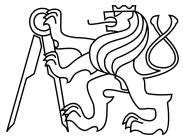
ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ



ZADÁNÍ DIPLOMOVÉ PRÁCE

Název:	Implementace nástrojů pro zlepšení procesu SW vývoje
Student:	Bc. Miroslav Štaffa
Vedoucí:	Ing. Michal Petřík
Studijní program:	Informatika
Studijní obor:	Webové a softwarové inženýrství
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	Do konce letního semestru 2018/19

Pokyny pro vypracování

1. Seznamte se se současným procesem vývoje SW u zadavatele (Profinit) z pohledu iniciálního sběru požadavků, tvorby odhadů, měření v rámci projektu i vyhodnocení v rámci tvorby historie projektu.
2. Proveďte analýzu a návrhy modifikace nástroje pro tvorbu odhadů a sběr metrik (bude se navazovat na obhájené Bc. práce).
3. Proveďte analýzu a návrh softwarové architektury pro podporu procesu vývoje (sběr, měření, issue tracking, historická data).
4. Navrhněte a zvolte vhodné technologie pro části, které z pohledu navržené SW architektury chybí, implementujte základní funkcionality.
5. K nově vytvářeným částem dodejte dokumentaci (uživatelská, administrátorská příručka, ...) a otestujte výstupy (unit testy jsou minimum).
6. Zhodnoťte výhody a nevýhody vašeho řešení vzhledem k možnosti budoucího rozvoje.
7. Prezentujte navržený proces vývoje z pohledu celého cyklu.

Seznam odborné literatury

Dodá vedoucí práce.

Ing. Michal Valenta, Ph.D.
vedoucí katedry

doc. RNDr. Ing. Marcel Jiřina, Ph.D.
děkan

V Praze dne 12. října 2017

FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

Diplomová práce

Implementace nástrojů pro zlepšení procesu SW vývoje

Bc. Miroslav Štaffa

Katedra softwarového inženýrství
Vedoucí práce: Ing. Michal Petřík

2. května 2018

Poděkování

Chtěl bych poděkovat Ing. Michalu Petříkovi za odborné vedení, jeho znalosti a ochotu, které mi velice pomohli při tvorbě této diplomové práce. Dále bych chtěl poděkovat společnosti Profinit EU, s.r.o. za umožnění tvorby této práce a poskytnutí času a prostředků nutných k její realizaci. V neposlední řadě bych rád poděkoval své rodině za veškerou podporu v průběhu celých mých studií.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval(a) samostatně a že jsem uvedl(a) veškeré použité informační zdroje v souladu s Metodickým pokynem o etické přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. V souladu s ust. § 46 odst. 6 tohoto zákona tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou, a veškeré jejich dokumentace (dále souhrnně jen „Dílo“), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla, a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené. Každá osoba, která využije výše uvedenou licenci, se však zavazuje udělit ke každému dílu, které vznikne (byť jen zčásti) na základě Díla, úpravou Díla, spojením Díla s jiným dílem, zařazením Díla do díla souborného či zpracováním Díla (včetně překladu), licenci alespoň ve výše uvedeném rozsahu a zároveň zpřístupnit zdrojový kód takového díla alespoň srovnatelným způsobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

V Praze dne 2. května 2018 .....

Odkaz na tuto práci

Štaffa, Miroslav. *Implementace nástrojů pro zlepšení procesu SW vývoje*. Diplomová práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2018.

Abstrakt

Tato práce se zabývá analýzou a úpravami procesu vývoje software společnosti Profinit EU, s.r.o. Práce navrhuje ideální stav tohoto procesu, který zajistí efektivní průběh celého životního cyklu projektu. Z provedené analýzy současného procesu jsou navrženy úpravy postupů i používaných nástrojů takovým způsobem, aby došlo k co nejbližšímu naplnění cílového stavu. Část navržených úprav a požadavků pak byla v rámci práce realizována, případně byla provedena příprava k jejich budoucí realizaci.

Klíčová slova Softwarový proces, Profinit, odhad, sběr požadavků, metriky, issue tracking, projektové řízení

Abstract

This thesis deals with analysis and alterations of software development process used by Profinit EU, s.r.o. The thesis proposes the ideal state of this process to ensure the efficient course of action during the whole project lifecycle. Based on the analysis of the current process modifications of the procedures and the tools used are proposed in such a way that the target state is reached as close as possible. Some of the proposed modifications and requirements were then implemented or prepared for their future implementation.

Keywords Software process, Profinit, estimate, requirement analysis, issue tracking, project management

Obsah

Seznam obrázků

Úvod

Profinit EU, s.r.o. patří k předním českým softwarovým společnostem působícím v oboru zakázkového vývoje, enterprise integrací a výstavby datových skladů. Na českém i zahraničním trhu se pohybuje již od roku 1998, kdy byla založena. Díky tomu se může chlubit realizací množství různorodých projektů v různých oblastech.

Profinit se zaměřuje převážně na klienty střední a větší velikosti. Od toho se odvíjí i většina realizovaných zakázek, která je převážně většího rozsahu (větší stovky MDs). S tím souvisí i zvýšený důraz na dodržování principů softwarového inženýrství a také schopnost přípravy kvalitních odhadů, jejichž tvorba u velkých projektů bývá náročnější a přesnost nižší.

K zajištění úspěchu na poli softwarového inženýrství pomáhá nemalým dílem kvalitní a dobře nastavený proces vývoje. Ten musí být zaměřený na co nejefektivnější využití dostupných prostředků a zároveň minimalizaci výskytu chyb ve vyvíjeném produktu tak, aby došlo k vývoji softwaru kvalitního, v požadovaném rozsahu a čase. Proces by měl také přispět k zaučování a kvalitnímu rozvoji nových i stávajících zaměstnanců.

Tato práce se snaží o konsolidaci a vylepšení stávajícího procesu softwarového vývoje společnosti Profinit tak, aby nejvíce usnadnil vývoj kvalitního softwaru, podílel se na spokojenosti zaměstnanců, a tím pádem i podporoval dobré jméno společnosti na českém i zahraničním trhu.

Kapitola 1

Cíl práce

Hlavním cílem této diplomové práce je, jak již bylo nastíněno v úvodní kapitole, zlepšení celého současného procesu softwarového vývoje ve společnosti Profinit. Práce se primárně zaměřuje na optimalizaci iniciálního sběru požadavků, tvorbu odhadů, měření v rámci realizace projektu, jeho následné vyhodnocení, uchovávání a zužitkování historie projektů.

Jednou z klíčových součástí softwarového procesu je odhadování pracnosti. Od kvality odhadu se odvíjí mnoho faktorů ovlivňujících úspěch projektu, a to ať už jde o jeho ziskovost, včasnou realizaci, spokojenost zákazníka apod. Odhad je také důležitý z hlediska plánování projektu, dle očekávané pracnosti lze například určit nutný počet vývojářů. Špatný odhad pak může mít za následek v lepším případě nadbytek zdrojů, v případě horším jejich nedostatek.

Přesný odhad však v praxi není jednoduchý a zdaleka není běžným standardem. Naopak, například studie [1], zaměřená na projekty menší velikosti, uvádí, že ze zkoumaných dat z roku 2012 došlo k úspěšnému dokončení projektu (v odhadovaném čase, za odhadovanou cenu, se všemi plánovanými funkcemi) pouze v 39 % případů. Zároveň lze z prezentovaných dat pozorovat nárůst úspěšnosti projektů (například v roce 2006, byla úspěšnost o 10 % nižší). Při omezení sledovaných projektů pouze na včasné dokončení jsou výsledky ještě o něco horší. V roce 2012 bylo pouze 26 % projektů dokončeno v plánovaném čase, opět však o 10 % více než v roce 2004.

Tvorbu kvalitních odhadů lze podpořit jednak zvolením vhodné metodiky, kterých existuje celá řada, dále je však nutné implementovat tuto metodiku do procesu vývoje a patřičnými nástroji podpořit její využití. Usnadnění podpory používané metodiky a zefektivnění jejího dodržování je jedním z cílů této práce.

V první řadě je nutné navrhnout cílový ideální stav vývojového procesu, kterého by mělo být dosaženo. Takto navržený proces by měl umožnit co nejefektivnější postup při realizaci projektu, minimalizovat rizika vzniku chyb (nejen při tvorbě odhadu, ale i při samotné realizaci projektu) a jejich případných dopadů na průběh a výsledek projektu.

Po zadefinování cílového stavu procesu je nutné analyzovat současný stav procesu používaného ve společnosti Profinit. Analyzován bude nejen samotný proces, ale i nástroje určené k jeho podpoře, budou identifikována jejich slabá a silná místa, a dojde k posouzení, zda vyhovují použití v rámci cílové podoby procesu.

Na základě této analýzy budou navrženy úpravy a možná vylepšení jak samotného procesu, tak používaných nástrojů, popřípadě i návrh a implementace nástrojů nových takovým způsobem, aby došlo k co největšímu přiblížení procesu cílovému stavu. Nelze však očekávat radikální změnu samotného procesu vývoje, který je ověřený časem a celou řadou projektů, které společnost Profinit realizuje a v minulosti realizovala. Změny musí být pokud možno neinvazivní a pokud možno evoluční, nikoliv revoluční. Nesmí dojít ke zhoršení současného stavu. Pokud však dojde, musí se jednat pouze o krátkodobý jev. Na druhou stranu lze předpokládat, že dojde k množství návrhů na potenciální úpravy podpůrných nástrojů, ať už menšího či většího rozsahu.

Z návrhu poté budou vybrány vhodné úpravy určené k realizaci v rámci této práce. Tyto budou implementovány se zachováním standardního postupu softwarového vývoje, tzn. po přesnější analýze a návrhu dojde k jejich implementaci, finální produkty budou řádně otestovány (pokrytí minimálně použitím unit testů) a následně aplikovány do praxe s využitím pilotního provozu na vybraných projektech.

Autorovi práce, jakožto zaměstnanci společnosti Profinit, se naskýtá výhoda v podobě každodenní spolupráce s týmy společnosti Profinit. Tato skutečnost velkou měrou podporuje tvorbu této práce. V první řadě je autor seznámen se stávajícím procesem vývoje a jeho nedostatky. Zároveň může diskutovat navrhované úpravy s ostatními zaměstnanci, kterých se vývojový proces bezprostředně týká, a obdržet užitečnou a rychlou zpětnou vazbu již ve stádiu návrhu. Dále je mu umožněno vytvářené úpravy v průběhu realizace testovat na dlouhodobě běžících projektech společnosti Profinit, a tak pozorovat pozitivní, či negativní dopady. Díky tomu je pak možné učinit rozhodnutí, zda bude vhodné danou úpravu zavést v rámci procesu napříč společností, anebo danou myšlenku opustit, případně upravit.

Kapitola 2

Cílový stav

V této kapitole bude navržen požadovaný cílový stav softwarového procesu vývoje, zejména po stránce tvorby odhadů, jejich dodržování a evidence historie projektů. Tato část se snaží o nastínění ideálního procesu, který začíná sběrem požadavků, pokračuje tvorbou odhadu a jeho využitím při realizaci, až po finální získání užitečných metrik z realizace s ohledem na projekt a jejich zanesení do historie projektů.

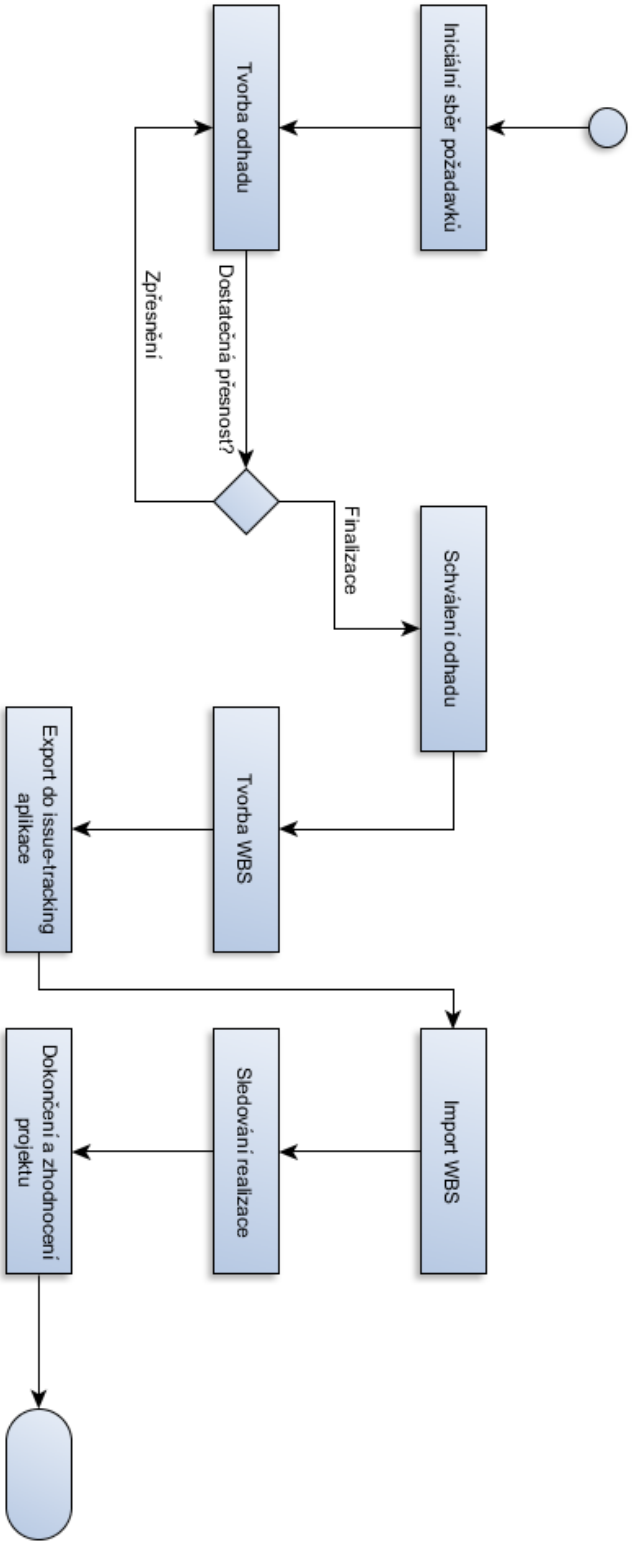
Finální návrh je zobrazen v grafu 2.1. Jedná se o optimistický průchod vhodný k ilustraci. Při tvorbě realistického modelu by jednotlivé aktivity měly vždy minimálně zpětnou šipku k návratu na předchozí fázi. Podrobněji bude průchod rozepsán v následujících odstavcích a v kapitole návrhu, kde jsou zároveň ilustrovány sféry působnosti aktérů a nástrojů tohoto procesu (4.1).

2.1 Iniciální sběr požadavků

Prvotní fází softwarového projektu vždy bývá sběr požadavků. Zákazník většinou zahájí potenciální vztah tvorbou formálního zadání nebo alternativně obecnější poptávkou. V případě domluvy však vždy dochází k postupnému zpřesňování požadavků. To s sebou nese množství komunikace mezi zákazníkem a dodavatelskou společností, výstupem bývá obvykle také četná dokumentace.

Prvním požadavkem na proces je tak správa evidence těchto požadavků a archivace komunikace. Bylo by určitě vhodné, kdyby obojí bylo uloženo na jediném místě, což usnadní celý proces identifikace požadavků, například díky snadnému dohledávání předchozí komunikace.

Pro účely tohoto požadavku se jeví být ideálním některý z dostupných issue-tracking nástrojů. Ten v první řadě zajistí snadné oddělení jednotlivých projektů nebo změnových řízení. Každá ucelená část projektu může být sdružována v rámci jednoho vytvořeného issue. Většina běžně používaných issue-tracking nástrojů podporuje ukládání souborů, což výše definovaný požadavek podporuje. Další běžnou, ne-li nejdůležitější součástí bývá možnost diskuse



Obrázek 2.1: Cílový proces

u každé dílčí úlohy. Tímto způsobem je možné vést diskusi se zákazníkem, který pak má také přístup k jednotlivým dotazům a připomínkám. Celé řešení je tak mnohem více transparentní, než kdyby se diskuse vedla například pomocí emailových zpráv.

Mezi nevýhody evidence v issue-tracking nástroji může patřit nepřehledná diskuse, která typicky vzniká u složitějších úloh. Pokud je v rámci jedné úlohy vedeno více diskusních linek, může dojít k opomenutí některých požadavků či připomínek. Tomuto problému lze předcházet včasným oddělením těchto linek do separátních issue. V případě evidence všech požadavků v jednom jediném dokumentu k takovému problému nedochází, nicméně nelze se v takovém případě vyhnout komunikaci emailem či telefonicky, která pak nemusí být nikde evidována.

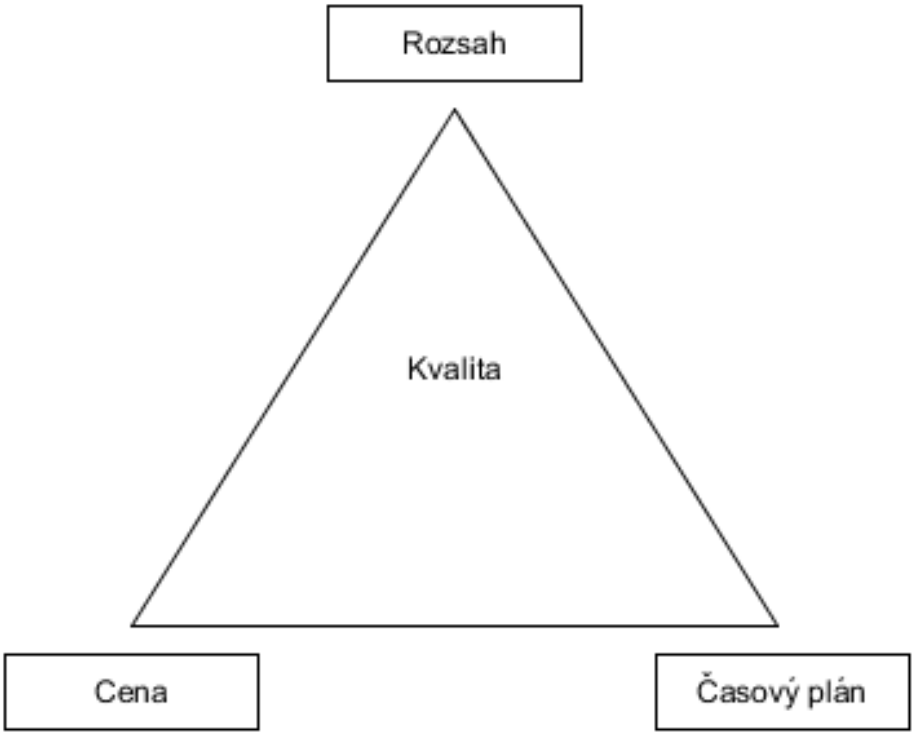
Dobrým kompromisem pak je komunikaci udržovat vždy na jediném místě v issue-tracking systému a zároveň u stejné úlohy evidovat důležitou dokumentaci. Zásadní pak je, aby tento postup dodržovali nejen analytici a vývojáři, ale i zákazník. V opačném případě může docházet k chaosu a z něho vycházejícím problémům.

2.2 Tvorba odhadu

Po iniciálním vydefinování požadavků a rámcového zjištění toho, co si vlastně zákazník přeje, přichází na řadu tvorba odhadu. Schopnost kvalitního odhadování patří k základním pilířům každé softwarové společnosti, která realizuje zakázkové projekty nebo je jiným způsobem závislá na plánování kapacit. Tzv. trojúhelník projektového řízení (2.2) (někdy také Iron triangle neboli Železný trojúhelník ([2]) uvádí, že zjednodušeně lze vyhodnotit úspěšnost projektu pohledem na tři jeho aspekty, kterými jsou cena realizace, časový plán projektu a jeho rozsah. Tyto tři stránky pak dohromady ovlivňují kvalitu výsledného díla. Zároveň platí, že jeden z těchto tří aspektů lze bez dopadů na výslednou kvalitu změnit na úkor ostatních dvou. Například pokud si zákazník vyžádá dodání projektu v bližším časovém horizontu, je nutné buďto snížit rozsah projektu ubráním funkcionalit nebo navýšit zdroje přidáním více vývojářů nebo jejich výměnou za zkušenější (i to má samozřejmě svá omezení a nemusí to tak nutně znamenat, že se opravdu podaří dodržet slíbený harmonogram). Pokud nebudou zvýšeny zdroje, ani snížen rozsah, s největší pravděpodobností se to podepíše na kvalitě projektu.

Odhadem je pak zvýšena flexibilita v tomto manévrovacím prostoru. Pokud je k dispozici dobrý odhad, lze přesněji určit, jakým způsobem a v jakém rozsahu upravit jednotlivé aspekty, aby projekt dosáhl požadované kvality.

