroji (lidé, čas nebo peníze). Žádná ze společností neodpověděla, že by měla neomezené zdroje.

S tím souvisí i další zjištěná skutečnost, že většina firem vyplnila, že nebývá nucena ke snížení odhadovaných nákladů z důvodu toho, že by výsledná cenová nabídka nebyla konkurenceschopná. Zde můžeme pouze usuzovat, že finální cena prezentovaná zákazníkovi má dostatečnou rezervu oproti odhadovaným nákladům nebo se firma nepouští do zakázek pod cenou.



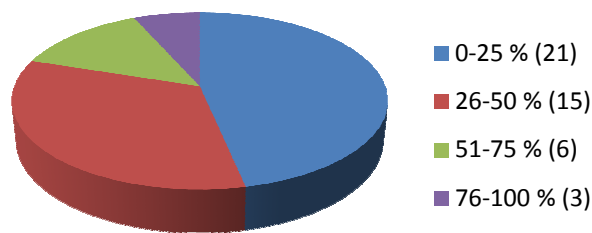
Graf 9. Počet firem omezených dostupnými zdroji

V další otázce odpověděla polovina respondentů, že projektový a finanční manažer se často dostávají do rozepří ohledně odhadované pracovní síly. Množina firem je téměř totožná s tou, která odpověděla, že je během odhadů omezena přidělenými zdroji.



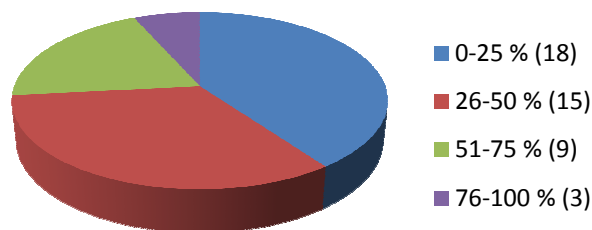
Graf 10. Konfrontace projektového a finančního manažera

3/4 dotazovaných společností nemá žádné nebo menší problémy s dodáním projektů včas. Aplikace metod pro odhadování pracovní síly neměla na tuto odpověď vůbec žádný vliv. Paradoxně společnosti, které nepoužívají metody pro odhadování pracovní síly, jsou zároveň schopné dodat software včas.



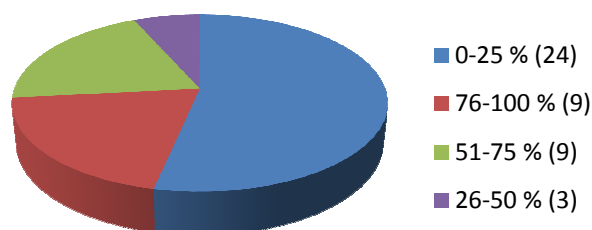
Graf 11. Počet projektů nedodaných včas

O trochu horší výsledek je v případě překročení počátečního rozpočtu. Sice 18 respondentů odpovědělo, že nemá žádné nebo minimální problémy v tomto směru. Problémy s rozpočtem má u 26-50 % projektů 15 firem a u 51-75 % projektů 9 firem. Opět firmy nepoužívající metody pro odhady pracnosti vyplnili spíše tu odpověď, že u nich k překročení počátečního rozpočtu nedochází.



Graf 12. Počet projektů s překročeným rozpočtem

Nadpoloviční většina odpověděla, že v některých případech je nutné vzdát se zavedených procesů, tzn. snížit kvalitu produktu, aby byl dodržen rozpočet nebo zpracován projekt včas.



Graf 13. Počet projektů, u kterých dochází k tzv. snížení kvality

4.4 Platnost nebo zamítnutí hypotéz

Více než 50 % společností aplikující metody pro odhadování pracnosti software, jsou schopné dodat software včas.

Klíčové otázky: 4, 6, 14, 15, 16.

