Semestrální práce ke kurzu 4IT421 Zlepšování procesů budování IS

Semestr	Zimní semestr 2017
Autoři	Daniel Nejezchleb, xnejd00 Filip Řeha, rehf01
Téma	Size Estimation Approaches for Use with Agile Methods
Datum odevzdání	31.12.2017

Abstrakt

Odhadování velikosti softwaru v agilním prostředí je důležitá součást při vývoji softwaru. Zpočátku se práce zabývá otázkou proč vůbec odhadovat, a dále rozebírá 3 základní principy odhadování. Následuje detailní rozbor nejpoužívanějších metod pro odhadování velikosti softwaru v agilním prostředí.

Klíčová slova

Agilní, odhadování, principy, metody

Celkový obsah

1 Úvod

Tato práce se zabývá přístupy odhadování velikosti softwaru používaných v agilním prostředí. Vychází zejména z práce (Reifer, 2017), ve které byl proveden průzkum 112 respondentů, a bylo hodnoceno 5 hlavních používaných metod. Zapojené organizace jsou z průmyslových oblastí jako automatizace, obrana, finančnictví, informačních technologie, telekomunikace.

Práce nejdříve podává odpověď na otázku, proč vůbec v projektech odhadujeme. Dále uvádí tři základní principy odhadování v agilním prostředí. Následně popisuje konkrétní metody odhadování velikosti softwaru, které byly uvedeny jako nejčastěji používané podle již zmíněného průzkumu. Na závěr hodnotí jejich využití a sděluje zajímavé poznatky.

2 Proč odhadujeme

Mezi agilními experty panuje kontroverze o tom, zda odhadování vůbec stojí za to dělat nebo zda třeba i neškodí.

Odhadování přináší hodnotu, když nám pomáhá udělat významné rozhodnutí. Příkladem odhadnuvši informovaného rozhodnutí je alokace zdrojů. Organizace mají většinou fixní množství peněz a lidí a obvykle se rozhodují do čeho tedy investují. U takovýchto rozhodnutí je užitečné znát kolik pracnosti a nákladů mě budou jednotlivé projekty stát a pro uskutečnění rozumného rozhodnutí potřebujeme mít alespoň náznaky nákladů a přínosů jednotlivých řešení. Dalším příkladem je pomoc s koordinací. Když jeden tým chce vydat novou funkcionalitu, ale nemůže, dokud druhý tým nedodá potřebná data. Takže jestliže první tým odhaduje, že mu vývoj nezabere déle než měsíc, přičemž druhému týmu to potrvá dva měsíce, první tým tedy ví, že není užitečné s prací začínat hned a může mezitím dodat funkcionalitu jinou. Tyto příklady mimo jiné poukazují na to, že jestliže žádáme odhady, měli bychom mít jasně specifikováno, jaké rozhodnutí jimi chceme podpořit. Jestliže žádné nemůžeme najít nebo rozhodnutí není významné, je na zvážení, zda vůbec něco odhadovat. Pochopení těchto rozhodnutí může také vést k úplně odlišným akcím, u nichž nebude třeba odhadů. Pro náš příklad to může znamenat, že třeba existuje způsob, jak by oba týmy mohly kooperovat a doručit funkcionalitu ještě dříve. Podobné je to se sledováním plnění plánu, které by mělo také být řízeno tím, jak to přispívá k rozhodovacímu procesu. Na plán by se mělo nahlížet jako na výchozí měřítko, které nám pomáhá s posuzováním změn. Odhady nám pak pomáhají pochopit kompromisy, a tak rozhodnout o reakci na změnu. Je ale nutné si pamatovat, že u sledování plnění plánu mají odhady limitovanou životnost. Odhady je dobré aktualizovat, jelikož po určitém čase ztrácejí na přesnosti s tím, jak se vyvíjí změny v projektu. Mnoho týmů shledává v odhadování užitečnost, protože pozitivně působí na komunikaci v nich. Odhadovací meetingy napomáhají lepšímu pochopení různých způsobů implementace nadcházejících stories, budoucím architektonickým směrům a problémům.