Tvorba kvalitního odhadu s sebou přináší otázku: Jaký odhad je kvalitní? Zdroje se v názorech liší. [3] uvádí, že lze dosahovat 90% přesnosti, ale pouze na dobře řízených projektech. [4] tvrdí, že kvalitní odhadovací proces se bude přesností pohybovat v rozmezí 25 % v 75 % případů. Lze tedy usoudit, že nelze



Obrázek 2.2: Trojúhelník projektového managementu

očekávat 100% přesnost odhadů při použití sebelepší metodiky, což potvrzuje i [5], dle kterého se i nejpokročilejší, velice matematicky náročné odhadovací metodiky, pohybují kolem 90% přesnosti. Na druhou stranu většina softwarových společností se alespoň snaží vyhnout odhadům s vyšší než 100% chybou.

[5] dále uvádí, že většina chyb v odhadech vzniká díky jednomu či kombinaci následujících důvodů:

- Nepřesné informace o odhadovaném projektu.
- Nepřesné předpoklady o schopnostech týmu realizující projekt.
- Chaotičnost projektu, která zamezuje přesnému odhadování.
- Nepřesnosti vznikající přímo ze způsobu odhadování.

První tři oblasti jsou kontrolovány zejména projektovým řízením, kvalitní proces vývoje a podpůrné nástroje však mohou zlepšit prevenci chyb vznikajících z těchto důvodů. Chybám způsobeným nepřesností odhadování lze zamezit zvolením, implementací a dodržováním vhodné metodiky.

Tvorba přesného odhadu tedy není jednoduchá a je potřeba zajistit co nejvhodnější podmínky pro jeho kvalitní provedení. Z hlediska kvality je důležité zejména dodržování předepsaného postupu a metodiky a minimalizace chyb. Celkově by měla být snaha o co největší automatizaci procesu, kdy se uživatel může soustředit již pouze na samotný odhad a nemusí být zatěžován formalitami.

Zároveň je velkým přínosem, pokud jsou všechny odhady zálohované a sdílené napříč týmem popř. týmy. Každý odhad musí projít schvalovacím procesem, kdy dojde k jeho revizi jiným člověkem než je jeho původní autor. Díky sdílení se tento proces usnadní.

Další důležitou podmínkou je možnost odhady verzovat. Jednak pro usnadnění revizního procesu, dále pak kvůli potřebě mít možnost odhad postupně zpřesňovat na základě nových poznatků a nasbíraných požadavků. Není rozumné kvůli každému novému zpřesnění zakládat nový odhad. Mnohem lepším řešením je umožnit vytváření nových verzí. Každá verze by pak měla být doplněna o auditní informace, obsahující minimálně autora verze a čas poslední úpravy této verze. Velkým plusem pak je možnost jednotlivé odhady porovnávat, pokud možno na více úrovních. Srovnání souhrnných pracností slouží k porovnání dvou verzí odhadů z hlediska projektového řízení, na druhou stranu srovnání jednotlivých požadavků pak může pomoci při potřebě preciznějšího porovnání. Například za účelem revize nové verze odhadu, či její úpravy.

Při znalosti dostatečného množství požadavků pak vznikne finální verze odhadu. Ta je nejprve schválena zodpovědnou osobou ze strany vývojové společnosti, po jejích připomínkách a úpravách pak dojde k předání zákazníkovi. Granularita předávaného odhadu bývá nižší než odhad určený pro realizační tým. Každá předávaná verze by však měla minimálně obsahovat souhrn pracnosti, výčet okrajových podmínek a v neposlední řadě cenu realizace projektu.

Po schválení této finální verze odhadu lze zahájit práce na realizaci.

2.2.1 WBS

Od finální verze odhadu se následně odvíjí iniciální nastavení projektu po stránce dostupných zdrojů, času a rozsahu. Je potřeba mít nad využívanými zdroji kontrolu a zároveň dodržet stanovený časový harmonogram. V neposlední řadě je nutné realizovat dílo v požadovaném rozsahu. Zároveň je třeba být flexibilní, jelikož všechny tyto aspekty se mohou v průběhu projektu měnit. Pro usnadnění zajištění těchto požadavků pak slouží tzv. WBS.

WBS, neboli work breakdown structure, česky nejednoznačně přeloženo jako hierarchická struktura činností (pro účely této práce bude dále sloužit také pojem „rozpad“). Tvorba WBS určené pro účely tohoto procesu vychází z finálního odhadu a jedná se také svým způsobem o druh odhadu, avšak s určitými odlišnostmi. Jednak většinou musí disponovat větší granularitou a jednotlivé položky by měly být dále nedělitelné, tedy atomické. Dané by mělo

platit z toho důvodu, aby mohli být dílčí úlohy jednoduše přidělitelné jednotlivým členům týmu k realizaci. Zároveň musí platit, že WBS musí zahrnout kompletní rozsah projektu, nesmí nic chybět, ale ani přebývat. Tím pádem by i odhadované časy na jednotlivých úkolech měli dát v součtu odhadovanou pracnost.

Obsah WBS musí pak být jednoznačným způsobem prezentován členům týmu, kteří se mají podílet na realizaci tak, aby mohla probíhat co nejefektivnější spolupráce. Také by měl být umožněn monitoring projektu ze strany projektových vedoucích pro zajištění hladkého průběhu realizace, dodržení pracností a termínů, a dalších aspektů projektového řízení. K tomu opět nejlépe slouží některý z issue-tracking nástrojů.

Protože nelze očekávat existenci jediného nástroje, který bude spravovat odhady, WBS a zároveň mít funkcionalitu issue-tracking aplikace, tak nejspíše v tomto místě procesu bude docházet k nějaké transformaci a exportu dat WBS do dedikovaného issue-tracking nástroje. V závislosti na velikosti projektu může objem dat WBS nabírat významných velikostí. Je tedy vhodné použít takový issue-tracking systém, který umožňuje efektivní import/export těchto dat.

2.3 Kontrola průběhu realizace

Po importu WBS do issue-tracking aplikace lze zahájit samotnou realizaci. V jejím průběhu je nutné zajistit odpovídající nástroje pro podporu sledování průběžného dodržování pracnosti a harmonogramu. Je zapotřebí, aby osoba odpovědná za projekt mohla včas identifikovat potenciální problémy, například začne-li některý z úkolů neúměrně překračovat odhadovanou pracnost, a na tyto problémy reagovat, např. výpomocí zkušenějšího vývojáře nebo přepracováním designu. Existence přehledu, ve kterém jsou tato data agregována, umožní rozpoznat výskyt problémů tohoto typu ve větším měřítku, což může být známkou velmi nepřesného iniciálního odhadu, nevhodného designu nebo třeba špatně poskládaného realizačního týmu. V každém případě je ale nutný zásah tak, aby došlo k zastavení tohoto sestupného trendu, který může v těch nejhorších případech zapříčinit neúspěch projektu.

K dodržení pracností slouží ve většině issue-tracking systémů speciálně určené záznamy u každého úkolu. Většinou se objevuje pole typu „Odhadovaný čas“ a „Odpracovaný čas“. Odhadovaný čas obsahuje informaci o tom, kolik času bylo na realizaci daného úkolu odhadnuto. Odpracovaný čas pak říká, kolik času již bylo odpracováno, tzv. „spáleno“. U pole odhadovaného času pak vyvstává zajímavá, až filozofická, otázka. Lze očekávat, že čas odhadovaný na implementaci úkolu se bude v průběhu času měnit? Může dojít ke zpřesnění požadavku, změně původně zamýšleného designu, výskytu neočekávaného problému, možností je mnoho. Má pak v takovém případě být změna odhadovaného času reflektována i ve změně informace o odhadovaném čase

vedené u úkolu? Pro účely sledování a predikci průběhu projektu je určitě nezbytné udržovat odhadovaný čas aktuální. Na druhou stranu zde pak dochází ke ztrátě důležité informace potřebné pro zhodnocení průběhu projektu po jeho realizaci, kdy by již nedocházelo k porovnání s původně odhadovaným časem, ale jeho upravenou formou. Řešením tohoto problému může být udržování informace o obou odhadovaných časech. Tedy jeden atribut obsahuje původně odhadovaný čas, který vychází z pracnosti ve WBS, a zároveň je vytvořen atribut nový, kde je udržován aktuálně odhadovaný čas potřebný k vyřešení tohoto úkolu.

Tímto je zajištěn požadavek sledování průběžného dodržování pracnosti i harmonogramu. Vhodnou selekcí a agregací dat z issue-tracking systému lze pak jednoduše identifikovat vznikající problémy. Díky poli obsahujícímu původní odhad lze rychle poznat, u kterých úkolů dochází k překročení původního odhadu, a kde naopak dojde k časové úspoře. Zároveň se díky položce aktuálně odhadovaného času dá určit, v jakém časovém horizontu je možné očekávat dokončení úkolu. Je již pak na každém řídícím pracovníkovi, aby tyto možnosti využil při plánování projektu. Díky dostupnosti těchto dat je však daná úloha bezesporu snazší a efektivnější.

Nástroj, který takovéto agregace dat umožní, může být jednak komponentou issue-tracking systému (například ve formě reportů či jiných KPI), nebo se může jednat o nástroj separátní. Z hlediska projektového řízení a uchovávání historických dat je zajímavá možnost takováto data v rozumném formátu extrahovat a využít je pro další reporting.

2.3.1 Hierarchie úkolů

Mezi úkoly lze většinou vysledovat známky určité hierarchie. Ta se projevuje již ve stádiu odhadu, například projekt se skládá z realizace více oddělených komponent. Při tvorbě WBS se zase komplexnější úkoly rozdělují na jednodušší podčásti. Některé takovéto hierarchické vztahy je vhodné zachovat i mezi úkoly v issue-tracking systému.

Některé metodiky vývoje tuto hierarchii využívají k organizaci práce. Například agilní metodika Scrum ([6]) používá terminologii stories a epics. Pojmem story se rozumí ucelená jednotka práce určená k implementaci. Epic je pak pojem nadřazený, většinou se jedná o rozsahem větší část realizace, která v sobě sdružuje více stories. Z hlediska funkcionality pak dodání epicu dává smysl pouze po dokončení všech jeho odpovídajících stories, kdežto story samo o sobě může být vhodná funkční jednotka určená k dodání.

Hierarchie však má smysl i v neagilních metodikách, právě třeba pro rozdělení jednotlivých funkčních celků, které lze dodat samostatně. Daný přístup má využití i pro měření a sledování projektu, kdy lze jednoduše bez tvorby složitých reportů sledovat, které části přesahují očekávanou pracnost, a u kterých se naopak zdá, že budou implementovány v odhadovaném čase (nebo rychleji).

Alternativou k běžnému hierarchickému uspořádání může být uspořádání tzv. ploché, kde místo vazeb nadřazenosti a podřazenosti se udržují vazby pouze na horizontální úrovni mezi úkoly stejné důležitosti. Zde je pak nutné informaci o zařazení úkolu evidovat jinak, například ukládáním položky s určitou hodnotou, která pak sdružuje veškeré souvislé prvky. Ploché uspořádání je jednodušší na údržbu, není třeba vytvářet složité struktury při tvorbě a přidávání nových úkolů. Na druhou stranu pro účely měření a sledování může být nezbytné z dat vytvářet složitější struktury a selekce nutné k tvorbě odpovídajících reportů.

Každý z těchto dvou přístupů má své pro i proti, pro účely „ideálního“ stavu v evidenci úkolů v issue-tracking systému se autor rozhodl zvolit spíše klasický hierarchický přístup.

2.4 Trasovatelnost požadavku napříč životním cyklem

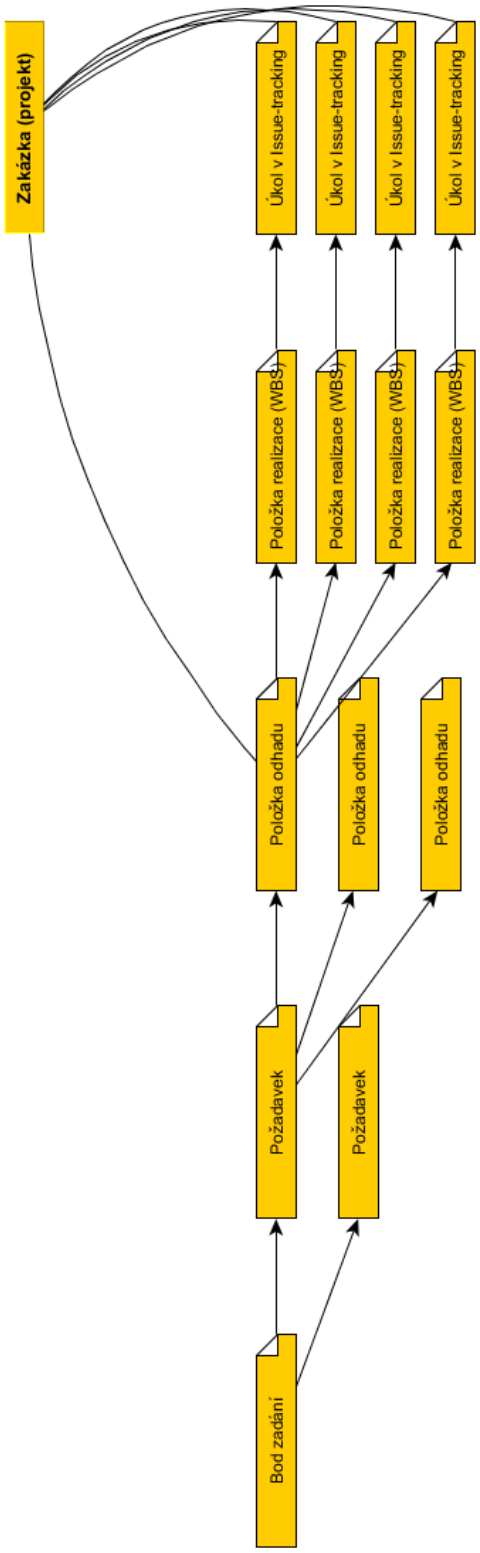
Jedním z důležitých požadavků na tento proces je sledování požadavku celým jeho životním cyklem a jeho zpětná trasovatelnost. Umožňuje nebo usnadňuje to určitou část výše zadefinovaných požadavků, jako je například sledování v rámci realizace (zde je možné se odkazovat na původní odhad) nebo zpětné vyhodnocení projektu.

Pro ilustraci životního cyklu požadavku slouží obrázek 2.3.

Každý požadavek začíná jako bod zadání či jeho část (záleží na konkrétním případu). Po specifikaci se požadavek může rozpadnout na jeden či více položek v odhadu. Dané se může v průběhu zpřesňování měnit, stále však platí, že každá položka odhadu musí odpovídat některému z požadavků. Je dobrým zvykem jednotlivé funkční i nefunkční požadavky číslovat (nebo jinak jednoznačně identifikovat), není tedy problém se v odhadu odkazovat, ke kterému požadavku se položka váže.

Při tvorbě WBS se položka odhadu může ještě více fragmentovat, položky WBS již však nestačí vázat pouze zpětně k požadavku, ale je nutné sledovat vazbu k položce odhadu. Po importu WBS do issue-tracking systému vzniknou jednotlivé úkoly pro implementaci, které by v ideálním případě měly vždy odpovídat některé položce WBS, a to jedna ku jedné. Úkol, který nevychází z WBS vznikne většinou tehdy, když se objeví nový požadavek v průběhu realizace, se kterým nebylo dříve počítáno. Příliš mnoho takovýchto úkolů může značit nedostatky v rozsahu nebo okrajových podmínkách odhadu (resp. WBS), či může naznačit identifikovaný bug (ty by však odhadem měly být pokryté).

Ve všech fázích životního cyklu je tak zajištěna existence trasovatelnosti jak zpětné, kdy je možné zjistit, odkud daná položka vychází, tak dopředné, kdy lze vidět, jak došlo k její realizaci.



Obrázek 2.3: Vazba položek napříč životním cyklem

2.5 Vyhodnocení a historie projektu

Jednou z nezbytných částí životního cyklu každého projektu je také finální zhodnocení celého projektu napříč jeho fázemi. Po realizaci projektu, ať už úspěšné či neúspěšné, by vždy mělo dojít k sepsání takového zhodnocení a jeho následné evidenci.

Zhodnocení může být pojato dvojí formou. Jednak se může jednat o textový soupis průběhu projektu, tedy vypsání toho, co autor považuje za důležité a podepsalo se to na průběhu projektu. Toto může zahrnovat různé zajímavé problémy, kvalitu součinnosti zákazníka, vliv velikosti a zkušeností vývojářského týmu nebo cokoliv dalšího. Takovýto soupis může v budoucnu sloužit jako návod či varování pro jiné nově vznikající projekty. Pro umožnění takovéhoto využití je nutné zajistit snadnou dohledatelnost takových historických shrnutí, které jsou v danou chvíli pro uživatele zajímavá. Toho lze dosáhnout například vhodnou kategorizací podle použitých technologií, zákazníka, podmínek pro tvorbu projektu apod. Protože se jedná o velice plochou kategorizaci, hodí se k tomu například použití tagů.

Druhou formou pojetí shrnutí projektu je pak pohled pomocí metrik. Na projektu lze měřit spoustu věcí, mezi nejdůležitější patří jednoznačně rozdíl odhadovaného času a finální časový náklad vynaložen na realizaci. Pokud dojde k překročení odhadované pracnosti, bývá pak z historického hlediska nejzajímavější identifikace původců těchto problémů, aby bylo možné se jim v budoucnu vyvarovat.

Ideální stav by byl takový, aby zhodnocení kombinovalo obě formy v jediném komplexním souhrnu, ze kterého pak jednak lze získávat relevantní data a použít je jako podklad při tvorbě nových projektů. Zároveň je však také možné dozvědět se o dalších vlivech, které by z pouhých naměřených hodnot nebyly patrné, ale které by zároveň mohly ovlivnit projekt. Kombinací těchto dvou typů informací si pak lze utvořit poměrně dobrou představu o průběhu daného projektu.

2.5.1 Databáze odhadů

Při tvorbě odhadu se může jeho tvůrci hodit možnost čerpat inspiraci z předešlých projektů a jejich odhadů. U softwarových společností pracujících na velkém množství projektů se často může ukázat, že dochází k tvorbě podobných funkčních celků, stojících mnohdy na stejných či podobných technologiích. Dané implikuje určitou pravděpodobnost, že v minulosti někdo již odhadoval dosti podobný požadavek. V takovém případě může být přínosem mít přístup k takovému historickému odhadu a zároveň vědět, jakým způsobem projekt nakonec dopadl - zda byl odhad nadhodnocený, či naopak nestačil. Zároveň je dobré vědět i důvody, proč k překročení či přebytku pracnosti došlo a tyto informace využít jako další prostředek k tvorbě nového odhadu.

Samozřejmě není vhodné využívat přesná čísla z historických odhadů a na jejich základě tvořit odhady nové, ale minimálně lze informace využít jako určité vodítko, kterého se lze držet. Na druhou stranu okrajové podmínky podobných projektů mohou být velice užitečné, protože často dochází k vymezení vůči podobným situacím. V kombinaci s historickým souhrnem projektu lze navíc identifikovat, kde nedošlo k dostatečnému vymezení a to se projevilo problémy při realizaci.

Kapitola 3

Shrnutí současného stavu

V této kapitole bude shrnut současný proces sběru požadavků, tvorby odhadů, měření pracnosti v rámci realizace a následná evidence historie projektů v rámci společnosti Profinit. Hlavním cílem je identifikovat rozdíly oproti požadovanému cílovému stavu nastíněnému v předešlé kapitole.

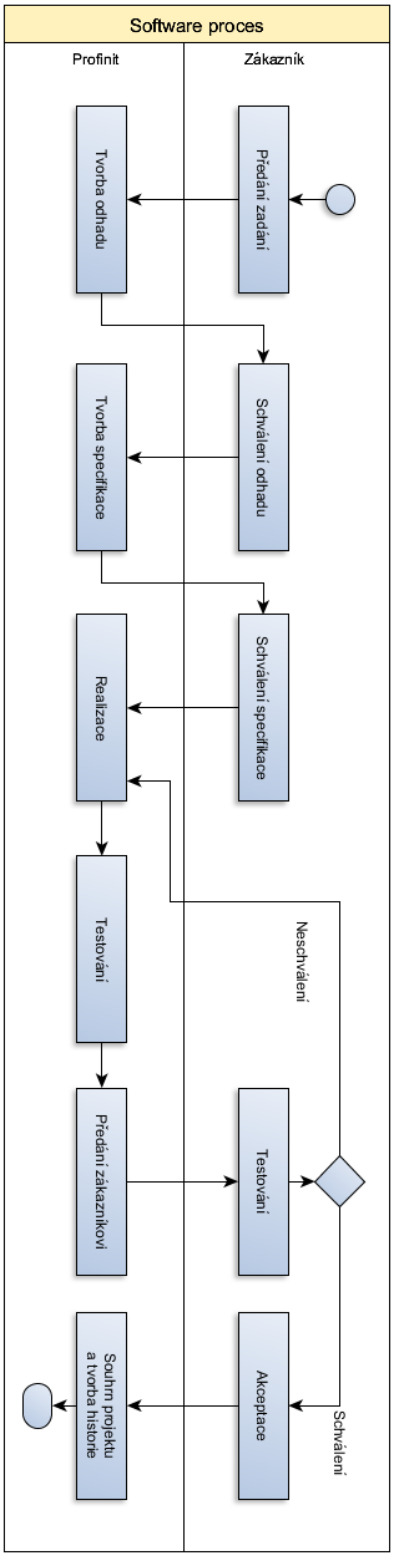
Díky určité decentralizaci a rozmístění týmů napříč různými projekty nemusí popisovaný stav nutně reflektovat reálný stav ve všech týmech společnosti, ale jedná se o jakýsi společný kvalitativní základ, který by měl být dodržován. Ne vždy to ale okolnosti působení týmu dovolují (může se jednat o součást zákazníkova interního projektu, kde mohou být využívány jiné postupy, apod.).