		Nedodání softwaru včas (%)				
		0-25	26-50	51-75	76-100	celkem
Aplikace metod pro odhad pracnosti	ano	18	12	6	3	39
	ne	3	3	0	0	6
	celkem	21	15	6	3	45

Tab. 26. Vztah mezi aplikací metod pro odhad pracnosti a nedodáním softwaru včas

Z tabulky (Tab. 26) je viditelné, že přestože firmy aplikují metody pro odhady pracností, dochází v některých případech k časovému skluzu. Pokud budeme vnímat hodnotu 0-25 jako, že byl software dodán včas nebo smírným zpožděním a sloupce 26-50, 51-75 a 76-100 jako zpoždění, tak přibližně 46 % společností aplikující metody je schopné dodat software včas. U zbývajících 54 % se v jejich projektech objevuje nezanedbatelné zpoždění.

V tomto případě hypotézu zamítáme, protože méně než 50 % firem je schopných dodat software včas, přestože používá metody pro odhadování pracnosti.

Odhady pracnosti omezené dostupnými zdroji v důsledku snižují kvalitu software.

Klíčové otázky: 10, 11, 12, 14, 15, 16.

		Snižování kvality software (%)				
		0-25	26-50	51-75	76-100	celkem
Omezení dostupnými zdroji	ano	6	2	2	2	12
	spíše ano	9	0	4	2	15
	spíše ne	10	0	0	2	12
	ne	0	0	0	0	0
	celkem	25	2	6	6	39

Tab. 27. Vztah mezi omezenými dostupnými zdroji a snížením kvality software

Sloupce 26-50, 51-75 a 76-100 budeme brát jako snížení kvality software. Celkem tedy 12 respondentů odpovědělo, že disponuje omezenými zdroji (odpovědi „ano“ a „spíše ano“) a jsou donuceni ke snížení kvality software. Na druhé straně pouze 2 respondenti odpověděli, že nejsou omezení dostupnými zdroji (odpovědi „spíše ne“ a „ne“) a přesto

musí snížit kvalitu produktu. Lépe viditelný je zjištěný rozdíl v relativních číslech, kdy 44 % firem s omezenými zdroji musí snížit kvalitu software a pouze 17 % firem musí přistoupit ke stejnému kroku, pokud jsou jejich kapacity mírně omezené nebo neomezené vůbec.

V tomto případě hypotézu potvrzujeme a ponecháváme v platnosti, protože 44 % firem odpovědělo, že jejich příčinou pro snížení kvality software je omezení dostupnými zdroji. Na druhé straně jenom 17 % respondentů musí udělat stejná opatření, pokud jsou jejich zdroje téměř neomezené. Viditelný je více než dvojnásobný rozdíl.

4.5 Současný stav v softwarových firmách

V této kapitole bychom chtěli přispět vlastními zkušenostmi s aplikací metod pro odhadování pracnosti software ve společnostech, v kterých jsme působili nebo působíme. Nebude se jednat pouze o naše postřehy, připomínky a věcné podněty, ale i našich spolupracovníků, kolegů a přátel.

Shodli jsme se, že firmy je možné rozdělit na několik skupin. Skupiny vznikly na základě velikosti firmy, a zda se jedná o českou nebo zahraniční společnost.

V menší české společnosti jakékoliv odhady týkající se pracnosti zpracovávají řadový programátoři, uplatňuje se tu prostá analogie mezi aktuálně zpracovávanými projekty a projekty v minulosti a zkušenost jednotlivých členů týmu. Nejsou udržovány žádné znalostní báze minulých projektů, kromě zdrojových kódů, takže odhad čistě závisí na dobré paměti. Na druhou stranu je toto chování pochopitelné, jedná se obvykle o projekty menšího rozsahu, kdy se uplatňují agilní metody vývoje, a jakákoliv aplikace složitějších metod by dělala tento způsob tvorby produktu neflexibilní vzhledem k zákazníkovi.