Odhadování není ani dobré ani špatné. Jestliže dokážeme pracovat efektivně bez odhadů, pak jen směle do toho. Jestliže myslíme, že potřebujeme nějaké odhady, pak se musíme ujistit, že rozumíme jejich roli v podpoře rozhodování. A když tyto odhady ovlivňují významná rozhodnutí, tak se pokusme udělat ty nejlepší odhady. Nadevše nutné je

vyvarovat se kohokoliv, kdo říká, že odhady jsou vždy potřeba nebo že nejsou nikdy potřeba. Veškeré argumenty týkající se odhadů by se vždy měli odrážet od agilních principů na základě kterých, bychom se měli rozhodnout, zda jsou to ty správné techniky pro náš případ (Fowler et al., 2013).

3 Základní principy agilního odhadování

V této kapitole jsou popsány 3 základní principy agilního odhadování (Berteig, 2014). Z nichž je hlavně princip relativního odhadování velmi dobře pozorovatelný v dále uvedených popisech konkrétních metod.

3.1 Princip kolaborace

Agilní přístup tkví ve spolupráci a u odhadů tomu není jinak. To znamená, že v agilních projektech spolupracuje několik lidí, aby přišlo na odhad pracnosti produktu. Tradiční techniky odhadování se většinou zaměřují na jednotlivcích, kteří odhadují práci, za kterou jsou oni zodpovědní. Kolaborativní odhadování znamená odhadování většinou na formálních schůzkách, kde lidé diskutují tváří v tvář. Princip spolupráce se obecně hodí do oblastí řešení problémů nebo objevování nových znalostí, jelikož různí lidé mají různé zkušenosti a znalosti. Největším důsledkem tohoto principu je, že není možné spojit finální odhad s konkrétní osobou, která by potom nesla zodpovědnost. Tento fakt vytváří pocit bezpečí a je nutný k tomu, aby se účastníci odhadování otevřeně a bez zábran vyjádřili, přestože by to mohlo odporovat očekávání stakeholderů. Jinými slovy řečeno, žádný jedinec nemůže být potrestán za špatný odhad.

3.2 Princip relativního odhadování

Nejdříve uvedeme příklad, na kterém si tento princip vysvětlíme. Představme si, že před námi stojí lahev s vodou a někdo se nás ptal, kolik je v ní vody. Mohli bychom se nejprve zamyslet nad celkovou velikostí sklenice a říct, že je z poloviny plná. A kdybychom byli dále dotazováni kolik přesně je tam vody, možná bychom si lahev připodobnili k nějaké podobné nádobě, u které známe její objem a podle úrovně vody potom odpovíme. V obou případech provádíme relativní odhad množství vody, srovnáváme s onou lahví.

Agilní odhadovací techniky využívají této schopnosti srovnávat. Takže když odhadujeme pracnost, tak si nejdříve stanovíme základ pro to, co porovnáváme v podobě jiného kusu

práce. Tento základ ohodnotíme libovolným odhadem v nějakých nestandardních jednotkách, například v bodech. Potom už stanovujeme pracnost ostatních částí práce vůči stanovenému základu. Je přitom důležité srovnávat stejné typy částí práce. Míněno tak, že srovnávaný kus i základ musí oba být prací, která ještě nebyla hotova. Jestliže něco vytvoříme, máme o tom daleko více znalostí a pakliže to budeme srovnávat s něčím, co jsme ještě nevytvořili, budeme nechtěně usuzovat, že tyto dvě věci jsou si více podobné, než doopravdy jsou. Nakonec jednou z vedlejších výhod využívání relativního odhadování je, že je velmi obtížné využívat naše stanovené základy pro měření výkonnosti nebo sledování odchylek od plánu, přičemž oba tyto ukazatele jsou nevýznamné ve správně nastaveném agilním prostředí.

3.3 Princip rychlosti

Není nutné, aby odhady byly přesné. Důležité je, že jsou provedeny rychle. Odhady nijak nepřispívají ke konečným vlastnostem dodávaného produktu. Existuje spousta kritiků odsuzujících odhadování v agilním prostředí. Sice na úrovni projektu a výš je dobré, aby alespoň vedení mělo přehled o přibližné výši nákladů a době trvání. Avšak agilní manifest říká jasně, že primárním měřítkem postupu projektu je fungující software, takže když už odhadovat, tak rychle a efektivně.