Pro zběžnou ilustraci tohoto procesu slouží diagram 3.1. Opět se jedná o zjednodušenou ilustraci, podrobnější diagram aktivit je k vidění na obrázku 3.2.

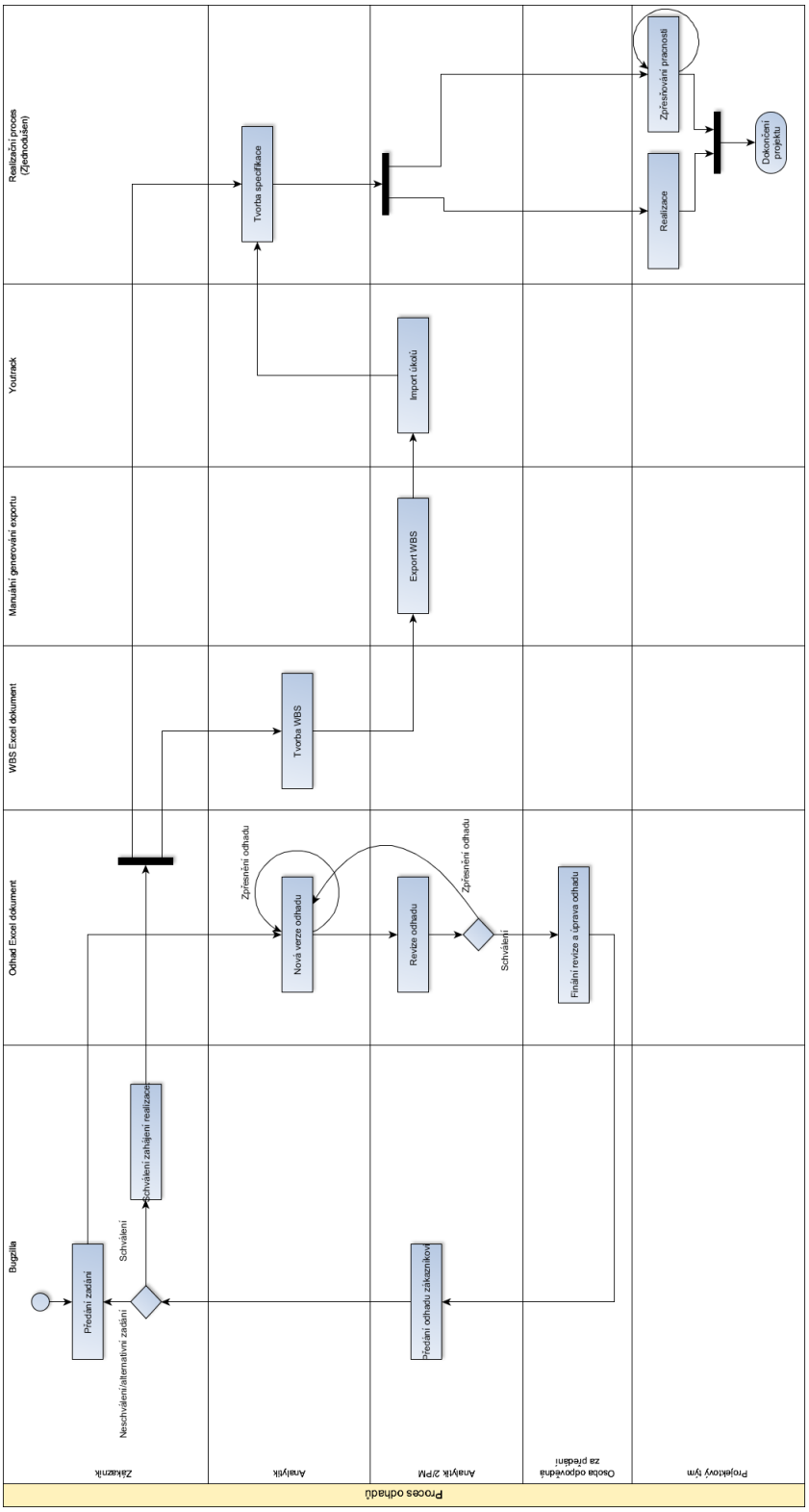
3.1 Iniciální sběr požadavků

Pro sběr požadavků je ve společnosti Profinit nejvíce využíván systém Bugzilla. Bugzilla je webová aplikace primárně určena ke sledování chyb, jedná se tedy o issue-tracking systém. Uživateli této aplikace je umožněno vytvoření tzv. bugů neboli chyb. Bug nicméně není primárně určen pouze k evidenci chyb, ale i k více obecnějšímu použití, například záznam nových funkcionalit, vylepšení, apod. U každého bugu je evidováno, kdo bug vytvořil, kdo je řešitel, lze jej kategorizovat, přikládat soubory a v neposlední řadě je možné vkládat komentáře, což umožňuje vést u každého bugu diskusi. Díky tomu, že systém Bugzilla umožňuje privátní komentáře, které může vidět pouze vymezená skupina uživatelů, je možné rozlišit mezi komentáři, které jsou viditelné pro zákazníka a komentáři určenými interně pro tým.

Ve společnosti Profinit je pro každé nové zadání vytvořen v první řadě úkol v systému Bugzilla obsahující základní informace o tomto úkolu. Je přiložen dokument se zadáním od zákazníka, pokud takové zadání již existuje.



Obrázek 3.1: Softwarový proces



Obrázek 3.2: Současný proces - diagram aktivit

Pokud ne, dojde k iniciální definici tohoto zadání. V obou případech může být zadání dále zpřesňováno jak ze strany zákazníka, tak pomocí dotazů ze strany analytiků a vývojářů.

V případech, kdy se jedná o celek s vyšší předpokládanou pracností, tzn., nejedná se pouze o drobnou úpravu stávajícího produktu, dojde většinou k rozdělení na samostatné logické celky. Pro každý takto vzniklý celek vznikne nový bug. Z původního bugu se stává hlavní projektový bug, který zastřešuje celý komplet. Účelem tohoto zastřešujícího bugu je evidence všech významných událostí týkajících se daného projektu, ať už technického, organizačního nebo obchodního charakteru. Dále pak většinou tento bug slouží jako hlavní způsob vedení komunikace o daném požadavku se zákazníkem.

3.2 Tvorba odhadu

3.2.1 Metodika tvorby odhadu

Metodika odhadu používaná společností Profinit čerpá především z [5] a [7].

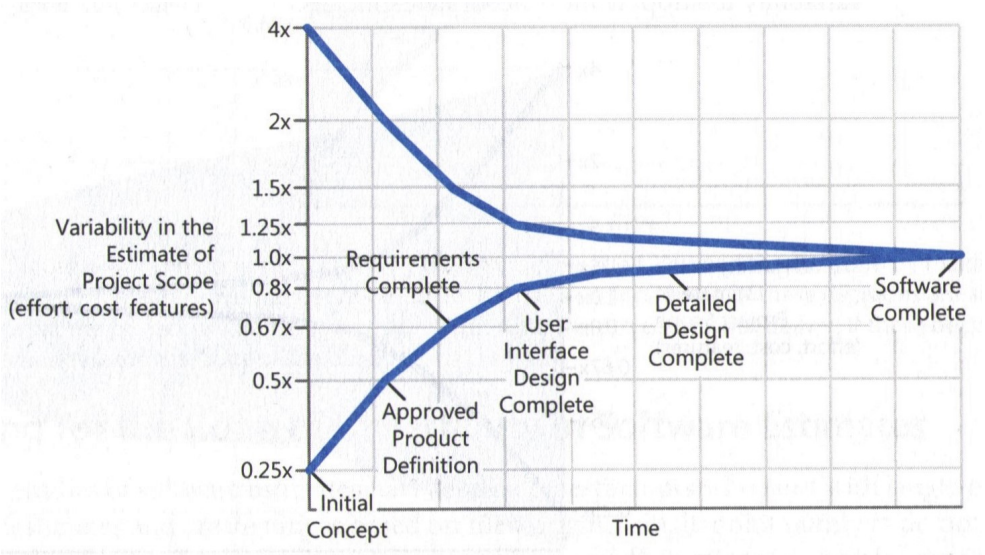
Tato metodika stojí na skutečnosti, že při odhadování projektů lze pozorovat tzv. kužel nejistoty (3.3). Během celého životního cyklu projektu existuje spoustu nejistot a proměnných, které se postupným zpřesňováním požadavků vyjasňují a ustalují. Zároveň se díky tomu zvyšuje přesnost odhadu. Kužel nejistoty ilustruje právě tento vývoj a ukazuje, v jakých fázích realizace projektu lze očekávat jakou chybu. V úvodní analýze se může odhad lišit až čtyřnásobně oproti skutečnosti (v obou směrech), přičemž tato čísla vycházejí ze stovek statistik již realizovaných projektů.

Cílem projektového týmu by mělo být odstraňování jednotlivých proměnných projektu, tak aby docházelo k co nejrychlejší konvergenci kuželu nejistoty. Pokud nebude docházet k dostatečnému konkretizování požadavků projektu včas, budou odhady v jednotlivých fázích obsahovat daleko větší rozptyl chybovosti, než udává kužel a než by bylo pro projekt příhodné.

3.2.1.1 Odhad

Odstraňování nejistot, a v důsledku zpřesňování projektu, je dosahováno zejména sběrem požadavků od zákazníka. Vytvářený odhad pak stojí na základu těchto znalostí. Proti nežádoucím vlivům neznámých skutečností a nejistot je třeba se vymezit pomocí omezujících podmínek. Ty jsou nedílnou součástí odhadu. Udávají, jaké jsou uvažovány předpoklady pro platnost daného odhadu. Například může být uvažován konkrétní design řešení nebo technologie, aplikace jiného návrhu pak může mít vysoce odlišnou pracnost.

Kromě omezujících podmínek obsahují odhady také rozpad jednotlivých požadavků na menší, ideálně nedělitelné položky. Každá položka se skládá z popisu dané činnosti, minimální a maximální očekávané pracnosti a tzv. expertního odhadu. Jedná se o odhad pracnosti očekávaný zkušeným odhadu-



Obrázek 3.3: Kužel nejistoty

jícím, a měl by se tedy logicky pohybovat v intervalu minimálního a maximálního odhadu. Z těchto tří hodnot se pak dopočítávají další důležité hodnoty. První z nich je průměrný odhad, který je pouze průměrem minimální a maximální hodnoty. Druhou hodnotou je pak očekávaná pracnost, která opět vychází z McConnelovy metodiky [5]. Jedná se o výpočet váženého průměru z minimálního, maximálního a expertního odhadu. Expertnímu odhadu je pak přikládána váha na základě zkušenosti odhadujících. Ve společnosti Profinit se expertnímu odhadu nejčastěji přikládá váha 4, vzorec pro výpočet je pak tedy následující rovnice 3.1:

$$\frac{Min + 4 * Exp + Max}{6} \tag{3.1}$$

kde *Min* a *Max* je minimální resp. maximální odhadovaná pracnost a *Exp* je expertní odhad. U zkušených vývojářů je možné přidělit expertnímu odhadu vyšší váhu, naopak u málo zkušeného týmu lze váhu snížit. Konkrétní koeficient však není jednoduše přenositelný a musí být po několika iteracích rekalibrován.

Jednotlivé položky jsou členěny do kategorií, kdy pro tradiční softwarové projekty je využito rozdělení na kategorie softwarového inženýrství:

- Analýza
- Design
- Implementace
- Testování

- PM
- Dodávka
- Ostatní
- Záruka

Používaná metodika společně s firemním know-how poskytuje některá doporučení, v jakém podílu by měli jednotlivé kategorie být v odhadu celkové pracnosti zastoupeny. Žádná z kategorií by neměla pracností výrazněji přesahovat ostatní. Některé často podceňované kategorie pak mají konkrétní minimální hodnoty, pod které by se neměla daná kategorie dostat. U testování se udává hodnota 20 %, lze se však také řídit poměrově k implementaci, kde by obě kategorie měli v nejlepším případě mít přibližně stejný podíl z celku. Doporučovaný podíl projektového managementu je pak 15 % u nových zákazníků. U zákazníka, se kterým již existuje předchozí zkušenost, lze toto číslo snížit.

Speciální kategorií je pak Záruka. Ta nespadá mezi ostatní kategorie a je brána odděleně. Slouží k pokrytí potenciálního času stráveného na opravách chyb v záruční lhůtě. V závislosti na parametrech poskytované záruky (délka, rozsah) a dalších okolnostech projektu se výše záruky pohybuje v jednotkách procent.

3.2.2 Excel dokument

V současné době je většina odhadů ve společnosti Profinit vytvářena pomocí dokumentu v nástroji Microsoft Excel. Tento dokument je svým způsobem šablona pro tvorbu odhadu podle uváděné metodiky. Ve velké většině projektů je používáno standardní kategorizace softwarového inženýrství, o kterém již bylo pojednáno v předchozí části 3.2.1.

Jednotlivé listy (3.4) dokumentu se pak vztahují k těmto kategoriím. Každý list obsahuje tzv. checklist, což je kontrolní seznam s body, které by měl odhadující při provádění odhadu dané kategorie projít. Hlavní částí listu je pak seznam položek odhadu vázajících se k dané kategorii. Tyto položky reprezentují jednotlivé části projektu. Každá položka obsahuje popis a odhadovanou pracnost určenou k realizaci dané části. Pracnost je v souladu s používanou metodikou vyjádřena více časovými údaji reprezentující minimum, maximum, expertní odhad. Z nich je dopočítaná průměrná a očekávaná pracnost. Dále je možné u každé položky využít příznak varianty, která umožňuje v jednom odhadu připravit více možných variant projektu. To přináší zvýšenou flexibilitu při předávání naceněného odhadu zákazníkovi.

Kromě jednotlivých kategorií obsahuje dokument také seznam omezujících podmínek a předpokladů. V každém odhadu je nutné mít pečlivě vymezený rozsah vytvářeného projektu. Nemůže se pak stát, že zákazník požaduje dodání

Implementace

Varianta	Popis činnosti	Min (MH)	Max (MH)	Nej. prav. (MH)	Prům. (MH)	Oček. (MH)
	[redacted]	3,00	5,00	4,00	**4,00**	**4,00**
		2,00	3,00	2,50	**2,50**	**2,50**
		1,00	2,00	1,50	**1,50**	**1,50**
		3,00	4,00	3,00	**3,50**	**3,17**
		1,80	2,80	2,20	**2,30**	**2,23**
					0,00	**0,00**
					0,00	0,00
					0,00	0,00
					0,00	**0,00**
					0,00	0,00
					0,00	0,00
					0,00	0,00

něčeho, s čím nebylo v odhadu počítáno a tedy ani není zohledněna pracnost této činnosti v ceně projektu. Opět je možné navázat jednotlivé předpoklady a omezující podmínky k různým variantám odhadu.

Největší silou tohoto odhadovacího nástroje ja pak stránka souhrnu (k vidění na obrázku 3.5), která nabízí celkový přehled vytvářeného odhadu. Jednak jsou poskytnuty souhrny pracností jednotlivých kategorií i celku, procentuální zastoupení jednotlivých kategorií v celkové pracnosti, poměrné riziko nejistoty kategorie vůči celku a další informace sloužící k tvorbě a revizím vznikajícího odhadu. Na tomto listu je také možné přepínat mezi jednotlivými variantami odhadu, měnit časovou jednotku zadávání odhadů z MD na MH (A) a nazpět. V neposlední řadě je zde tabulka s kalkulací ceny daného projektu určena k předání zákazníkovi. Ta je rozdělena na analýzu, realizaci a dodávku.

Dokument také podporuje export odhadu v textové podobě určené k předání zákazníkovi. Výstupem je souhrn omezujících podmínek a pracnosti společně s cenou.

3.2.3 Estimate

Jako alternativa k vytváření odhadů za pomocí Excel šablony vznikl v rámci bakalářských prací Bc. Milana Vancla a Bc. Juraje Polačoka nový nástroj Estimate. Cílem této aplikace bylo a je nahrazení stávajících Excelových šablon.

Jedná se o webovou aplikaci vytvářenou v jazyce Java za použití frameworku Spring. GUI aplikace je generováno frameworkem Vaadin. Data odhadů jsou ukládána do noSQL databáze Neo4j.

K hlavním motivacím tvorby dedikované aplikace pro vytváření odhadů patří unifikace tvorby odhadů napříč společností, sběr a lepší použití okrajových podmínek odhadů, evidence historie metrik k usnadnění budoucích odhadů a integrace procesu tvorby odhadů do stávajícího ekosystému společnosti.

Při přihlášení do systému Estimate je uživatel zaveden na přehled jednotlivých odhadů, ke kterým má právo na čtení. Tvorba nového odhadu pak, po samotném založení, probíhá na obrazovce (3.6), ze které je patrná inspirace původním odhadovacím Excel dokumentem. Je tedy možné vytvářet nové úkoly, doplňovat jejich pracnost, psát okrajové podmínky podobně jako dříve. Lze používat jednoduché rovnice, podobně jako funkce v MS Excel.

Oproti původnímu dokumentu je souhrn veden v samostatném okně aplikace. Místo listů obsahujících kategorie je zde uživatelsky přívětivější postranní komponenta, ve které je možné přepínat mezi jednotlivými kategoriemi.

V současné době zatím tato aplikace není standardně využívána v žádném z týmů společnosti Profinit, je provozována pouze v pilotním provozu. Jedním z cílů této práce je také příprava tohoto nástroje k jeho využitelnosti ve stávajícím vývojovém procesu a usnadnění jeho použití. Po vytvoření aplikace se

Celkový přehled

	Min (MD)	Max (MD)	Nej. prav. (MD)	Prům. (MD)	Oček. (MD)	Riziko (%)
Analýza	1,1	1,8	1,4	1,4	1,4	31,2%
Design	0,0	0,0	0,0	0,0	0,0	0,0%
Implementace	1,4	2,1	1,7	1,7	1,7	37,4%
Testování	1,1	1,4	1,2	1,2	1,2	15,6%
PM	0,5	0,8	0,6	0,6	0,6	11,0%
Dodávka	1,0	1,0	1,0	1,0	1,0	0,0%
Ostatní	0,0	0,0	0,0	0,0	0,0	0,0%
Záruka	*0,3*	*0,3*	*0,3*	*0,3*	*0,3*	4,8%
Realizace (MD)	**2,9**	**4,2**	**3,5**	**3,6**	**3,5**	
Realizace včetně záruky (MD)	**3,2**	**4,6**	**3,8**	**3,9**	**3,8**	
Celkem bez záruky (MD)	**5,1**	**7,0**	**5,8**	**6,0**	**5,9**	
Celkem včetně záruky (MD)	**5,3**	**7,3**	**6,1**	**6,3**	**6,2**	

Hodnoty zadávám v	MH

Již zadané hodnoty se při změně nepřepočítají!