Model české firmy většího rozsahu se nevidí příliš často. Vstup zahraničního partnera do prosperující a rostoucí české firmy evidujeme jako záchranu této společnosti před vlastní záhubou. V několika desítkách zaměstnanců dokáže společnost fungovat, všichni na sebe tzv. vidí, většina projektů se řeší operativně, stejně tak i jejich plánování a odhady. Problém nastává v situaci, kdy si majitelé myslí, že příjmou dvojnásobek nebo trojnásobek zaměstnanců, především programátorů a zisk firmy se zdvojnásobí nebo ztrojnásobí. Opak je ale pravdou, pokud nejsou učiněny kroky pro řízení takového počtu osob. To co fungovalo doteď, logicky přestává fungovat. Společnosti obvykle rostou za účelem toho, aby získali zakázky většího rozsahu, tyto projekty jsou ale i podstatně složitější, než

s kterými se setkali doposud. Je nutné tedy vždy propracovat plánování, řízení, odhady pracnosti atd. v takto rostoucích firmách. Vstup zahraničního kapitálu vidíme jako eliminaci nebo úplné odstranění pozdějšího nepříjemného probuzení do reality.

V pobočce většího rozsahu zahraniční společnosti působící u nás se spíše už jen odráží dobře fungující systém v její mateřské základně. Vstup zahraniční firmy na český trh s pracovní silou vyžaduje jasně definovaná pravidla, pravomoci a dobře fungující model. U společností zabývajících se vývojem firemních systémů jako např. ERP nebo CRM je již v průběhu jednání přítomen technický pracovník nebo projektový manažer investora, který společně s dodavatelem odhaduje pracnosti a tím pádem i termíny budoucího dodání. Na druhé straně pokud firma produkuje krabicový software, tak termín vydání a minimální chybovost jsou zde dvojnásobně důležité. U krabicového software se navíc velmi často vyhnete kontaktu se zákazníkem, pouze mu nasloucháte formou diskuzí, e-mailů nebo telefonů a ve finále zapracujete to, po čem nejvíc touží. I v těchto případech jsme se setkali pouze s expertními odhady a analogií s minulými projekty.

V zahraniční pobočce menšího rozsahu funguje podobný systém jako v předešlém odstavci.

ZÁVĚR

Cílem diplomové práce bylo zhodnotit problematiku odhadu pracnosti softwarových projektů. Zabývali jsme se dvěma dílčími cíly a to komparací metod Use Case Points a COCOMO II a provedení průzkumu týkající se aplikace metod pro odhad pracnosti mezi softwarovými firmami na základě anonymního dotazníku.

Srovnání metod bylo provedeno na reálném projektu, jež se v návrhu skládá z 41 use case, komunikuje se 4 aktory a tvoří jej přes 6000 řádků kódu. Komponenta aplikace běží jako Windows Service na serverové části systému. Zajímavé bylo, že rozdíl mezi nejkratším a nejdelším odhadem byl téměř 9 člověkoměsíců. Obecně můžeme říci, že metoda COCOMO II nám generovala spíše příznivěji vypadající odhady. Při použití doporučených parametrů v dané metodě se odhad pracnosti pro projekt v takovém rozsahu vyšplhal na 22 člověkoměsíců. Pokud jsme zapojili nákladové faktory a faktory prostředí, tak odhady ještě o 3-4 člověkoměsíce vzrostly. Metoda Use Case Points se naopak vyznačovala nejvyšší odhadovanou pracností. Téměř 31 měsíců při práci jednoho člověka by mělo trvat vytvoření této komponenty. Už v průběhu analýzy jsme se přesvědčily, že v případě COCOMO II jsou korekční faktory významným hybatelem výsledné hodnoty celkového odhadu. Opravdu mírné nuance způsobují výkyvy několik měsíců jedním nebo druhým směrem. Odhady metodou Use Case Points nejvíce ovlivňuje samotný návrh aplikace, z kterého vychází. To znamená, že v průběhu analýzy by se měly dodržovat stanovená pravidla, aby odhady jednotlivých projektů byly co nejpřesnější. Nejsme schopni z těchto dvou postupů vybrat jeden, který by byl vhodnější než ten druhý, každá z metod staví na jiném základě, což jistě ovlivní počáteční výběr. Naším doporučením je vybrat si jednu z metod a tu v průběhu času přizpůsobit vlastním potřebám na několika zpracovávaných projektech.