Odhad pracnosti - var. 1

Činnost	Analýza	Realizace (předběžná)		Dodávka
		Minimum	Maximum	
Pracnost v čd	1,5	3,5	5,0	1,0
Cena bez DPH				

Záruka	5%
Cena za MD	

Dostupné varianty	1	2	3	4	5	6
Uvažované varianty	1					

Poznámka: varianty, které nebudou realizovány, vymažte z řádku **"Uvažované varianty"**

Podíl kategorií v celku

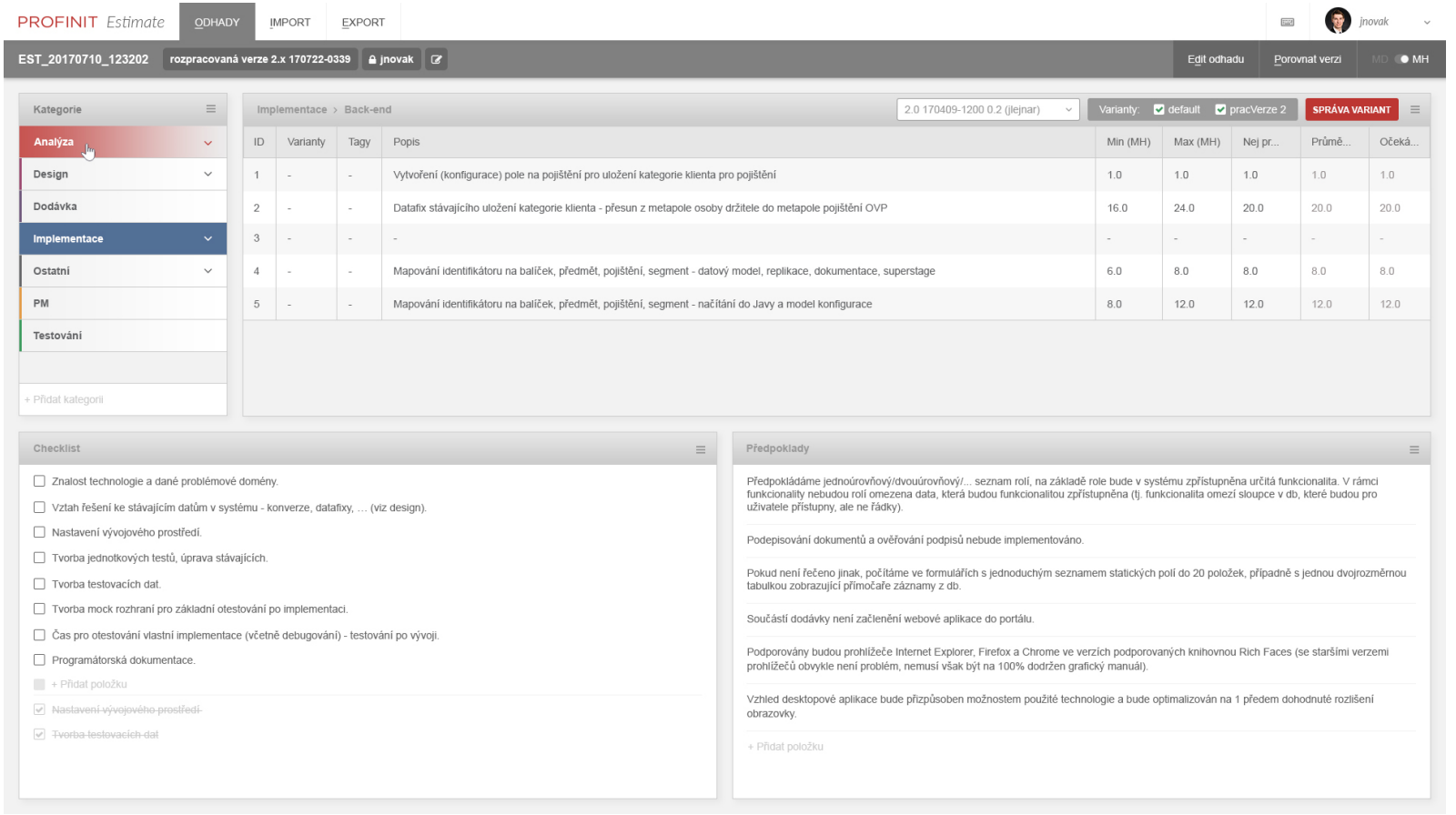
	Min (MD)	Max (MD)	Nej. prav. (MD)	Prům. (MD)	Oček. (MD)
Analýza	21,14%	23,89%	22,43%	22,74%	22,54%
Design	0,00%	0,00%	0,00%	0,00%	0,00%
Implementace	25,37%	28,67%	26,92%	27,28%	27,04%
Testování	19,97%	18,77%	19,37%	19,28%	19,34%
PM	9,96%	10,24%	10,20%	10,12%	10,17%
Dodávka	18,79%	13,65%	16,31%	15,82%	16,15%
Ostatní	0,00%	0,00%	0,00%	0,00%	0,00%
Podíl realizace	55,30%	57,69%	56,49%	56,68%	56,56%

Podíl kategorií v celku se bere ze Souhrnu včetně záruky

Pomocí klávesové zkratky CTRL + SHIFT + B spustíte export do Bugzilly

Uvažované varianty	1					

Obrázek 3.5: Excel odhad - souhrn



Obrázek 3.6: Estimate - přehled odhadů (vizuální návrh)

bohužel ukázalo, že některá návrhová rozhodnutí nejsou slučitelná s provozem aplikace v ostrém prostředí, alespoň tedy ne bez výraznějších úprav.

Jedním z problematických částí aplikace se ukázalo být použití grafové databáze Neo4j. Neo4j byla v prvotní realizaci zvolená proto, že použití grafové databáze odpovídalo modelované doméně více, než použití databáze relační. Ukázalo se, že lze bez problému danou doménu modelovat i v relační databázi, jejíž použití, je z hlediska dalšího rozvoje projektu a údržby daleko výhodnější než použití databáze grafové. Je tomu tak hlavně díky daleko větší komunitě, starající se o vývoj a podporu nástrojů určených pro relační databáze, dále také mnohem větší dostupností specialistů, kterých je u velký grafových databází nedostatek. To není ideální stav pro budoucí rozvoj projektu, je zapotřebí volit takové technologie, které je možné jednoduše udržovat.

Další překážkou pro rozšíření používání napříč společností jsou mírné nedostatky v GUI, které se však podepisují na použitelnosti. Ukazuje se, že uživatel efektivní ve vytváření odhadů ve starém dokumentu není plně spokojen při používání nového GUI. Možnost využití mnoha klávesových zkratek na této situaci příliš nezměnila. Pro zajištění větší podpory od uživatelů je potřeba GUI doladit, nejlépe za zpětné vazby od uživatelů.

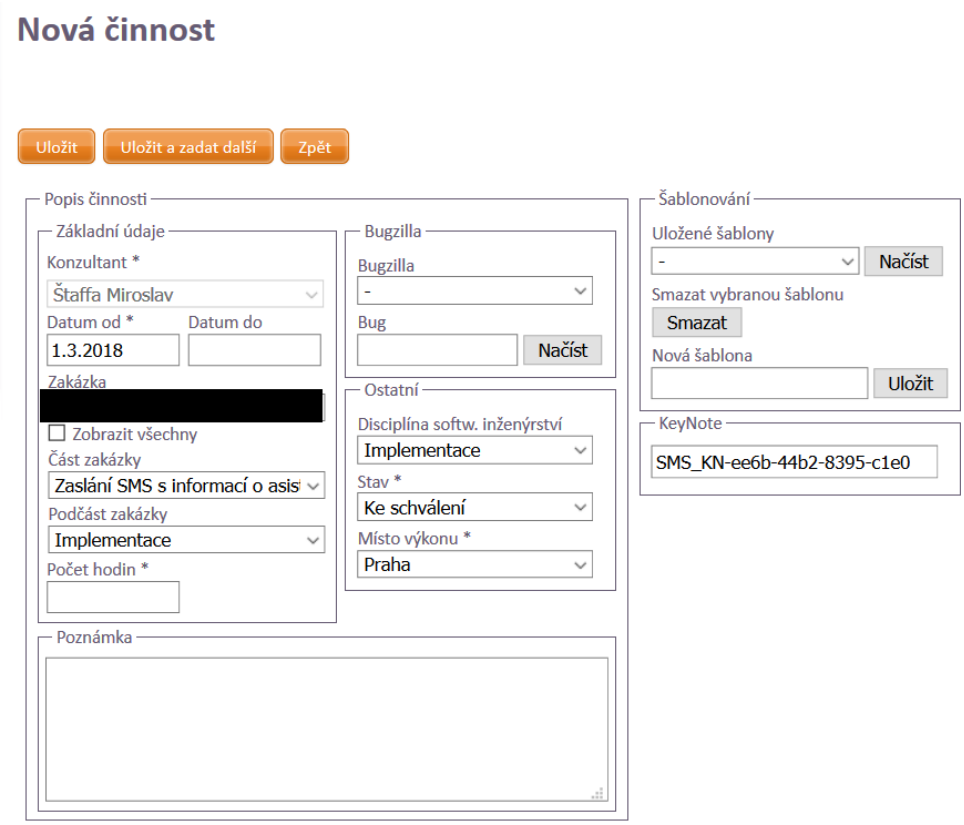
3.3 Realizace projektu

Předtím než bude rozebrán samotný proces realizace, je nutné představit informační systém Profis, který zasahuje do většiny částí zbývajícího životního cyklu projektu.

3.3.1 Profis

Profis je Profinitem vyvíjený informační systém určený pro interní potřeby. Sdružuje v sobě systém na vykazování času, plánování podpor, CRM a mnoho dalšího. Z pohledu procesu vývoje, a tím pádem i této práce, je však nejdůležitější právě část starající se o udržování informace o odpracovaném čase na dílčích projektech, resp. úlohách.

Profis v sobě obsahuje jednotky korespondující s jednotlivými projekty, pro které je použita terminologie zakázka. Zakázky lze dále členit na části a podčásti, které odpovídají dalšímu členění projektů. Odpracovaný čas na jednotlivých projektech je pak ve formě tzv. výkazů uložen do systému. Každý výkaz (3.7) obsahuje informace o tom, k jakému datu se vztahuje, jaké zakázky se týká, o jakou se jednalo činnost a mimo dalších rozšiřujících údajů také keynote, pro umožnění navázání na konkrétní úkol. V ideálním světě by bylo možné využívat pouze jednotlivé zakázky a jejich části a podčásti, nicméně tyto tři úrovně ne vždy stačí. Keynote, díky zploštění této hierarchie, umožňuje držet informaci o odpracovaném čase při libovolné hloubce a struktuře daného projektu.



Obrázek 3.7: Profis výkaz

3.3.2 Zahájení realizace

Po schválení odhadu ze strany zákazníka je možné zahájit práce na realizaci projektu. Před počátkem realizace dojde k tvorbě WBS (viz 2.2.1). V kontextu Profinitu se jedná o podrobnější rozpad jednotlivých položek odhadu na drobnější celky, tak aby byly pokud možno řešitelné jedním členem týmu, a to v maximálním časovém rozsahu do dvou dní. Rozpad vychází z finální, schválené verze odhadu, případně specifikace, a musí pokrývat celkový rozsah projektu.

3.3.3 WBS

Ve společnosti Profinit není pro tvorbu a využití WBS v současnosti standardizovaný postup. Každý projekt si tak tuto část vývojového procesu řeší svým vlastním způsobem. Autor této práce se měl možnost setkat se dvěma přístupy. Prvním z nich, který je používán častěji, je tvorba WBS do šablony k tomu určené, v dalším Excelovém dokumentu, tzv. plánu projektu.

Projektový plán je opět šablona určená pro nástroj MS Excel. Dokument není určen výhradně k tvorbě WBS, ale před uvedením systému Youtrack sloužil zároveň jako sledovací nástroj pro měření vynaloženého úsilí v exekuční fázi realizace. Dále obsahuje spoustu dalších rozšíření určených zejména pro usnadnění projektového řízení, jako jsou evidence zakázek souvisejících s projektem, časové harmonogramy členů týmu usnadňující plánování, a spoustu dalších funkcionalit. Funkce náhrady měřícího nástroje v exekuční fázi byla umožněna propojením s informačním systémem Profis, který obsahuje informace o vykázaném čase (další informace v kapitole 3.3.1).

Hlavní částí plánu projektu obsahující i WBS je pak list úloh (3.8). Ten obsahuje již na nejmenší podúlohy rozpadlý projekt, stejně jako v tradiční WBS. Každý záznam pak kromě jiného obsahuje původní a aktuálně odhadovaný čas, odpracovaný čas, výsledný rozdíl odhadované a využité pracnosti, procentuální realizaci daného úkolu, odpovědnou osobu a mnohé další informace.

V poslední době však místo projektových plánů převládá odlišný proces této fáze vývoje. WBS je vytvářená do, oproti projektovým plánům, velice zjednodušené Excelové šablony. Ta u každé jednotlivé položky obsahuje název a popis činnosti, její pracnost, příslušnost ke kategorii softwarového procesu a, podobně jako v případě projektových plánů, keynote zajišťující mapování na Profis. Hlavním rozdílem oproti staršímu přístupu je ten, že tento dokument slouží pouze jako zdroj dat do tradičního issue-tracking systému Youtrack. Výhody tohoto přístupu se ukáží až v následující kapitole 3.3.4.

Import WBS do Youtrack je v současnosti prováděn z části manuálním způsobem. WBS je po schválení transformována z CSV do XML souboru. Tento soubor je následně pomocí nástroje SoapUI nahrán RESTovým voláním do systému Youtrack. Z důvodu použití volání přes SoapUI je nutné řešit i autentizaci uživatele do REST API systému Youtrack, kterou je nezbytné provádět separátním voláním a následným kopírováním zaslaných cookies. Tento postup tedy není příliš efektivní.

3.3.3.1 Zohlednění metodiky odhadů ve WBS

Běžnou součástí tvorby WBS může být využití určitého pravidelného postupu či zohlednění metodiky před finalizací WBS. Například autor práce se měl možnost na projektu setkat s oddělením části pracnosti určené pro implementaci a testování na opravy chyb zjištěných během akceptačního testování zákazníkem. Je tak činěno manuálně autorem WBS, což s sebou přináší mnoho problémů. Jednak přesnost jednotlivých výpočtů stojí pouze na pečlivosti autora, může tedy snadno dojít k pochybení, dalším problémem je potřeba tyto výpočty přepočítávat vždy při každé úpravě pracnosti ve WBS. Taková situace může nastat snadno a často, například pokud nedojde ke shodě mezi tvůrcem a revidujícím. Tato část procesu tvorby WBS tak může někdy být poněkud zdlouhavá.

orita	Kategorie	Stav	Název činnosti (popis je ve sloupci F)	Osob	TM	Hotovo [%]	Celkem	Odpracov áno [č	Zbývající pracnost [čh]	Zbývající pracnost [čd]	Původní odhad [čh]	Původní odhad [čd]	Rozdíl [čd]
1	Implementace	Hotovo	S-Cube - Úprava tisků pro změny	TS	TEST								
			Implementace PZO										
1	Implementace	Hotovo	PZO - Karta PZ - Úprava obrazovky Karta PZ a d_uziv_zprostredkovatel_bal (JSP,	LB	TEST	100%							
1	Implementace	Hotovo	PZO - Admin - Úprava obrazovky konfigurace PZ a c_bal_zprostredkovatel_bal (	MC	TEST	100%							
1	Implementace	Hotovo	PZO - 310 - Úprava formuláře Předsmluvní dokumentace (JSP, Java, DB)	MC	TEST	100%							
1	Implementace	Hotovo	PZO - 004 - Přidání nových polí PZ1 (JSP, Java, DB)	LB	TEST	100%							
1	Implementace	Hotovo	PZO - 004 - Ukládání lokálního PZO (Java, DB, bez DM - viz task níže)	LB	TEST	100%							
1	Implementace	Hotovo	PZO - 004 - Vyhodnocení aktivnosti a povinností polí (Java) po změně polí (volě	LB	TEST	100%							
1	Implementace	Hotovo	PZO - 004 - Přednastavení polí ze SEZ / Karty PZ (Java) po stisku tlačítka (volání	LB	TEST	100%							
1	Implementace	Hotovo	PZO - 004 -Zpracování PZO v IK při přechodu na 310 (Java) - volání P_ZPRACU, op	SP	TEST	100%							
1	Implementace	Hotovo	PZO - 004 - Zpracování PZO v EK po uložení PS (Java) - vytvoření a odeslání požac	SP	TEST	100%							
1	Implementace	Hotovo	PZO - Komunikace SEZ - wrapper pro dotaz do view V_VYJIMKA	LB	TEST	100%							
1	Implementace	Hotovo	PZO - Komunikace SEZ - wrapper pro volání procedury P_STAV	SP	TEST	100%							
1	Implementace	Hotovo	PZO - Komunikace SEZ - wrapper pro dotaz do view V_POVINNOST	LB	TEST	100%							
1	Implementace	Hotovo	PZO - Komunikace SEZ - wrapper pro volání procedury P_ZPRACUJ	SP	TEST	100%							
1	Implementace	Hotovo	PZO - 004 - Tlačítko pro načtení polí ze SEZ / Karty PZ - nová action, struts config	LB	TEST	100%							
1	Implementace	Hotovo	PZO - 004 - Kontrola zprostředkovatele včetně konfigurace	SP	TEST	100%							
1	Implementace	Hotovo	PZO - DM - Tabulka D_DATA_SPOLUPRAC_BAL - inter, exter, replikace	LB	TEST	100%							
1	Implementace	Hotovo	PZO - DM - Tabulka D_ZISKATEL_BAL - inter, exter, replikace	LB	TEST	100%							
1	Implementace	Hotovo	PZO - Konfigurace práva na editaci polí, metoda do Javy	LB	TEST	100%							
1	Implementace	Hotovo	PZO - Úprava kontroly provizí a práva na produkt (Java) - předání NADRAZEN_ID	SP	TEST	100%							

Obrázek 3.8: Plán projektu - WBS

Na jiných projektech se mohou používat, a často používají, podobné praktiky v závislosti na okolnostech a zavedených postupů. Pokud by došlo k automatizaci takovéto transformace, usnadnila by se tím spousta práce při odhadovacím procesu.

3.3.4 Kontrola průběhu realizace

Pro účely kontroly exekuční fáze projektu je ve společnosti Profinit používán issue-tracking systém Youtrack.

3.3.4.1 Youtrack

Youtrack je issue-tracking systém vyvíjený společností JetBrains. Z pohledu vývojáře nabízí poměrně standardní přístup k evidenci úkolů, jejich členění do projektů, odhadům pracností a měření vynaloženého úsilí. Kromě jiného podporuje agilní vývoj díky funkcím umožňujícím vizualizaci Scrum a Kanban tabulí a některých dalších jiných pohledů (mezi jeho silné stránky však patří hlavně tvorba reportů a poskytované REST rozhraní).

U jednotlivých úkolů se evidují různé údaje jako autor, řešitel, priorita úkolu, typ úkolu, odhadovaná pracnost, odpracovaný čas apod. Pokud má uživatel potřebu evidovat některé další údaje, nechybí možnost nakonfigurovat nová pole určená k evidenci. Lze u nich zvolit datový typ, připravit sadu povolených hodnot i třeba určit výchozí hodnotu. Pro účely odhadovacího procesu je zde využíváno vlastní pole původní pracnosti, které na rozdíl od běžné pracnosti zůstává konstantní a neměnné bez ohledu na vývoj daného úkolu. Standardní pole pracnosti pak lze upravovat s ohledem na nové okolnosti. Důvod i využitelnost udržování původní pracnosti vychází z 2.3.

Další dobrou vlastností aplikace Youtrack, která je poměrně unikátní, alespoň co se nástrojů tohoto typu týče, je možnost anotace položek pomocí tzv. tagů. Tagy jsou v Youtracku široce používané. V první řadě je jejich tvorba velice snadná, tag se vytvoří již pouze tím, že se jej k nějakému issue snaží uživatel přidat. Po jejich tvorbě je lze dále upravovat, například jejich viditelnost pro ostatní uživatele, v omezené míře lze přidávat i chování na základě určitých tagů (třeba provádění emailových notifikací).

Tagy jsou velice užitečné pro kategorizaci jednotlivých úkolů. Ideálním použitím tagů může být třeba evidence oblasti softwarového inženýrství, které se daný úkol týká (implementační úkol, úkol určený pro analýzu apod.). Pomocí tagů lze jednoduše seskupovat úkoly projektu do více plochého uspořádání, což může být alternativa k tradičním stromovým hierarchiím, které Youtrack také podporuje a které mohou být užitečné například při využívání některých agilních metodik, jakou je například Scrum.

Ve společnosti Profinit se dosud v kombinaci s Youtrackem využívalo spíše ploché uspořádání, kdy jednotlivé úkoly jsou sdružovány pomocí metadat, buďto anotací, nebo speciálním polem obsahující hodnotu. V průběhu tvorby

této práce dochází k experimentálnímu přechodu na strukturu hierarchickou. Je vytvořen hlavní úkol zahrnující jeden projekt či důležitý release. Ten v sobě poté sdružuje podúkoly zastřešující větší logicky ucelené komponenty a jednotlivé úkoly lze nalézt až v úrovni podřazené této. Automaticky se také sčítají jednotlivé odhadované a odpracované časy a propisují se na vyšší úroveň. Díky této vlastnosti lze snadno sledovat souhrnné agregace odhadovaných pracností i odpracovaného času na jednotlivých vytvářených komponentách i projektech, bez nutnosti tvořit složité reporty. Jelikož pole pro původní odhadovanou pracnost není standardní pole aplikace Youtrack, nedochází k jeho automatickému propisování na vyšší úrovně a je třeba toto řešit ručně, většinou zduplikováním hodnoty aktuálního odhadu. Toto představuje určité problémy, pokud v mezičase došlo k úpravě odhadované pracnosti. V takovém případě se nelze na vyšších úrovních na hodnotu původního odhadu spolehnout.

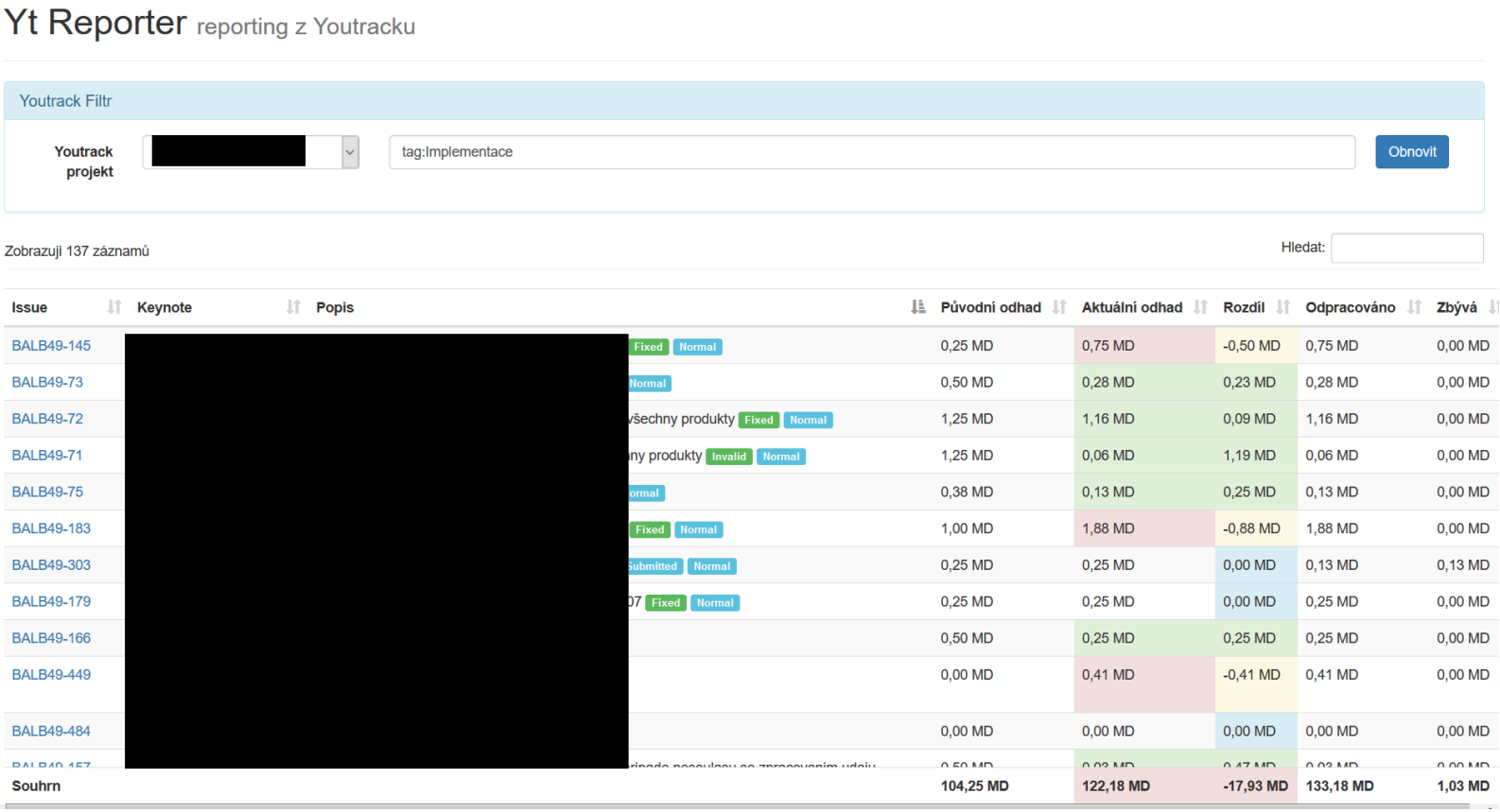
Youtrack umožňuje vyhledávání issues na základě připravených dotazů. Dotazy jsou zapisovány v pro Youtrack určeném dotazovacím jazyce, který je však velice podobný běžně používaným vyhledávacím jazykům v jiných issue-tracking systémech. Umožňuje vyhledávání na základě všemožných parametrů, nejčastěji využívané jsou pak filtry pro rozlišení projektu, řešitele úkolu, dodávané verze, apod.

3.3.4.2 YtReporter

Díky Youtrackem poskytovanému REST rozhraní je možné vytvářet různé nástroje, které rozšiřují možnosti jeho použití. Jedním z těchto již vytvořených nástrojů je aplikace YtReporter. Jedná se o webovou aplikaci vytvořenou za pomoci programovacího jazyka Groovy. I když nativní reportovací možnosti Youtracku patří mezi konkurenci issue-tracking systémů k těm lepším, z pohledu požadavků na proces vývoje ve společnosti Profinit nejsou plně dostačující. Hlavním problémem je absence jednoduchého tabulkového reportu pracnosti nad danými úkoly, navíc jsou všechny reporty poměrně statické, nelze jednoduše měnit vyhledavací query, ale je nutné pro každé zvlášť vytvářet nový report.

Nástroj YtReporter poskytuje užitečný náhled (3.9) na srovnání aktuálního odhadu s odhadem původním, aktuálně odpracovaným časem a časem zbývajícím. Dané je možné provádět jak na jednotlivých úkolech, tak i souhrnně. Lze tak snadno usoudit, zda projekt probíhá podle plánu, kde mohou vznikat potenciální problémy a kde se naopak čas ušetřil. Díky robustnímu REST rozhraní Youtracku je možné filtrovat úkoly stejně, jako to lze učinit přímo v nástroji Youtrack. Lze tak filtrovat například na základě tagů, kterými jsme schopni vyfiltrovat pouze části projektu, které nás aktuálně zajímají.

Nástroj YtReporter je již v současnosti reálně využíván pro kontrolu průběhu realizace projektů.



Yt Reporter reporting z Youtracku

Youtrack Filtr

Youtrack projekt [redacted] tag:Implementace Obnovit

Zobrazuji 137 záznamů Hledat:

Issue	Keynote	Popis	Původní odhad	Aktuální odhad	Rozdíl	Odpracováno	Zbývá
BALB49-145	[redacted]	Fixed Normal	0,25 MD	0,75 MD	-0,50 MD	0,75 MD	0,00 MD
BALB49-73		Normal	0,50 MD	0,28 MD	0,23 MD	0,28 MD	0,00 MD
BALB49-72		všechny produkty Fixed Normal	1,25 MD	1,16 MD	0,09 MD	1,16 MD	0,00 MD
BALB49-71		ny produkty Invalid Normal	1,25 MD	0,06 MD	1,19 MD	0,06 MD	0,00 MD
BALB49-75		ormal	0,38 MD	0,13 MD	0,25 MD	0,13 MD	0,00 MD
BALB49-183		Fixed Normal	1,00 MD	1,88 MD	-0,88 MD	1,88 MD	0,00 MD
BALB49-303		ubmitted Normal	0,25 MD	0,25 MD	0,00 MD	0,13 MD	0,13 MD
BALB49-179		07 Fixed Normal	0,25 MD	0,25 MD	0,00 MD	0,25 MD	0,00 MD
BALB49-166			0,50 MD	0,25 MD	0,25 MD	0,25 MD	0,00 MD
BALB49-449			0,00 MD	0,41 MD	-0,41 MD	0,41 MD	0,00 MD
BALB49-484			0,00 MD	0,00 MD	0,00 MD	0,00 MD	0,00 MD
Souhrn			104,25 MD	122,18 MD	-17,93 MD	133,18 MD	1,03 MD

Obrázek 3.9: YtReporter - Report

3.3.4.3 Profis - Youtrack integrace

V issue-tracking systémech je běžně udržována informace o odpracovaném čase, nicméně ve společnosti Profinit již je, jak je uváděno výše, k tomu určen informační systém Profis. Aby se předešlo duplicitnímu vykazování, bylo nutné nějakým způsobem zajistit synchronizaci těchto dvou systémů. Toho je dosaženo pomocí již dříve zmíněnými keynote, které jsou evidovány u každého Youtrack úkolu. Při vykazování času v systému Profis je pak mimo jiné zadán keynote. Pravidelně je pak spouštěno dávkové zpracování s pomocí systému průběžné integrace Jenkins. Tato dávka sesynchronizuje časy z Profis s odpovídajícími úkoly v systému Youtrack (provázání je zajištěno právě pomocí keynote). Informace o již vynaloženém úsilí je tak pravidelně aktualizována podobně, jako kdyby docházelo k vykazování času přímo v Youtrack.

Aby však nebylo nutné manuálně dohledávat data v systému Profis, existuje možnost použití Javascript rozšíření (3.10), využívající další metadata úkolu pro identifikaci odpovídající zakázky v systému Profis. Na jejich základě lze jedním kliknutím otevřít nové okno s již předvyplněními hodnotami pro výkaz v systému Profis.

Tímto způsobem je zajištěna obousměrná synchronizace Youtracku a Profisu z pohledu udržování informací o pracnosti.

3.4 Vyhodnocení a historie projektů

Vyhodnocování projektů společnosti Profinit probíhá v současnosti v určitém souladu s navrhovaným stavem (2.5).

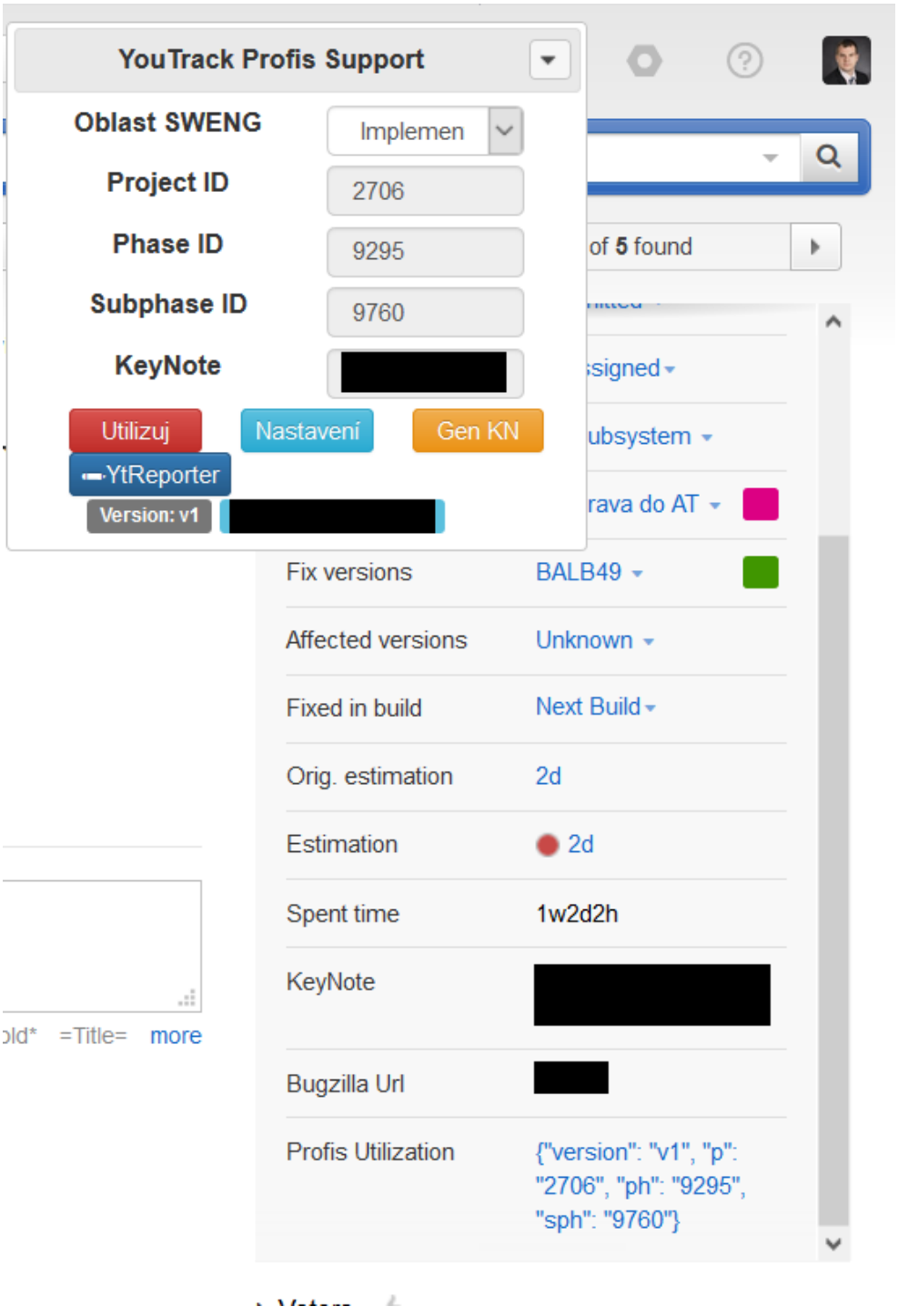
Na konci každého projektu vzniká dokument určený k jeho shrnutí a vyhodnocení. Každý dokument obsahuje základní informace o projektu, popis průběhu jednotlivých fází, měřená data a nakonec souhrn celého projektu.

Mezi základní informace o projektu patří údaje jako název, jeho stručný popis, výčet zodpovědných osob, odkazy na základní dokumenty apod.

Po úvodních informacích následuje textový souhrn jednotlivých fází projektu. Ten umožňuje popsat nastalé problémy či překážky, které ovlivnili průběh projektu, jejich důvody, případně prevence a varování k zabránění těchto problémů v budoucnu na jiných projektech. V této části se také vyskytuje seznam osob spolupracujících na dané fázi.

Dále je v souhrnu poskytnut náhled na naměřená data a statistiky. Obsahuje jednak velice důležité srovnání odhadu a výsledné pracnosti, následují souhrny testování (počet nalezených a opravených chyb v průběhu vývoje a akceptačního testování) a v neposlední řadě různé statistické zajímavosti jako počet řádků kódu, funkčních bodů apod. Typ statistických údajů se samozřejmě může lišit z hlediska typu projektu, vždy je snaha o zobrazení těch dat, která jsou pro daný projekt zajímavá.

Dokument je zakončen finálním shrnutím, které ukazuje hlavní přínosy projektu, problémy, na které při realizaci tým narazil, kvalitu součinnosti se



Obrázek 3.10: Youtrack Profis Support

zákazníkem, poznatky o používaných technologií a další podrobnosti, které nelze vyjádřit pouze měřenými daty.

3.4.1 Historie

Dokumenty popsané v předešlé kapitole 3.4 jsou využívány i pro historii projektů. Součástí každého souhrnu mohou a většinou bývají anotace, které umožní popsat použité technologie, business doménu a běžné problémy takovým způsobem, aby bylo možné podle těchto anotací projekty vyhledávat. Při vytváření nového projektu je pak možná inspirace staršími projekty podobného charakteru. Lze se tak poučit z chyb, předejít běžným problémům, napomoci vedoucím projektu lépe rozvrhnout kapacity a v neposlední řadě pomoci s tvorbou dalších odhadů.

Dokumenty jsou pak udržovány jednak v textové podobě, hlavně však v nástroji Confluence. Právě ten umožňuje anotaci jednotlivých dokumentů a jejich snazší vyhledávání.

Evidence historie v této formě přináší jak výhody, tak nevýhody. Výhodou těchto dokumentů může být to, že obsahují i textový popis, který vždy napoví více, než pouhá strohá data. Je možné přesně vědět, kde daný projekt narazil nebo naopak, co se ukázalo jako správná cesta v dané situaci a projektu prospělo.

Drobnou nevýhodou mohou být omezené možnosti hledání. Vyhledávání dokumentů v Confluence není příliš uživatelsky přívětivé, najít data vhodná k podpoře nových projektů může být složitý úkol. Bohužel však neexistuje příliš mnoho vhodnějších alternativ pro tento typ uchovávání a evidence dokumentů, který by zároveň umožňoval jednoduchou editaci, verzování, zajištění autorizace apod. Pokud ale uživatel dokáže akceptovat nepříjemnou syntax vyhledávacích query systému Confluence, má možnost čerpat z dnes již velice slušného množství dokumentů. Ty díky bohaté historii společnosti Profinit dávají k dispozici velice obsáhlou základnu všemožných vědomostí a historických zkušeností.

3.4.1.1 Databáze odhadů

S tímto souvisí také absence centrální databáze všech odhadů napříč společností. Každý tým si většinou udržuje vlastní repositář často s využitím verzovacích nástrojů jakými jsou SVN nebo GIT, nicméně přístup jednotlivých týmů k odhadům jiných týmů je většinou poněkud nepraktický.

Rozšíření využívání aplikace Estimate by tento problém vyřešilo, protože by veškeré odhady byly evidovány právě zde. Záleží pak na nastavení politiky, kdo a k jakým odhadům bude mít přístup. Díky importnímu modulu Estimate je pak možné do této databáze nahrát i starší odhady. Momentálně je podporován import nejrozšířenějšího typu odhadu (3.2.2), existuje však možnost navrhnout standardizovaný formát vstupních dat (například pomocí CSV či

XML souboru) a odhady neodpovídající podporovaným typům převádět právě do tohoto formátu.

Kapitola 4

Analýza a návrh

V této kapitole dojde k návrhu úprav stávajícího procesu (viz předešlá kapitola 3) softwarového vývoje s primárním zaměřením na zkvalitnění a zefektivnění odhadovacího procesu. Dále pak budou navrženy úpravy stávajících komponent nástrojů tohoto procesu a návrhy komponent nových, vycházející zejména z požadavků na úpravu vývojového procesu, ale i ze zpětné vazby z používání těchto nástrojů. Hlavním cílem je existence souhrnného seznamu potenciálních úprav s jasně definovanými přínosy. Z tohoto seznamu by pak na základě určených priorit mělo dojít k výběru komponent, které budou implementovány.

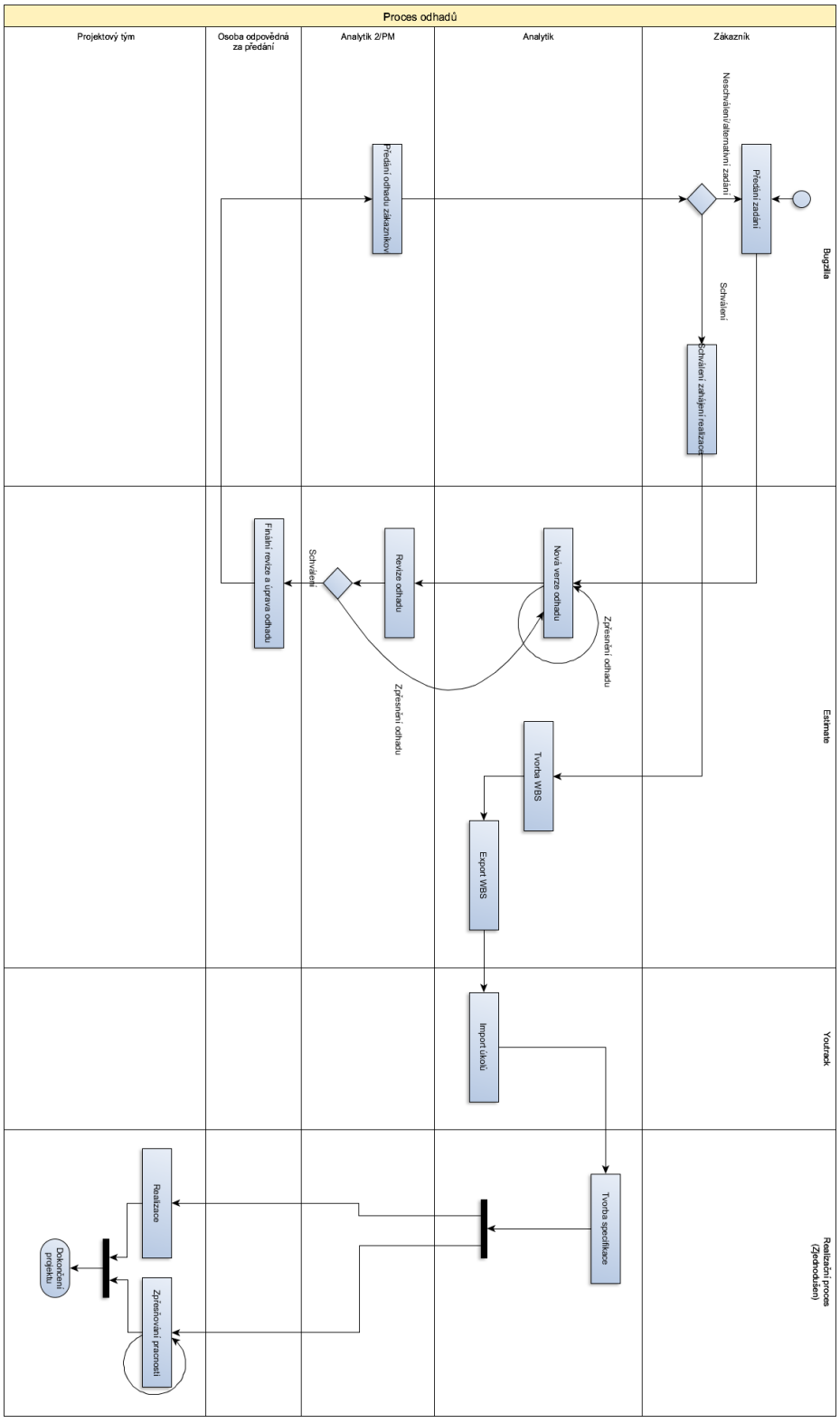
V průběhu analýzy a návrhu autor práce pravidelně konzultoval analyzovaný stav a navrhované změny s vedoucím práce, delivery managery a dalšími seniorními členy týmu. Díky tomu byla obdržena zpětná vazba k jednotlivým návrhům, zlepšeno povědomí o částech procesu, na kterých autor zatím neměl tu čest se podílet, a byly získány připomínky a nedostatky aktuálního procesu. Zároveň bylo možné udělat si obrázek o prioritě, s jakou by měly ty které požadavky být řešeny. Některé části procesu jsou i s určitými nedostatky dobře použitelné, pouze by různé úpravy mohly přinést nová a další zlepšení. Jiná místa aktuálního procesu však mohou s realizací návrhů profitovat prakticky ihned.