K vytvoření dotazníku, který se zabývá odhadem pracnosti softwarových projektů, nás vedla snaha o zmapování současného stavu v jednotlivých společnostech. Probíhající ekonomická krize snižuje rozpočty vyhrazené pro informační technologie, proto i dodavatelé těchto řešení musí odhadovat a hospodařit co nejefektivněji. Vybrali jsme anonymní formu dotazníku, abychom dosáhli co největší zpětné vazby od respondentů. Výzkum probíhal přibližně 11 dní a z oslovených 78 společností nám odpovědělo 45.

Přibližně třetina firem zaměstnává 100 a více zaměstnanců a jedná se spíše o ryze české firmy nebo s maximálně 25 % zahraničního kapitálu. Nadpoloviční většina všech

dotázaných zpracovává 21 a více projektů za rok a téměř 90 % aplikuje metody pro odhady pracnosti. Tyto společnosti nejčastěji používají expertní odhady a analogii na základě minulých projektů. Ani jeden z respondentů neodpověděl, že by využíval nějaký parametrický matematický model. Objevili se i odpovědi, že firma nepoužívá žádnou metodu pro odhad pracnosti, v kontrolní otázce jsme se však dozvěděli, že aplikují neformální přístupy při tvorbě odhadu.

Polovina všech firem odpověděla, že nemá k dispozici detailní požadavky zadavatele. Dalším zjištěním bylo, že do odhadu je zapojeno pouze několik zainteresovaných osob, které jsou zároveň omezeny dostupnými zdroji. Finanční a projektový manažer se v polovině případů dostávají do vzájemné rozepře a i to může být důvodem, že málo firmám se stává, že by jejich cenová nabídka nebyla konkurenceschopná.

I když společnosti neaplikují žádné formální metody pro odhady pracnosti, tak se to nijak negativně neprojevuje na jejich schopnosti dodat software včas. Podobná situace nastává i u překročení počátečního rozpočtu.

Na základě provedeného výzkumu jsme zamítli hypotézu „Více než 50 % společností aplikující metody pro odhadování pracnosti software, jsou schopné dodat software včas.“, protože pouze 46 % společností je schopných splnit slíbené termíny.

Druhou definovanou hypotézu „Odhady pracnosti omezené dostupnými zdroji v důsledku snižují kvalitu software.“ jsme ponechali v platnosti, protože 44 % firem s omezenými zdroji musí snížit kvalitu software a pouze 17 % firem musí přistoupit ke stejnému kroku, pokud jsou jejich kapacity mírně omezené nebo neomezené vůbec.

ZÁVĚR V ANGLIČTINĚ

The aim of this thesis was to evaluate the issue of time consumption estimate software projects. We deal with two sub-goals a comparison of methods Use Case Points and COCOMO II and the survey on the application of methods for estimating the labor input among software companies on an anonymous questionnaire.

Comparison of methods have been done on real projects that the proposal consists of 41 use cases, communicating with 4 actuators and comprises over 6000 lines of code. Component of the application runs as a Windows Service on the server component. Interestingly, the difference between the shortest and longest estimated person-months was almost 9. Generally speaking, the method of COCOMO II generated more favorable estimates. When using the recommended parameters in the method of estimation of time consumption for the project to the extent to 22 person-months. If we involved cost factors and environmental factors, such estimates have increased by 3-4 person-months. Use Case Points method to estimate the contrary, characterized by high labor input. Nearly 31 months work of one man would take the creation of this component. Already during the analysis, we are convinced that in the case of COCOMO II is an important driver of correction factors resulting value of the total estimate. Indeed a slight nuance of the variations for several months in one direction or another. Estimates by Use Case Points, most influences the application design, from which it is based. This means that the analysis should follow established rules, estimates of individual projects to be as accurate as possible. We are not able of the two procedures to choose one that would be better than the other, each method is based on a different basis, which will certainly affect the initial choice. Our recommendation is to choose one of the methods and over time to adapt to your needs on a number of projects processed.

To create a questionnaire that deals with the estimated time consumption of software projects, the led effort to map the current state of the individual companies. The ongoing economic crisis has reduced budgets for information technology, so the suppliers of these solutions to predict and to manage effectively. We chose the form of an anonymous questionnaire in order to achieve as much feedback from respondents. The research was conducted approximately 11 days and from 78 interviewed companies we have got feedback from 45.