Diagram aktivit procesu, zahrnující změny navrhované v této kapitole, lze vidět na obrázku 4.1.

4.1 Úprava procesu odhadů a sběru metrik

4.1.1 Iniciální sběr požadavků

V oblasti sběru požadavků současný stav poměrně vyhovuje stavu kýženému. Místem k zamyšlení může být současné využívání dvou různých issue-tracking systémů, z nichž Bugzilla (viz 3.1) slouží právě ke sběru požadavků. Zároveň je to však jediný issue-tracking nástroj dostupný zákazníkovi, který slouží také jako místo pro reportování chyb. Díky tomu vzniká rozdělení reportovaných



Obrázek 4.1: Navrhovaný proces - diagram aktivit

chyb mezi dva různé systémy. V budoucnu může být cestou využití pouze jednoho nástroje, Bugzilla by pak mohla být využívaná pouze jako pomocný nástroj ke sběru požadavků, chyby by byly reportovány pouze v systému Youtrack.

4.1.2 Tvorba odhadu

Hlavní prioritou úpravy procesu tvorby odhadu je v současnosti provedení úprav nástroje Estimate tak, aby došlo k jeho rozšíření v celé společnosti. Momentálně je nástroj používán pouze zkušebně a odhady jsou stále převážně tvořeny za pomocí Excelové šablony (viz 3.2.2). Potřebné změny, které pomohou zajistit běžné používání této aplikace, jsou diskutovány v 4.2.1.

Samotný proces tvorby odhadu funguje na uspokojivé výši, existuje prostor k efektivnější tvorbě odhadů, jejich revizím a evidenci, ale to jsou všechno požadavky, které mohou být zajištěny podpůrnými nástroji.

4.1.3 Tvorba WBS

Jednou z možných úprav je zjednodušení procesu tvorby WBS a následného exportu úkolů do Youtrack. Jelikož je WBS svým způsobem pouze podrobnější odhad, bylo by možné využít odhadovacího nástroje Estimate, který již umožňuje verzování odhadů tak, že by WBS vznikaly právě v tomto systému. WBS by byla vytvářena stejně jako doposud v počátku realizace, tedy v době, kdy je již finální odhad schválen a nacenění projektu je předáno zákazníkovi.

Nebylo by tak nutné vytvářet WBS ve formě Excel dokumentu, ale k účelu by sloužil dedikovaný systém, ve kterém již navíc existuje odhad, ke kterému je WBS vytvářena. Při vhodně provedených úpravách systému Estimate by vytvoření WBS mělo být také méně časově náročné, například by nemuselo docházet k duplikacím úkolů, které se již dále nerozpadají. U těch úkolů, které se dále rozkládají, lze uživatelsky zjednodušit proces rozpadu.

4.1.3.1 Transformace nad WBS

Jak bylo uvedeno v analýze současného stavu (viz 3.3.3.1), existuje využití pro provádění automatické (nebo alespoň konfigurovatelné a manuálně spustitelné) transformace před dokončením WBS, které v závislosti na projektu upraví odhad tak, aby zohlednil určitý postup či metodiku.

Automatizace této transformace se jeví ideálně jako doplnění nástroje Estimate a o konkrétním návrhu řešení tak bude pojednáno v 4.2.1.8

4.1.4 Zjednodušení procesu exportu WBS

V současné době je proces exportu WBS složitý a využívá zbytečné množství různých nástrojů. Nabízí se možnost zjednodušení tohoto procesu pomocí přesunutí tvorby WBS do nástroje Estimate. Uživateli by tak, namísto zdlouha-

vého ručního transformování WBS do formátu vhodného pro použití k exportu a následného ručního provolávání webové služby pro export do Youtracku, bylo nabídnuto automatizování tohoto procesu. Výsledkem tohoto zjednodušení nebude pouze zefektivnění práce a snížení nákladů na řízení projektu, ale umožní delegování této činnosti, kterou momentálně vykonává výhradně řídící pracovník. Výstupem bude také minimalizace chyb, jejichž pravděpodobnost je při automatickém zpracování minimální v porovnání s ručním exportem.

Při exportu bude nutné k exportovaným položkám doplnit informaci o zakázce ze systému Profis. Tato potřeba vychází z nutnosti evidence této informace u položky v Youtracku pro zajištění navázání na Profis, například pro usnadnění utilizace pomocí doplňkových nástrojů (viz 3.3.4.3). Profis sice neposkytuje žádné API, které by umožnilo přístup k těmto položkám pomocí jednoduchého volání, nicméně lze zvolit cestu přímého přístupu do databáze tohoto informačního systému. Uživatel tak bude moci při exportu určit, ke které zakázce se dané položky váží a exportní komponenta se následně postará o zbytek.

4.1.5 Vazba issue na položku odhadu

Zavedení exportu ze systému Estimate by navíc umožnilo navázání realizovaného úkolu na odhad. To v současnosti možné není. Úkoly WBS jsou vytvářeny sice za pomocí odhadu, ale již neexistuje přímé navázání na položku, ze které byly tyto úkoly vytvořeny. Za pomoci vhodně zvolených identifikátorů, které by byly vedené u každé položky ve WBS a zároveň v Youtrack, je pak možné zpětně identifikovat korespondující položky odhadu. Protože navíc již existuje propojení mezi Profisem a položkami v Youtrack, bude tak zajištěná vazba jednotlivých položek napříč celým životním cyklem projektu.

To poskytuje velmi širokou řadu výhod. Bude tak možné sledovat nejen aktuální průběh projektu a dodržování pracnosti, ale i usnadní práci při finálním shrnutí projektu a následné historické evidenci všech projektů. Využitím těchto a dalších výhod se bude tato práce zabývat v pozdějších částech.

4.1.6 Anotace

4.1.6.1 Anotace položek

Další možnou úpravu, která se přímo týká exportu, je využití anotací. V současnosti již nástroj Estimate podporuje přidávání anotací k položkám odhadu ve formě tzv. tagů. Uživatel není volbou tagů nijak omezen, může tedy vytvářet libovolné tagy. Vhodným rozšířením by bylo umožnit napovídání tagů z množiny již existujících tagů. Tagy jsou ukládány společně s odhadem, ale zatím nemají žádné využití. Youtrack, do kterého budou úkoly exportovány, také umožňuje evidenci tagů u jednotlivých úkolů. Této funkcionality by tedy šlo využít a při exportu by došlo i k založení správných tagů.

Nemusí ale docházet pouze k obohacování o tagy definované uživatelem, ale mohou být také automaticky přidávány tagy aplikačně. Běžně je nyní například evidováno, jaké oblasti softwarového inženýrství se daný úkol týká. To je evidováno i v Estimate za pomoci kategorií, ale nebylo by příliš uživatelsky přívětivé chtít po uživateli doplňování kategorií u každé jednotlivé položky. Proto by se o to mohl starat Estimate sám. Šlo by tak například u každé kategorie nebo projektu definovat, jaké anotace budou doplněny ke každému úkolu v této kategorii, resp. projektu.

4.1.6.2 Anotace projektů

Není nutné omezovat se pouze na tagování položek, nabízí se také anotování projektů. Lze tak evidovat například použité technologie (programovací jazyky, frameworky, databázové servery) nebo třeba označit problémy, se kterými se projekt potýkal (nedostatek zdrojů, časová tíseň). Toto by mohlo sloužit při analýze historie a sběru metrik například k vyhledávání předchozích projektů se stejnými technologiemi, které by mohly sloužit jako inspirace pro další tvorbu odhadů. Díky anotacím jednotlivých úkolů by například mohl kdokoliv zjistit, kolik trvala tvorba jedné obrazovky projektu vytvořeného pomocí jazyka Java a technologie Spring. Možností je v tomto ohledu mnoho.

Tagování projektů v současné době není možné ani v Estimate, ale ani v Youtrack. Muselo by tedy dojít k rozšíření nástroje Estimate o podporu tagování projektů. V Youtrack lze využít popisu projektu, ve kterém by byla udržována informace o použitých anotacích ve vhodném formátu tak, aby bylo možné vyhledávat projekt pomocí REST rozhraní. Jinou možností může být vytvoření hlavního issue každého projektu, který by obsahoval právě projektově specifické anotace a další metadata, jako třeba celkovou odhadovanou a aktuální pracnost.

4.1.7 Kontrola průběhu realizace

Další prostor ke zlepšení se vyskytuje v procesu dohledu nad průběhem realizace projektu, zejména pak kontrolou dodržování pracností a časového plánu projektu. Provázáním úkolů s položkami v odhadu se nabízejí nové možnosti pro tvorbu realtime reportů a generování souhrnů.

V současné době lze využít nástroj YtReporter, který poskytuje přístup k REST API Youtracku a generuje přehled srovnání původních odhadů s již odpracovaným časem (viz 3.3.4.2). K zadávání reportů v nástroji YtReporteru je použit query jazyk Youtracku, uživatel jej však musí ovládat. Pomocí úprav uvažovaných v 4.2.3.1 je možné nároky na uživatele snížit.

4.2 Úprava používaných nástrojů

Tato část návrhu se bude zabývat úpravami stávajících nástrojů takovým způsobem, abychom mohli provést požadované změny procesu odhadů a sběru metrik definované v 4.1.

4.2.1 Estimate

V nástroji Estimate je nutné provést některé úpravy jednak pro zlepšení jeho použitelnosti pro současně podporované funkcionality, dále však také pro zajištění nových návrhů změn vývojového procesu.

Jedná se zejména o vyřešení problémů analyzovaných v 3.2.3 jako je použitá databáze a vylepšení použitelnosti, například úpravou některých GUI prvků.

Z hlediska zajištění podpory úprav vývojového procesu jde o návrh nových komponent pro automatický export WBS do systému Youtrack, zajištění vazby mezi issue v Youtracku a položkou odhadu a v neposlední řadě anotace položek.

4.2.1.1 Migrace databáze

Kvůli problémům se současně používanou grafovou databází (viz 3.2.3) bylo rozhodnuto o přechodu z používané grafové databáze Neo4j na databázi relační. Kromě výkonnostních problémů stály za tímto rozhodnutím i další aspekty. Neo4j je sice z technologického hlediska velice zajímavou databází, nicméně skutečných expertů na NoSQL databáze je v současné době nedostatek a relační databáze je přeci jen pro většinu dnešních softwarových inženýru uchopitelnější a familiárnější technologií. Navíc hlavní důvod použití Neo4j, jímž byla podobnost modelované domény grafové reprezentaci, se ve výsledku neukázal jako natolik zásadní a doménu lze bez problémů namapovat na tradiční relační model.

4.2.1.2 Úpravy GUI

Další překážkou širšího používání aplikace Estimate jsou některé nedostatky v GUI. Jedná se zejména o časovou prodlevu při používání GUI, což je pochopitelně problém při přesvědčování uživatelů pro přechod z rychlého offline nástroje, kterým je šablona v MS Excel. Pro nápravu tohoto neduhu bude potřeba dlouhodobější proces a součinnost s uživateli. Tyto úpravy nejsou součástí rozsahu této práce a budou prováděny separátně. V rámci této práce však budou provedeny přípravy pro usnadnění tohoto procesu.

Jednou z těchto přípravných prací je upgrade použitého GUI frameworku Vaadin na verzi 8.3. Oproti doposud používané verzi 7 nabízí knihovna podporu Javy 8 včetně všech novinek, například v podobě rozšíření podpory funkcionálního programování. S novou verzí byla do knihovny zařazena i velká

změna používání komponent, proto se nejedná o upgrade triviální, ale vyžaduje přepsání rozsáhlých částí aplikace.

Upgrade Vaadinu s sebou přináší podporu nových nativních komponent, zejména pak stromové tabulky TreeGrid, pro kterou bylo doposud nutné používat separátní neoficiální komponentu.

4.2.1.3 Verzování odhadů

Estimate momentálně umožňuje odhady verzovat, je však zapotřebí provést některé změny, zejména pak umožnit tvorbu WBS. Zároveň s tím je třeba vyladit systém zamykání odhadů uživatelem.

4.2.1.4 Tvorba WBS

Jelikož lze na WBS pohlížet jako na odhad ve speciálním stavu, bude na ní takto pohlíženo i v rámci aplikace Estimate. Při tvorbě nové verze odhadu bude moci uživatel zvolit, zda tvoří WBS. WBS bude pak oddělena separátně, mimo standardní verze odhadu. K WBS bude možné přistupovat ze speciálního výčtu určeného pouze pro WBS. Je tomu tak z toho důvodu, že je nutné zamezit tvorbě odhadu z WBS, na druhou stranu je důležité umožnit další úpravy odhadu. WBS se tedy bude vázat vždy k jedné konkrétní verzi odhadu, ze které vznikla.

Zobrazení WBS bude podobné odhadu, avšak s drobnými odlišnostmi a speciálními funkcemi. Například volby pro transformaci, popř. export WBS, které jsou uvažovány dále, budou dostupné pouze pro WBS.

4.2.1.5 Automatický export WBS

Úprava umožňující export WBS do Youtracku vyžaduje vytvoření několika nových komponent do nástroje Estimate.

Exportní proces bude zahájen manuálně uživatelem nad verzí odhadu označenou jako WBS. Dojde k otevření dialogového okna, ve kterém uživatel zvolí, do kterého Youtrack projektu se má WBS vyexportovat. Dále bude mít možnost určit, zda a jakou transformaci nad WBS provést (více v 4.2.1.8). Poté provede přiřazení jednotlivých kategorií WBS na odpovídající části zakázek v systému Profis. To je nutné pro zachování integrace mezi systémem Youtrack a vykazovacím modulem IS Profis. Nebylo by nutné provádět mapování v aplikaci Estimate, ale zbaví to tak uživatele odpovědnosti přiřazovat projekty manuálně, což ušetří poměrně velké množství času.

Po přiřazení dojde k zobrazení kontrolního okna s finální WBS již přímo určenou k exportu, tak aby uživatel mohl potvrdit požadovaný stav. Po potvrzení dojde k exportu, při němž již nebude vyžadován po uživateli žádný vstup. Po dokončení bude uživatel informován o stavu.

Finální proces exportování odhadu pak lze vidět na obrázku 4.2.

4.2.1.6 Vazba issue na položku odhadu

K zajištění dohledatelnosti položky odhadu, ke které se vztahuje dané issue, je možné využít uložení identifikátoru u issue v Youtracku. Metadata pro issue jsou v Youtracku vysoce customizovatelná, takže není problém jednoduše přidat novou položku pro držení tohoto identifikátoru.

Zde se pak nabízí několik možností. Buďto může dojít k použití stávajícího id záznamu v Estimate, vytvoření nového id určeného primárně k zajištění tohoto propojení, anebo využití keynote, který se již používá k propojení položek mezi Youtrackem a Profisem. Zároveň je již doplňován do stávajících WBS. Každá z možností má svoje pro a proti.

Stávající id záznamu ze systému Estimate by mohlo být využito, ale jelikož se jedná pouze o umělý klíč bez jakéhokoliv business významu, nemusí se jednat o příliš dobrý nápad. Na druhou stranu již ale id existuje, takže je poměrně jednoduché na implementaci. Stačilo by do systému Youtrack přidat nové pole, které by obsahovalo toto id. Navíc by pak provázání položky mezi odhadem a Profisem mohlo být řešeno pouze skrze položku v aplikaci Youtrack.

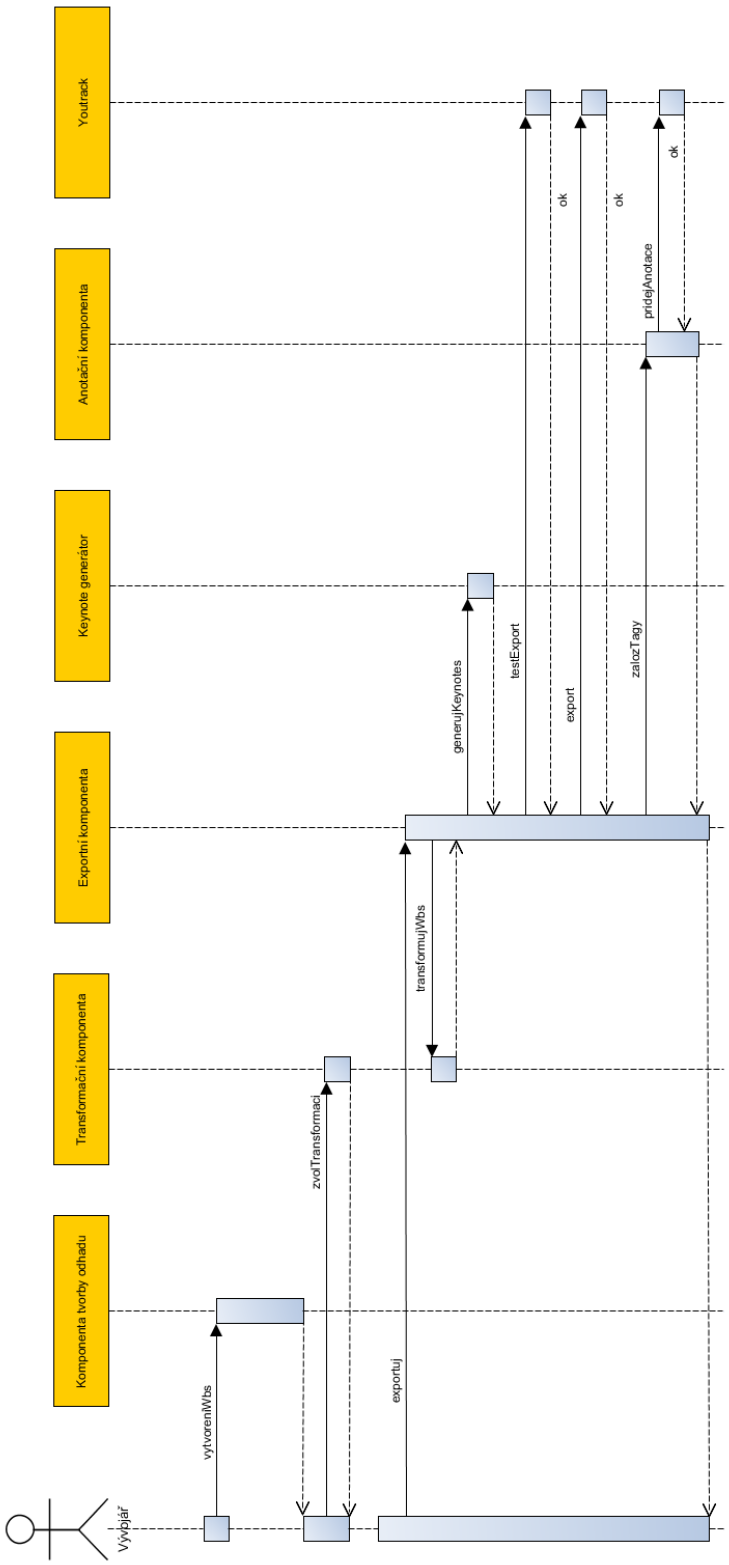
Nové id záznamu ze systému Estimate a jeho použití by obnášelo jednak rozšíření datového modelu této aplikace a znovu také rozšíření polí v Youtracku. Opět by ale nastal problém se zajištěním vazby mezi Estimate a Profisem.

Keynote je již existující záznam, který provazuje systémy Youtrack a Profis, navíc historicky používaný již v projektových plánech. Již tedy je používán jako jednoznačný identifikátor položky realizace. Může mít také business význam, například podle prefixu lze často identifikovat projekt nebo část projektu, nejedná se však o pravidlo, nelze se tedy na toto spolehnout. I tak se ale jedná pravděpodobně o nejlepší možnost zajištění vazby položky odhadu napříč celou infrastrukturou vývojového procesu.

Rozhodnutí pro využití keynote sebou obnáší přesun zodpovědnosti za jeho tvorbu. Až doposud byl keynote vytvářen, případně generován, vývojářem při tvorbě WBS nebo issue v Youtrack. Takto by byl tvůrce WBS nucen vymýšlet nebo generovat keynote a ten by byl vytvořen automaticky před exportem WBS z Estimate do Youtrack. Jednalo by se tedy o další rozšíření exportní komponenty.

4.2.1.7 Anotace

Anotace položek je již umožněna jak v systému Estimate, tak v Youtracku, tudíž je potřeba hlavně zajistit správný import těchto tagů. Bohužel Youtrack API v současné verzi neumožňuje import položek zároveň s jejich tagy a je



Obrázek 4.2: Exportní proces

tak potřeba využít separátní volání určené k aplikaci jakéhokoliv příkazu nad daným issue, tedy i přidání tagů. Problémem však může být absence možnosti volání nad více položkami najednou. To je možné obejít duplikací volání pro každé issue, které se však může podepsat na době trvání exportu odhadu. Export však nebude prováděn v takové míře, aby jeho případný slabší výkon měl nějaký znatelnější dopad. Bylo by však vhodné výsledný dopad zhodnotit. Jelikož je systém Youtrack stále aktivně vyvíjen, lze do budoucna doufat v rozšíření Youtrack API o možnost zjednodušující zavádění anotací do projektu.

4.2.1.8 Transformace WBS

Automatickou transformaci WBS je nutné zajistit během tvorby WBS, tudíž by komponenta starající se o tuto funkčnost měla být součástí nástroje Estimate.

Existují různé části procesu tvorby WBS, kde by mohlo dojít k vsunutí této funkcionality. Jednak by se mohla transformace provést na vyžádání uživatelem v rámci tvorby WBS. Uživatel by vybral transformaci z předem nakonfigurovaných možností a potvrdil by její provedení. Díky možnosti zpět v aplikaci Estimate, by bylo možné transformaci vzít zpět. Alternativou je provádění transformace během exportu. Jedním z kroků exportu by bylo zvolení transformace, která by poté byla automaticky provedena. Uživatel by mohl stav zkontrolovat a případně odmítnout, v tomto stádiu by však již bylo obtížné manuálně upravovat WBS po transformaci.

Z tohoto důvodu bylo rozhodnuto, že lepší alternativou je separátní volba provedení transformace na odhadu. Další výhodou tohoto přístupu je, že je tak možné jednoduše provádět více různých transformací nad stejnou WBS.

Konfigurovatelnost a tvorba jednotlivých transformací není jednoduchý problém. V rámci této práce bude navrženo konkrétní provedení jedné transformace, s kterou je autor obeznámen nejlépe. Po její realizaci a odladění je možné uvažovat rozšíření o obecný postup.

Konkrétním navrhovaným procesem je vymezení času určeného k akceptačnímu testování a opravování souvisejících nalezených chyb. Proces probíhá tak, že z implementačních resp. testovacích úkolů je odebrán podíl pracnosti. Tento podíl je sečten a výsledná pracnost je vyhrazena do separátního úkolu (pro každou z obou kategorií je určený jeden), který slouží právě pro tuto akceptační fázi. V závislosti na okolnosti projektu je možné tyto zvláštní položky ještě dále členit, pro účely tohoto pilotního testování ale tato úprava prováděna nebude.

4.2.2 Youtrack

Dalším polem možných úprav je aplikace Youtrack. Jednoduché změny a konfigurace lze provádět přímo v administrátorském rozhraní, složitější změny pak použitím Workflow API.

4.2.2.1 Workflow API

Youtrack Workflow API je rozhraní umožňující tvorbu rozšiřující logiky do systému Youtrack. Logiku lze přidávat pomocí tzv. workflows. Workflow je jednotka sdružující související funkcionalitu. Youtrack v základní instalaci již obsahuje spousty workflows, které je možné nakonfigurovat na úrovni jednotlivých projektů. Workflow se pak dále skládá z jednoho či více modulů, který již obsahují samotné skripty. Existuje několik typů modulů, v závislosti na požadovaném chování. Jedná se například o moduly, které jsou spouštěny při změně na issue, o moduly pouštěné v periodickém časovém intervalu, ale lze konfigurovat i vlastní akce či stavové automaty.

Vlastní skripty jsou psané v jazyce Javascript. Šablona modulu pak nabízí možnosti definování spouštěcí podmínky, ověřování stavu issue před jeho spuštěním, v případě automatů přechodové stavy a pak samotnou výkonnou funkci, ve které již lze psát požadované úpravy. Možnosti úprav jsou pak poměrně rozsáhlé, API poskytuje přístup prakticky ke všem typům úprav, které jsou dostupné z GUI.

Vytvoření vlastního workflow umožňuje vyřešit některé drobnější neduhy současného procesu, jako například nepropisování pole obsahující původní odhad na rodičovské úkoly. Implementací jednoduchého workflow, které by spočítalo rozdíl, o který je potřeba pozměnit tento údaj u nadřazeného úkolu, bude problém vyřešen.

4.2.3 YtReporter

Samostatnou kategorií možných rozšíření je nástroj YtReporter.

4.2.3.1 Úpravy zadávání query

Issue určená pro generování aktuálního reportu jsou vybírána na základě uživatelem zadaného query. Díky možnostem vystaveného API je možné zadávat query ve stejné podobě, jako je možné přímo v aplikaci Youtrack - query je voláním přeposláno tak, jak je. Jelikož lze jednotlivé části query poměrně jednoduše skládat do složitějších dotazů, nabízí se možnost rozšíření o automatickou tvorbu částí query, na základě uživatelsky příjemnějšího vstupu.

Jednou z možností je výběr oblasti softwarového inženýrství pomocí select boxu. Nejčastějším využitím YtReporteru bývá zobrazení pouze stavu implementačních úkolů současného projektu, kdy kategorie jako testování nemusí

být příliš zajímavé. Může tedy dojít k jejich vyfiltrování bez nutnosti pokaždé zadávat volbu manuálně textově.

Další možností je zavedení zkrácených zadání běžně používaných voleb, jako zobrazení pouze dokončených či nedokončených úkolů, seznam tagů používaných uživatelem (na toto lze opět krásně využít API aplikace Youtrack, které poskytuje endpointy na operaci s tagy) apod.

U všech těchto změn pak stojí za zvážení přínos oproti ručně zadávaným query. Navzdory výše vypsaným výhodám lze očekávat, že uživatel používající nástroj YtReporter zná Youtrack natolik, kdy možnost zatrhnout položku ručně příliš času a efektivity nepřidá. Význam by to mělo pouze v případě obskurnějších filtrů, které nemusí být všem uživatelům známé.

4.2.3.2 Export reportů

Dalším potenciálním vylepšením je export reportů ve formátu vhodným pro další zpracování. Ideálním typem reprezentace těchto dat je CSV formát. Export bude proveden vždy nad zobrazovanými daty (tedy jedná se o data pro aktuálně zadanou query). Do budoucna je možné zpřístupnit další formáty a reprezentace (například export pouze souhrnu nikoliv jednotlivých úkolů, ve formě HTML tabulky určené pro vložení do historie projektu apod.).

Kapitola 5

Realizace

V této kapitole budou zdokumentovány změny prováděné na dříve uváděných a aktuálně používaných nástrojích. Všechny tyto změny vychází z kapitoly návrhu (4).

5.1 Popis použitých technologií

5.1.1 Java

Programovací jazyk Java je jedním z nejrozšířenějších a nejpoužívanějších ([8]) programovacích jazyků vůbec. Jedná se o objektově orientovaný jazyk, který je díky využití virtuálního stroje rozšiřitelný na mnoho platforem. Díky své popularitě existuje velká spousta různorodých frameworků a knihoven, které tento jazyk podporují. Java je zároveň nejpoužívanějším jazykem ve společnosti Profinit, tudíž je pochopitelné, že bude využíván i v rámci tvorby interních nástrojů.

5.1.2 Spring Boot

Spring Boot ([9]) je rozšířením velice populárního Spring frameworku. Hlavním cílem je dosáhnout běžící aplikace při potřebě minimálního množství konfigurace. Spring Boot proto v základu poskytuje věci jako vestavěný aplikační server, základní bezpečnostní modul, apod. Zároveň však neochuzuje o žádné funkcionality Spring frameworku, stačí přidat požadovaný modul jako závislost (například pomocí Maven) a Spring Boot danou funkci zaktivuje za použití nejzákladnější konfigurace, kterou tak není nucen specifikovat vývojář (avšak může ji kdykoliv jednoduše upravit).

Spring Boot je použit pro tvorbu aplikace Estimate a velice usnadnil vývojový proces původní implementace i nových úprav. Kromě standardního využití dependency injection byly ze Spring frameworku použity zejména moduly pro práci s JPA, Security, LDAP a další.

5.1.3 Hibernate

Hibernate ([10]) je knihovna pro zajištění objektově relačního mapování neboli ORM. ORM usnadňuje práci s relačními databázemi tak, že jednotlivé entity z datového modelu zprostředkovává jako Javovské (nebo jiné, v závislosti na jazyku) objekty. Samotný Hibernate pak je jednou z nejznámějších a nejpoužívanějších implementací ORM v Javovském světě. Zároveň splňuje specifikaci JPA. Hibernate bylo využito pro migraci datového modelu aplikace Estimate do relační databáze.

5.1.4 Vaadin

Vaadin ([11]) je framework pro jazyk Java určený k tvorbě tzv. bohatých internetových aplikací (z anglického Rich Internet Applications neboli RIA). [12] udává, že hlavními odlišnostmi mezi tradiční webovou aplikací (tenkým klientem) a RIA je zejména v částečném zapojení klientské části ve výpočetní logice, vykreslování stránek, ukládání dat apod. V případě tenkého klienta se o toto stará vždy server, u RIA aplikací je zodpovědnost za tyto akce sdílena. Výhodami RIA řešení pak dle ([13]) jsou odložení části výpočtů na klienta, a tím pádem rychlejší odezva a komunikaci mezi serverem a klientem, šetření objemu přenášených dat a v neposlední řadě širší možnosti uživatelské interakce a prezentace (single page aplikace).

Při použití frameworku Vaadin je GUI vytvářeno přímo v Javě, při sestavování projektu pak dochází k jeho překládání do klienta v Javascriptu. Vaadin pro tento překlad využívá GWT (Google Web Toolkit).

Při realizaci byl použit Vaadin ve verzi 8.3, na který se přecházelo z verze 7. Mezi hlavní změny patří podpora Javy 8 a její využití v poskytovaných rozhraních.

5.1.5 Groovy

Groovy je programovacím jazykem postaveným nad virtuálním strojem jazyka Java. Díky tomu je sémantika jazyka Javě velice podobná, dokonce většina (až na určité výjimky) kódu určená pro Javu je validním kódem také pro Groovy. Mezi výhodami tohoto jazyka je jeho zvýšená expresivita, dynamické typování a některé další aspekty funkcionálního programování, které zvyšují jeho využitelnost k rychlé tvorbě užitečných nástrojů a utilit. Těchto vlastností lze využít pro zaplnění různých hluchých míst v procesu vývoje, tvorbě potřebných převaděčů dat pro usnadnění importu/exportu, apod. Ve stávajícím procesu byl využit zejména pro tvorbu YtReporteru a několika nástrojů k usnadnění projektového řízení (například konverze dat pro export WBS do YtReporteru).

5.1.6 Git

Git je verzovací systém určený primárně k verzování zdrojových kódů softwaru. Dle průzkumu prováděného společností Eclipse ([14]) se v roce 2014 jednalo o nejpoužívanější verzovací systém vůbec. Autor práce odhaduje, že se jedná o trend vzestupný a i dnes se bude jednat o jedničku mezi verzovacími systémy. Git ([15]) byl vytvořen Linusem Torvaldem a původně byl určen k verzování Linuxového jádra. Hlavním cílem bylo vytvoření takového verzovacího nástroje, který bude plně distribuovaný, rychlý, vcelku jednoduchý, bude umožňovat existenci velkého množství paralelních větví a v neposlední řadě bude efektivní i pro používání na velkých projektech (čímž Linuxové jádro bezesporu je).

Při používání Gitu se lze potkat se všemi standardními principy známými i s jiných verzovacích systémů, jako jsou větve, tagy apod. Pracuje se zde s nimi obdobně.

5.2 Estimate

5.2.1 Migrace na relační databázi

Jednou z prvních prováděných změn vůbec byla migrace původně použité grafové databáze na databázi relační. Stávající grafová databáze byla Neo4j, jejíž využití bylo umožněno frameworkem Spring, resp. jeho modulem Spring Data. Díky stávajícímu preciznímu dodržování MVC hierarchie a separaci datové vrstvy byla výměna využívané databáze celkem přímočará.

Došlo k rozhodnutí využít ORM z důvodu snadného přechodu z původního řešení Neo4j, kdy mapování objektů na grafovou databázi pomocí frameworku Spring se podobá využití ORM. Jako konkrétní implementace byla zvolena Hibernate se snahou o využívání pouze JPA API, tzn., byla co největší snaha o to, aby aplikace nebyla závislá na konkrétní implementaci. Nicméně na určitých místech bylo nutné od této filozofie ustoupit za účelem optimalizace výkonu.

Pro testovací účely byla využita vlastnost Hibernate pro automatickou tvorbu datového modelu v databázi na základě ORM konfigurace u modelových tříd. Z takto vytvořené databáze lze jednoduše (při použití vhodného nástroje) získat DDL skripty. Ty lze po úpravách přepsat na databázi, která bude využívána v ostrém provozu. Momentálně byla pro zajištění funkčnosti migrace používána pouze in-memory databáze H2.

Zároveň došlo k úpravě skriptů s testovacími daty tak, aby byly použitelné pro relační databázi.

5.2.2 Úprava GUI

Další nutnou úpravou, která patří mezi hlavní překážky pro rozšíření využívání nástroje Estimate, je vyhlazení nedostatků v GUI. V rámci této práce byla provedena příprava na tento proces, kterým byl zejména upgrade frameworku Vaadin na novou verzi 8.3.

5.2.2.1 Upgrade Vaadin

V rámci úprav GUI bylo zároveň rozhodnuto o upgrade na Vaadin 8, jednak z důvodu umožnění použití Javy 8, dále však také kvůli tomu, že Vaadin 8 přináší některé nové komponenty, které v předchozí verzi chyběli a jsou vhodné pro použití v aplikaci.

Používání novinek z Javy 8 umožňuje tvorbu úhlednějšího a kompaktnějšího kódu. Například vytvoření jednoduché Grid komponenty lze díky referencování metod napsat mnohem úsporněji. Ve Vaadin 7:

```
Grid grid = new Grid();
grid.setContainerDataSource(new BeanItemContainer<>(persons));
grid.removeAllColumns();
grid.addColumn("firstName");
grid.getColumn("firstName").setHeaderCaption("First Name");
grid.addColumn("lastName");
```

Ve Vaadin 8:

```
Grid<Person> grid = new Grid<>();
grid.setItems(persons);
grid.addColumn(Person::getFirstName).setCaption("First Name");
grid.addColumn(Person::getLastName).setCaption("Last Name");
```

Zároveň si lze všimnout, že došlo k přidání možnosti doplnění typového argumentu u jednotlivých komponent, což v předchozích verzích frameworku chybělo. To přináší další kontrolu vývojáře a prevenci chyb.

Jedná se především o komponentu TreeGrid, která umožňuje tvorbu klasické tabulky nad stromovými daty. To se hodí zejména pro zobrazování kategorií odhadů. Doposud byla používána rozšiřující komponenta z Vaadinovského repozitáře. Jelikož nešlo o nativní komponentu, přinášelo toto řešení určité problémy, kterých se takto lze zbavit. Vzhledově jsou však komponenty víceméně totožné (k náhledu na obrázku 5.1), rozdíly jsou zejména ve způsobu použití a v některých funkcionalitách.

5.2.3 Export WBS

V této části budou shrnuty všechny prováděné úpravy pro vytvoření části aplikace Estimate, která se bude starat jednak o vytvoření WBS přímo v rámci

Project Name	Hours Done	Last Modified
Year 2010	86	Sun Aug 08 00:00:00 GMT 2010
Year 2011	132	Fri Dec 02 00:00:00 GMT 2011
Year 2012	261	Fri Dec 07 00:00:00 GMT 2012
Year 2013	268	Wed Oct 02 00:00:00 GMT 2013
Year 2014	243	Wed Nov 05 00:00:00 GMT 2014
Year 2015	327	Wed Dec 02 00:00:00 GMT 2015
Year 2016	143	Thu Dec 01 00:00:00 GMT 2016

Obrázek 5.1: Treegrid komponenta - Vaadin 8

Estimate, dále pak o jeho vhodné pozměnění pro účely exportu do issue-tracking systému a o export samotný. Návrh této funkcionality byl proveden v 4.2.1.

5.2.3.1 Navázání na Profis projekt

V procesu exportu bude uživatel vyzván pro navázání jednotlivých kategorií na projekt v informačním systému Profis. V již otevřeném dialogovém okně dojde k zobrazení jednotlivých kategorií ve stromové hierarchii. Pro každou kategorii pak bude moci zvolit zakázku, část zakázky a podčást zakázky, které korespondují s položkami ze systému Profis. Pro zobrazení těchto položek je získat data z Profisu, kvůli neexistenci webového rozhraní je nutné přistupovat přímo k databázi.

Bude tedy vytvořená nová služba, jejímž hlavním účelem bude načtení celé struktury zakázek a podzakázek z Profis databáze. Jelikož se jedná o systémy na firemním intranetu, není potřeba řešit separátní autentizaci uživatele a bude se spoléhat na existující přihlášení do aplikace Estimate. Pro samotný přístup do databáze pak bude používán nový databázový účet, který bude mít práva pouze pro čtení. Tento účet bude napevno nakonfigurován a sdílen napříč nástrojem Estimate. Zároveň není nutné starat se o autorizaci, protože viditelnost jednotlivých projektů není schovávána za uživatelská oprávnění. V případě budoucí potřeby je však vždy možnost autorizační logiku doimplementovat.

Údaje o odpovídající zakázce budou uloženy a při serializaci do webové služby budou zpracovány spolu se zbytkem WBS.

Tato komponenta není pro funkčnost samotného exportu zásadní, zbaví však uživatele nutnosti zadávat vazbu na Profis projekt manuálně po exportu.

5.2.3.2 Generování keynote

V další fázi procesu exportu dojde ke generování keynotes. Každý keynote se bude skládat z prefixu, následovaného číslem. Prefix bude unikátní v rámci

WBS a kategorie. Bude složen ve formátu:

$$PK\text{-}PP\text{-}K\text{-}$$

, kde *PK* a *PP* označují business id projektu resp. jeho podčásti v aplikaci Estimate a *K* zkratku kategorie, do které daná položka spadá. Číslo následující prefix bude generováno pro každou položku. Na konkrétní hodnotě příliš nezáleží, nutné však je, aby se jednalo o hodnotu unikátní. Nejjednodušším řešením tak bude použít pořadí v rámci dané kategorie.

Generování keynote je zásadní pro zachování funkčnosti vykazování času. Manuální zadávání po exportu již nepřichází příliš do úvahy, protože by musel být zapisován keynote pro každou jednotlivou položku zvlášť. Jedinou možnou alternativou, pokud tato komponenta ještě nebyla dostupná, je manuální zadávání keynote před exportem, například pomocí přidání nového sloupce u položky odhadu, nicméně ušetřená pracnost je zanedbatelná a tudíž by tato komponenta měla být implementována s vyšší prioritou.

5.2.3.3 Serializace odhadovaných položek

Před odesláním dat WBS do Youtracku je nutné tato data vytvořit. Protože nelze použít klientské API (viz 5.2.3.4), které by již obsahovalo předpřipravené třídy usnadňující použití, postará se o to další z komponent exportního modulu. Ta z datového modelu WBS vytvoří XML data připravená k exportu. Bude využita knihovna Jackson, tedy jedna z nejrozšířenějších knihoven pro XML serializaci a deserializaci pro Javu.

Použití této knihovny je velice jednoduché. Struktura vytvářeného XML je generována přímo z existujících Java bean tříd. Zjednodušenou ukázku bean třídy pro položku odhadu lze vidět na 5.1.

Kód 5.1: Jackson bean

```
public class ItemBean {
   private Long numberInProject;

   private String description;

   //Both estimates are represented by string which corresponds with
       Youtrack API.
   private String originalEstimate;

   private String estimate;

   public Long getNumberInProject() {
      return numberInProject;
   }

   public void setNumberInProject(Long numberInProject) {
```

```
        this.numberInProject = numberInProject;
    }
    //Rest of getters and setters
}
```

Jednotlivé beany samozřejmě mohou obsahovat beany jiné. Pro import projektu bude tedy existovat jediná instance bean třídy Project, která bude obsahovat položky odhadu a metadata. Po naplnění celé struktury daty z odhadu pak stačí zavolat odpovídající logiku Jackson knihovny, která se postará o zbytek a pomocí properties dané bean třídy vygeneruje odpovídající XML strukturu (5.2).