About a third of companies employing 100 or more employees are rather a purely Czech company or has less than 25% foreign capital. More than half of all respondents handles 21 or more projects per year and almost 90% of the applied methods for estimating time and efforts. These companies often use an analogy and expert estimates based on past projects. None of the respondents answered that they used a parametric mathematical model. Emerged as the responses that the company does not use any method for estimating the labor input in the control issue, we have learned that informal approaches are applied in making the estimate.

Half of all firms said that it does not have the detailed requirements of the contracting authority. Another finding was that the estimates involve only a few interested persons, which are also limited available resources. Financial and project manager in the receiving half of the mutual quarrels, and even that may be the reason that few companies have become, that their bid was not competitive.

Even when companies do not apply any formal method for estimating labor input, so it does not show in any way negatively on their ability to deliver software on time. A similar situation also occurs in excess of the initial budget.

Based upon our research, we reject the hypothesis "More than 50% by applying methods for estimating the labor input software, is able to deliver software on time.", because only 46% of companies are able to meet promised deadlines.

Another defined a hypothesis "Estimates of the limited resources available time consumption due to lower quality software." we have kept in force since 44% of companies with limited resources to reduce the quality of software and only 17% of companies must undertake to follow suit, if their capacity is slightly limited or unlimited.

SEZNAM POUŽITÉ LITERATURY

- [1] HARDING ROBERTS, Mike. *I.T. Project Management Training Course - 3 days* [online]. c2011 [cit. 2011-02-05]. Chapter 4 - Project Definition. Dostupné z WWW: <<http://www.hraconsulting-ltd.co.uk/project-management-book-0401.htm>>.
- [2] *Wikipedia, the free encyclopedia* [online]. 15.1.2011 [cit. 2011-02-09]. Software development methodology. Dostupné z WWW: <http://en.wikipedia.org/wiki/Software_development_methodology>.
- [3] K. JAYASWAL, Bijay; C. PATTON, Peter. *Design for Trustworthy Software: Tools, Techniques, and Methodology of Developing Robust Software*. USA : Prentice Hall, 2006. Software Development Methodology Today, s. 840. ISBN 978-0-13-187250-9.
- [4] *Wikipedia, the free encyclopedia* [online]. 12.12.2010 [cit. 2011-02-10]. Waterfall model. Dostupné z WWW: <http://en.wikipedia.org/wiki/Waterfall_model>.
- [5] *Wikipedia, the free encyclopedia* [online]. 29.1.2011 [cit. 2011-02-13]. Software prototyping. Dostupné z WWW: <http://en.wikipedia.org/wiki/Software_prototyping>.
- [6] *Wikipedia, the free encyclopedia* [online]. 2.2.2011 [cit. 2011-02-15]. Spiral model. Dostupné z WWW: <http://en.wikipedia.org/wiki/Spiral_model>.
- [7] *Wikipedia, the free encyclopedia* [online]. 15.1.2011 [cit. 2011-02-17]. Iterative and incremental development. Dostupné z WWW: <http://en.wikipedia.org/wiki/Iterative_and_incremental_development>.
- [8] APPELO, Jurgen. *NOOP.NL* [online]. 18.7.2008 [cit. 2011-04-21]. The Definitive List of Software Development Methodologies. Dostupné z WWW: <<http://www.noop.nl/2008/07/the-definitive-list-of-software-development-methodologies.html>>.
- [9] KADLEC, Václav. *Živě.cz* [online]. 5.8.2003 [cit. 2011-02-21]. Rational Unified Process: základní pojmy. Dostupné z WWW: <<http://www.zive.cz/clanky/rational-unified-process-zakladni-pojmy/sc-3-a-113011/default.aspx>>.
- [10] *Wikipedia, the free encyclopedia* [online]. c2011 [cit. 2011-02-24]. Scrum (development). Dostupné z WWW: <[http://en.wikipedia.org/wiki/Scrum_\(development\)](http://en.wikipedia.org/wiki/Scrum_(development))>.
- [11] *VERSIONONE Agile Made Easier* [online]. 8.1.2011 [cit. 2011-02-24]. Agile Methodologies. Dostupné z WWW: <<http://www.versionone.com/Agile101/Methodologies.asp>>.

- [12] *Wikipedia, the free encyclopedia* [online]. c2011 [cit. 2011-02-26]. Lean software development. Dostupné z WWW: <http://en.wikipedia.org/wiki/Lean_software_development>.
- [13] *CodeBetter.Com* [online]. 2.2.2005 [cit. 2011-02-26]. Lean Software Development Overview. Dostupné z WWW: <<http://codebetter.com/darrellnorton/2005/02/02/lean-software-development-overview/>>.
- [14] *Agile Modeling* [online]. c2009 [cit. 2011-02-28]. Feature Driven Development (FDD) and Agile Modeling. Dostupné z WWW: <<http://www.agilemodeling.com/essays/fdd.htm>>.
- [15] *Wikipedia, the free encyclopedia* [online]. c2011 [cit. 2011-03-01]. Extreme Programming. Dostupné z WWW: <http://en.wikipedia.org/wiki/Extreme_Programming>.
- [16] *Wikipedia, the free encyclopedia* [online]. c2011 [cit. 2011-03-01]. Test-driven development. Dostupné z WWW: <http://en.wikipedia.org/wiki/Test-driven_development>.
- [17] BOUCHNER, Roman. *SystemOnLine* [online]. 2010 [cit. 2011-03-03]. Rizika při vývoji podnikového softwaru. Dostupné z WWW: <<http://www.systemonline.cz/erp/rizika-pri-vyvoji-podnikoveho-softwaru.htm>>.
- [18] PALETA, Petr. *Co programátory ve škole neučí : aneb Softwarové inženýrství v reálné praxi*. Praha : COMPUTER PRESS, 2003. 360 s. ISBN 9788025100738.
- [19] *Chris-Kimble.Com* [online]. c2011 [cit. 2011-03-04]. The Software Crisis. Dostupné z WWW: <http://www.chris-kimble.com/Courses/World_Med_MBA/Software_Crisis.html>.
- [20] *SystemOnLine* [online]. 2009-01-06 [cit. 2011-03-04]. Finanční krize v IT: nejvíce se bude šetřit na hardwaru. Dostupné z WWW: <<http://www.profit.cz/clanek/financni-krize-v-itnejvice-se-bude-setrit-na-hardwaru.aspx>>.
- [21] CHEMUTURI, Murali. *Chemuturi Consultants* [online]. c2011 [cit. 2011-03-06]. Analogy based Software Estimation. Dostupné z WWW: <<http://www.chemuturi.com/Analogy%20based%20Software%20Estimation.pdf>>.
- [22] RUSH, Christopher; ROY, Rajkumar. *Expert judgement in cost estimating: Modelling the reasoning process* [online]. United Kingdom : Cranfield University, 2006 [cit. 2011-03-08]. Dostupné z WWW: <<http://www.semgrid.net/Citation-Before-2006.1/expert-judgement-in-cost.pdf>>.

- [23] *Parametric Estimating Handbook* [online]. 2007 : International Society of Parametric Analysts, 2007 [cit. 2011-05-02]. Dostupné z WWW: <http://www.ispa-cost.org/ISPA_PE_Hdbk_4thED.pdf>.
- [24] STELLMAN, Andrew; GREENE, Jennifer. *Applied Software Project Management*. USA : O'Reilly Media, 2005. Estimation, s. 336. ISBN 978-0-596-00948-9.
- [25] J. ALEXANDER, Alvin. *Devdaily.com* [online]. 2009-02-03 [cit. 2011-03-12]. How to Determine Your Software Application Size Using Function Point Analysis. Dostupné z WWW: <<http://www.devdaily.com/FunctionPoints/FunctionPoints.shtml>>.
- [26] *The Code Project* [online]. 2005-03-22 [cit. 2011-03-15]. Project Estimation with Use Case Points. Dostupné z WWW: <<http://www.codeproject.com/KB/architecture/usecasep.aspx>>.
- [27] BOEHM, Barry. *COCOMO II Model Definition Manual* [online]. USA : University of Southern California, 2000 [cit. 2011-03-17]. Dostupné z WWW: <<http://sunset.usc.edu/research/COCOMOII/Docs/modelman.pdf>>.
- [28] Sedláčková, J.: *Cenové odhady softwarových projektů*. Masarykova univerzita v Brně, Fakulta informatiky, Brno, 2005. Diplomová práce.
- [29] LEUNG, Hareton; FAN, Zhang. *Software Cost Estimation* [online]. E Hong Kong : The Hong Kong Polytechnic University, 2001 [cit. 2011-03-17]. Dostupné z WWW: <<ftp://cs.pitt.edu/chang/handbook/42b.pdf>>.