Kód 5.2: Jackson serializace

```
public String serializeToXml() {
    XmlMapper xmlMapper = new XmlMapper();
    return xmlMapper.writeValueAsString(new ProjectBean());
}
```

String vrácený z této metody je pak XML, určené ke zpracování exportní službou.

5.2.3.4 Export

Samotný export bude probíhat pomocí vystaveného REST API aplikace Youtrack. Bohužel v současné době neexistuje kompletní klient dostupný pro jazyk Java, který by byl použitelný pro naše účely, takže je nutné vytvořit vlastní zjednodušenou implementaci. Bude obsahovat pouze takové části API nutné pro zajištění této funkcionality, převážně se jedná o součásti zaměřené na import issue do Youtracku.

Jelikož vystavené API umožňuje také nahrání dat v testovacím režimu, dojde k využití tohoto testovacího režimu, kdy nejprve dojde ke kontrole, že je umožněno nahrání všech exportovaných úkolů a pouze v případech, kdy nedojde k žádné chybě, bude proveden „ostrý“ export. Minimalizuje se tak riziko nekompletního importu do systému Youtrack. Ten je v tomto případě nežádoucí, neboť není zájem umožňovat nahrávání jednotlivých issue pro jejich nápravu. WBS tak bude vždy nahrána buďto kompletní, nebo nebude nahrána vůbec.

V průběhu tvorby této práce docházelo k postupnému přechodu projektu, na kterém byl proces laděn, do jiné instance aplikace Youtrack, která je společná pro celý Profinit. Ukázalo se, že uživatelská práva Youtracku nutná pro import úkolů přes poskytované API, bohužel nelze vystavit běžnému uživateli. Export pomocí aplikace Estimate však tento problém vyřeší. Bude vytvořen technický účet, který bude používán pro export, a uživatel tak bude od nutnosti vlastnictví potřebných práv odstíněn. Bude docházet pouze k ověření

práv na projekt, což lze vyřešit opět díky LDAP účtu, který je totožný pro Estimate i Youtrack.

Než však bude exportní funkcionalita implementována, může docházet ke zpomalování procesu exportu WBS. Je proto vhodné vytvořit proxy, ke které bude moci podobně, jako v případě exportní komponenty aplikace Estimate, přistupovat uživatel přímo. Proxy pak po základním ověření uživatelských práv požadavek převolá pod technickým uživatelem do Youtracku. Tato proxy bude jednodušší na implementaci, takže může být vytvořena poměrně rychle a umožnit tak efektivní fungování procesu do té doby, než bude dokončena tato funkcionalita v aplikaci Estimate. Zároveň může proxy posloužit jako základ pro finální implementaci v Estimate.

5.2.3.5 Anotace

Po dokončení exportu je nutné provést zanesení anotací, které, kvůli omezením API Youtracku, nelze importovat společně s odhadem.

5.2.4 Transformace WBS

Následující realizace vychází z návrhu 4.2.1.8, ve kterém bylo uvedeno, že pro zkušební účely bude naimplementována jedna konkrétní transformace popsaná právě v této části.

V první řadě je nutné vytvořit nové tlačítko v GUI určené k výběru transformace. Po jeho aktivaci dojde k otevření modálního okna, ve kterém bude možné zvolit transformace ze seznamu.

Po zvolení transformace a jejím potvrzení, dojde k jejímu provedení. Implementace této transformace bude zajištěna implementací nové služby v kódu aplikace.

Transformace bude možné konfigurovat na projektový kontext, uživateli se budou u odhadu zobrazovat jen takové transformace, které jsou pro daný kontext nakonfigurované. Konfigurace transformací na kontext bude v první fázi zajištěna staticky, po implementaci konfigurovatelných transformací bude možné jednotlivé transformace nastavovat s vhodným oprávněním z aplikace Estimate.

Tato úprava není pro funkčnost a zavedení aplikace Estimate zásadní. Momentálně jsou tyto úpravy prováděny manuálně a není nejmenší problém je takto provádět dále, a to i v aplikaci Estimate. Implementací této funkcionality by však jednak došlo ke značné úspoře času, ale hlavně by tento proces byl imunní vůči uživatelské chybě. Nicméně momentálně tato úprava nebyla naimplementována a je s ní počítáno spíše do pozdějších fází.

5.3 Youtrack

5.3.1 Workflow

Jak bylo nastíněno v návrhu (4.2.2.1), pomocí workflow lze řešit drobné neduhy a nedostatky konfigurace aplikace Youtrack.

Jedním ze zmiňovaných problémů, bylo propisování odhadovaného času k hierarchicky nadřazeným úkolům. Vlastní pole s původním odhadem není propisováno automaticky, workflow je proto ideálním kandidátem na úpravu tohoto chování. V rámci implementace tedy vzniklo nové workflow (5.3), navázané na změnu issue. Toto workflow zjistí, zda při změně došlo k modifikaci hodnoty pole originální pracnost. To by se sice nemělo měnit, ale jelikož workflow API bere jako změnu i vytvoření úkolu, nevadí to. Pokud ke změně času došlo, spočítá rozdíl původní a aktuální hodnoty a tento rozdíl odečte/přičte k tomuto poli u rodičovského úkolu. Tím je zajištěno propsání času výše. Jedná se o jednoduchý skript, který však usnadní spoustu práce při nahrávání a tvorbě nových úkolů.

Kód 5.3: Workflow - Update rodičovské originální pracnosti

```
/**
* Pravidlo navýší original estimate u~parenta podle subtasku.
*/
var entities = require('@jetbrains/youtrack-scripting-api/entities');
var dateTime = require('@jetbrains/youtrack-scripting-api/date-time');
exports.rule = entities.Issue.onChange({
   title: 'Update Original Estimate Parent',
   guard: function(ctx) {
      return ctx.issue.fields["Original Estimate"] &&
          ctx.issue.links['subtask of'].first();
   },
   action: function(ctx) {
      var issueFields = ctx.issue.fields;
      var parentFields = ctx.issue.links['subtask of'].first;

      var toMilis = function(period) {
         return period ? (1000 * (period.getSeconds() +
            60 * (period.getMinutes() +
               60 * (period.getHours() +
                  8 * (period.getDays() +
                     5 * period.getWeeks()))))) : 0;
      };

      var addPeriods = function(parent, per) {
         console.log(toMilis(parent));
         return dateTime.toPeriod(toMilis(parent) + toMilis(per));
      };

      var estField = issueFields['Original Estimate'];
```

```
    var parentEstField = parentFields['Original Estimate'];
    parentFields['Original Estimate'] = addPeriods(parentEstField,
        estField);
    },
  requirements: {}
});
```

Dalším možným workflow, jehož potřeba vyšla najevo v jedné z pozdějších fázích úprav procesu, je workflow umožňující podědění některých metadat úkolu. Jedná se zejména o informace nutné při využití nástrojů integrace Youtracku a Profisu, které je často zbytečné přepisovat, protože budou mít naprosto stejné hodnoty jako úkol rodičovský. Bude proto naimplementováno nové workflow, které přesně toto chování zajistí. Část logiky bude vycházet s předchozího workflow, dojde však k přepsání hodnoty, což bude implementačně jednodušší. Workflow bude spouštěno vždy při změně rodičovského úkolu a bude fungovat pouze na takových úkolech, u kterých dává kopírování těchto dat smysl. U ostatních úkolů toto workflow nebude spuštěno.

5.4 YtReporter

V rámci návrhu potenciálních změn aplikace YtReporter bylo uvažováno zejména doplnění automatického přidání částí filtrů na základě jednoduššího uživatelského vstupu, než je psaní celých dlouhých filtrů, například pomocí checkboxů.

V průběhu pravidelných konzultací s uživateli této aplikace se ukázalo, že jim chybí možnost filtrovat pouze takové úkoly, které jsou listy v dané hierarchické struktuře. Jedná se o velmi častý požadavek, jelikož YtReporter v současnosti nepodporuje zobrazení stromové struktury úkolů, na kterou se aktuálně přechází. Tím pádem dochází k pokřivení reportovaných hodnot. Uživatelé tak jsou nuceni ke psaní složitých filtrů na základě anotací ne-listových úkolů.

Ukázalo se, že sice existuje filtr umožňující vyselektovat pouze listové úkoly, ale je poměrně obskurní a v dokumentaci aplikace Youtrack špatně dohledatelný. Proto bylo rozhodnuto, že tato možnost bude ve formě checkboxu doimplementována do systému YtReporter. Po zakliknutí tohoto checkboxu bude uživatelem zadaný filtr na pozadí obohacen o podmínku, která vyselektuje pouze případy, kdy se jedná o listový úkol.

Aplikace YtReporter je složená z tenkého webového klienta, který přistupuje k RESTovému rozhraní Youtracku. Toto rozhraní je jádro YtReporteru a stará se o veškerou logiku. Bylo proto nutné rozšířit jednak samotného klienta, tak i aplikaci. Implementované řešení bylo navrhnuto a napsáno tak, aby mohlo být v budoucnu jednoduše rozšířeno o další filtry podobného charakteru. Pokud z dalšího běžného používání vyvstane nová potřeba, není problém chování filtrů rozšířit.

5.5 Verzování

Při vývoji systému Estimate je samozřejmě využíváno verzování. K jeho zajištění je používán jeden z nejrozšířenějších verzovacích systémů Git.

Je využíváno poměrně standardní nastavení. To předpokládá existenci hlavní větve, tzv. masteru. V závislosti na tom, jakou činnost chce vývojář provádět, pak dochází k vytvoření buďto tzv. feature nebo bug větve. Feature větev je využívána pro vývoj nových funkcionalit, naproti tomu bug větev se používá k opravě bugů. To, zda se jedná o feature či bug větev, je rozlišeno jejím pojmenováním, standardně se využívají prefixy „feature/“ a „bug/“. Po dokončení a otestování vývoje je větev zamergována zpátky do master větve. Takto je zajištěno, že vždy existuje stabilní verze, ze které je možné provádět další vývoj, testovat ji, apod.

V budoucnu pak dojde k využití dalších větví pro umožnění verzování nasazovaných verzí. V pilotním programu je zatím nasazena verze sestavovaná z master větve.

Ve společnosti Profinit je pro údržbu Git repozitáře používán systém GitLab. Ten pomocí webového rozhraní poskytuje jednak nástroje určené ke správě samotného repozitáře, členění, zakládání nových projektů, autorizace apod., dále však také různé podpůrné nástroje, například pro vytváření merge requestů, provádění revizí kódu, apod.

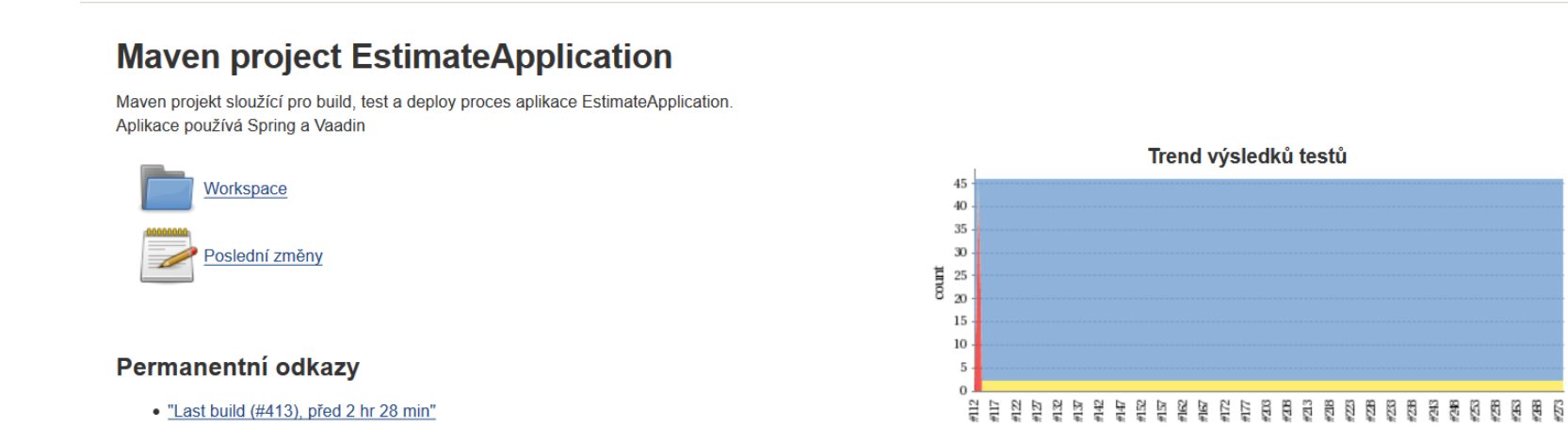
5.6 Řízení změn

Vývoj aplikace Estimate probíhá v souladu s vývojovým procesem společnosti Profinit, platí to tedy i s ohledem na řízení verzí. Pro aplikaci Estimate existuje ve firemním Youtracku samostatný projekt, ve kterém je udržován soubor úloh k dořešení. V průběhu analýzy byly úlohy průběžně aktualizovány a přidávány nové s ohledem na výstupy s analýzy.

5.7 Průběžná integrace

Další běžně používanou disciplínou softwarového inženýrství je využívání systému průběžné integrace (někdy také continuous integration). Ve společnosti Profinit se běžně používá software Jenkins. Pro aplikaci Estimate byl proto také využíván právě tento nástroj (5.2). V něm je vytvořen job (5.3), pouštěný každý den, který z master větve sestaví aplikaci Estimate, otestuje ji vytvořenými unit testy a nakonec nasadí na k tomu určený server. Celý průběh je samozřejmě logován, výsledný stav (pokud vše neproběhne v pořádku) je zaslán emailovou notifikací všem zúčastněným osobám.

Obrázek 5.2: Jenkins - přehled jobů



Obrázek 5.3: Jenkins - Estimate job

5.8 Testování

Prováděné úpravy systému Estimate byly pokryté existujícími i upravenými jednotkovými testy. Bylo tak zajištěno, že chování před i po úpravách bude totožné. V případech, kde došlo ke změně chování po úpravě, došlo k modifikaci testu, tak aby verifikoval chování nové funkcionality.

Závěr

V rámci této práce došlo k návrhu cílového stavu (2) vývojového procesu určeného pro prostředí společnosti Profinit. Navržený proces byl konzultován jak s vedoucím práce, tak i s každodenními účastníky tohoto procesu z řad delivery managerů a konzultantů.

Aby bylo možné vydefinovaný cílový stav aplikovat v kontextu společnosti, došlo následně k provedení analýzy aktuálně využívaného procesu (3). Byl zohledněn celý životní cyklus projektu od iniciálního sběru požadavků, přes tvorbu odhadu, realizaci, její monitoring, až po finalizaci, tvorbu historie a vyhodnocení projektu. Během procesu analýzy měl autor práce možnost potkat se s velkou řadou různorodých technologií, počínaje programovacími jazyky (Java, Groovy, Javascript), přes frameworky (Spring, Vaadin, Hibernate), SQL i NoSQL databáze (Neo4j), buildovací (Maven, Gradle, Ant) a verzovací (Git, SVN) nástroje, issue-tracking systémy (Bugzilla, Youtrack) a konče rozličnými podpůrnými nástroji.

Na základě výstupů této analýzy pak mohl být proveden návrh změn nutných k tomu, aby současný proces vývoje mohl dosáhnout stavu cílového (4). Kromě úprav samotného procesu bylo také zapotřebí se seznámit s existujícími podpůrnými nástroji, často do hloubky analyzovat jejich zdrojový kód tak, aby navržené úpravy mohly být co nejefektivněji implementovány. Dané se primárně týkalo nástroje Estimate, u kterého došlo k návrhu změn a funkcionalit ve větším rozsahu. Úpravy se dotýkají jednak samotné architektury a použitých technologií aplikace, tak i přidání nové exportní funkcionality, která značně zjednoduší nejen administrativní přechod od odhadu k realizaci projektu.

Díky autorově možnosti každodenní práce v jednom z týmů společnosti Profinit mohly být jednotlivé úpravy procesu vždy řádně vyzkoušeny v reálném vývojovém prostředí. Konkrétně pak byly změny zkoušeny na procesu vývoje jednoho z větších projektů, jehož pracnost se pohybuje ve stovkách MD. Změny tak musely být vždy zaváděny inkrementálně a iterativně, aby nedošlo k narušení běhu projektu. Úpravy, které se osvědčily, mohly být ponechány, u

ostatních byla zohledněna zpětná vazba a mohlo dojít k jejich pozměnění.

Návrhů změn, zároveň realizovaných v rámci této práce, bylo několik. V první řadě došlo k implementacím změn v aplikaci Estimate. Jednalo se zejména o migraci z grafové databáze Neo4j na tradiční relační databázi a s tím související zavedení frameworku Hibernate. Byl také proveden upgrade na nejnovější verzi RIA frameworku Vaadin, který přináší zejména podporu Javy 8 se všemi jejími novinkami. Ze změn, které nebyly prováděny pro systém Estimate, lze zmínit například implementace workflow do issue-tracking nástroje Youtrack, které usnadňují a automatizují některé části životního cyklu evidovaných položek. Byla také doplněna aplikace YtReporter o prvky uživatelského rozhraní usnadňující její použití.

Provedené úpravy se již dnes podílejí na zefektivnění a usnadnění vývojového procesu ve společnosti Profinit. Úpravy, které zatím nebyly implementovány, jsou navrženy takovým způsobem, aby byla budoucí implementace co nejjednodušší. Autor práce očekává, že po implementaci všech navrhovaných úprav dojde k dosažení cílového stavu v takovém rozsahu, v jakém byl navržen v této práci.

Literatura

[1] Crear, J.; Gesmer, L.; Johnson, J.; aj.: Chaos Manifesto 2013: Think Big, Act Small. Technická zpráva, Standish Group International, Inc., 2013.

[2] Atkinson, R.: Project management: cost, time and quality, two best guesses and a phenomenon, its time to accept other success criteria. *International Journal of Project Management*, 1999.

[3] Jones, C.: *Estimating Software Costs*. McGraw-Hill, 1998.

[4] Conte, S. D.; Dunsmore, H. E.; Shen, V.: *Software engineering metrics and models*. Benjamin-Cummings, 1986.

[5] McConnell, S.: *Software Estimation: Demystifying the Black Art (Developer Best Practices)*. Microsoft Press, 2006.

[6] Jarell, J.: Stories versus Themes versus Epics. 2014. Dostupné z: `https://www.scrumalliance.org/community/articles/2014/march/stories-versus-themes-versus-epics`

[7] Brooks, F. P.: *The Mythical Man-Month*. Addison-Wesley, 1995.

[8] TIOBE - The Software Quality Company: TIOBE Index. Technická zpráva, TIOBE - The Software Quality Company, 2018. Dostupné z: `https://www.tiobe.com/tiobe-index/`

[9] Webb, P.; Syer, D.; Long, J.; aj.: *Spring Boot Reference Guide*. 2018. Dostupné z: `https://docs.spring.io/spring-boot/docs/current/reference/htmlsingle/`

[10] Ottinger, J.; Linwood, J.; Minter, D.: *Beginning Hibernate*. Apress, 2016.

[11] Vaadin Ltd.: *Vaadin Documentation*. 2018. Dostupné z: `https://vaadin.com/docs/v8/framework/tutorial.html`

[12] Tůma, J.: RIA. 2012. Dostupné z: https://www.webcesky.cz/ria/

[13] Fraternali, P.; Rossi, G.; Sánchez-Figueroa, F.: Rich Internet Applications. *IEEE Computer Society*, 2010.

[14] Skerrett, I.: Eclipse Community Survey 2014 Results. 2014. Dostupné z: https://ianskerrett.wordpress.com/2014/06/23/eclipse-community-survey-2014-results/

[15] Chacon, S.; Straub, B.: *Pro Git*. Apress, 2014. Dostupné z: https://git-scm.com/book/en/v2

Příloha **A**

Seznam použitých zkratek

API Application programming interface

CSV Comma-separated values

CRM Customer relationship management

DDL Data definition language

GUI Graphical user interface

GWT Google web toolkit

HTML Hypertext markup language

JPA Java persistance API

LDAP Lightweight Directory Access Protocol

MD Man-day neboli člověko-den. Množství práce, vykonané člověkem za jeden den (v ČR nejčastěji 8 hodin čistého času).

MH Man-hour neboli člověko-hodina. Množství práce, vykonané člověkem za 1 hodinu.

MVC Model-view-controller

ORM Objektově relační mapování

REST Representational state transfer

RIA Rich internet application

XML Extensible markup language

WBS Work breakdown structure

Příloha **B**

Obsah přiloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `Estimate` ..... zdrojové kódy aplikace Estimate
- `text` ..... text práce
  - `src` ..... zdrojová forma práce ve formátu LaTeX
  - `thesis.pdf` ..... text práce ve formátu PDF
- `YtReporter` ..... zdrojové kódy aplikace YtReporter
- `update-original-estimate.zip` ..... workflow nástroje Youtrack