SEZNAM OBRÁZKŮ

Obr. 1. Vodopádový model se zpětnou vazbou	13
Obr. 2. Spirálový model.....	14
Obr. 3. Iterační model v rámci modelu přírůstkového.....	14
Obr. 4. Iterační model v rámci RUP	15
Obr. 5. SCRUM	16
Obr. 6. Plánovací smyčka se zpětnou vazbou.....	18
Obr. 7. Test Driven Development.....	19
Obr. 8. Lineární a logaritmický průběh závislosti nákladů.....	33
Obr. 9. Náklady projektu	33

SEZNAM TABULEK

Tab. 1. Seznam minulých projektů	27
Tab. 2. Současný a vybraný projekt s hodnotou CWF	29
Tab. 3. ILFs stupně složitosti.....	41
Tab. 4. ILFs matice složitosti.....	41
Tab. 5. EIFs stupně složitosti.....	41
Tab. 6. EIFs matice složitosti.....	41
Tab. 7. EIs stupně složitosti	42
Tab. 8. EIs matice složitosti.....	42
Tab. 9. EOs stupně složitosti	43
Tab. 10. EOs matice složitosti	43
Tab. 11. EQs stupně složitosti	43
Tab. 12. EQs matice složitosti	43
Tab. 13. Technické faktory systému	45
Tab. 14. Environmentální faktory systému.....	46
Tab. 15. Rozdělení use case dle složitosti	47
Tab. 16. Rozdělení aktorů dle složitosti	47
Tab. 17. Nákladové faktory aplikace	50
Tab. 18. Scale Factors.....	50
Tab. 19. Technické faktory komponentu LBS.....	54
Tab. 20. Faktory prostředí pro komponentu LBS.....	55
Tab. 21. UUCW pro LBS	55
Tab. 22. UAW pro LBS	56
Tab. 23. Nákladové faktory komponenty LBS	57
Tab. 24. Scale Factors pro komponentu LBS	58
Tab. 25. Srovnání metod pro odhad pracnosti	58
Tab. 26. Vztah mezi aplikací metod pro odhad pracnosti a nedodáním softwaru včas.....	71
Tab. 27. Vztah mezi omezenými dostupnými zdroji a snížením kvality software.....	71

SEZNAM GRAFŮ

Graf 1. Počet zaměstnanců ve společnosti.....	66
Graf 2. Objem zahraničního kapitálu ve společnosti.....	66
Graf 3. Počet zpracovávaných projektů za rok.....	66
Graf 4. Počet firem používajících metody pro odhadování pracnosti	67
Graf 5. Aplikované metody pro odhad pracnosti.....	67
Graf 6. Druhy informací pro provedení odhadu	68
Graf 7. Počet osob zapojených do odhadů pracnosti	68
Graf 8. Pozice osob provádějící odhady	68
Graf 9. Počet firem omezených dostupnými zdroji	69
Graf 10. Konfrontace projektového a finančního manažera.....	69
Graf 11. Počet projektů nedodaných včas	70
Graf 12. Počet projektů s překročeným rozpočtem	70
Graf 13. Počet projektů, u kterých dochází k tzv. snížení kvality	70

SEZNAM PŘÍLOH

PŘÍLOHA P I: USE CASE DIAGRAM