4 Metody odhadování velikosti softwaru

Na základě našeho průzkumu jsou zde popsány nejpoužívanější metody spolu s jejich výhodami a nevýhodami (Reifer, 2017).

4.1 Metoda odhadování analogií

Metoda odhadování analogií spočívá ve srovnání aktuálního projektu s minulými projekty. K aplikaci metody odhadování analogií potřebujeme několik reprezentativních softwarových balíčků, ke kterým disponujeme informacemi ohledně jejich předchozí realizace a velikosti. Při vypracovávání odhadu analogií pak srovnáváme charakteristiky a rozdíly aktuálního projektu se softwarovými balíčky v naší databázi. Jako příklad můžeme uvést vývoj operačního systému. Jelikož známe informace k již dříve vytvořeným softwarovým balíčkům, jako je například scheduler, memory manager, command manager a další, můžeme na základě těchto částí odhadnout velikost aktuálního projektu. K vypočtené velikosti pak můžeme dále přidat několika procentní přirážku, která bude zahrnovat další rizika spojená se složitostí a rozsahem reálného vývoje.

Hlavní výhody

- Jednoduché, pokud máme databázi, kterou lze použít pro srovnání
- Může být použito pro aplikace jakékoli velikosti

Hlavní nevýhody

- Velmi náchylné na kvalitu databáze. Pokud nemáme v databázi přesná a reálná data, lehce dospějeme k velmi zkreslenému a špatnému odhadu.

4.2 Metoda user stories/story points

Než se dostaneme k popisu story points, je třeba představit úvodní terminologii:

- **Story** – Popis nějaké funkcionality nebo rozšíření, které je třeba vyvinout, specifikované uživatelem.
- **Theme** – Kolekce souvisejících stories
- **Epic** – Velká user story, která může být dekomponována na několik menších.

Story points se používají jako relativní měřítko úsilí, které je třeba k implementaci dané funkcionality. Je ovlivňováno množstvím práce, složitostí, rizikem a nejistotou. Nezáleží při tom na konkrétních hodnotách, ale pouze na vztahu hodnot mezi sebou. Mějme tedy 3 funkcionality a přiřaďme jim hodnoty 3, 6 a 1. Tyto hodnoty nám pak říkají, že první funkcionalita vyžaduje třikrát více úsilí než třetí a zároveň druhá je dvakrát náročnější než první. Pokud hodnoty zvolíme jako 4.5, 9 a 1.5, budou nést stejnou informaci.

Hlavní výhody

- Jednoduché k použití
- Umožňuje nám vytvořit relativní velikost aktuálního projektu. Na základě ní pak můžeme odhadovat rychlost implementace, zbývající čas do dokončení a další agilní metriky.

Hlavní nevýhody

- Metoda byla vytvořena zejména pro využití ve Scrumu
- Metoda dává vývojovým týmům absolutní volnost ve vytvoření měřítka. Jako důsledek je však bohužel fakt, že měřítko vytvořené jedním týmem skoro nikdy nepoužívají další týmy pracující na stejném projektu.
- Použití metody je vždy unikátní a metoda postrádá standardizaci

4.3 Metoda funkčních bodů

Metoda funkčních bodů poskytuje měřítko nejmenších jednotek aktivit, kterým dokáže porozumět uživatel nebo zákazník. Funkční body definují velikost ve smyslu dobře definovaných charakteristik softwarové dodávky. Odhady velikosti jsou založeny na počtu parametrů, jako jsou externí vstupy, externí výstupy, externí dotazy (query), externí a dalších relevantních parametrech zobrazených na obrázku Obrázek 1. Samotné funkční body se pak počítají podle standardizovaných výpočetních konvencí a vzorců. Je důležité si uvědomit rozdíl iterativního vývoje softwaru od tradičního a to, že například nemůžeme automaticky přiřazovat jeden funkční bod jedné user story. Zřídka se stává, že by toto odpovídalo a jak se na požadavku pracuje, přičemž se může měnit, tak je většinou jeden „funkční požadavek“ roztažen přes několik user stories. Nemusí být tedy jednoduché rozhodnout, kdy vlastně je funkcionalita doručena, hlavně vezmeme-li v úvahu definice IFPUG (Dekkers, nedatováno).

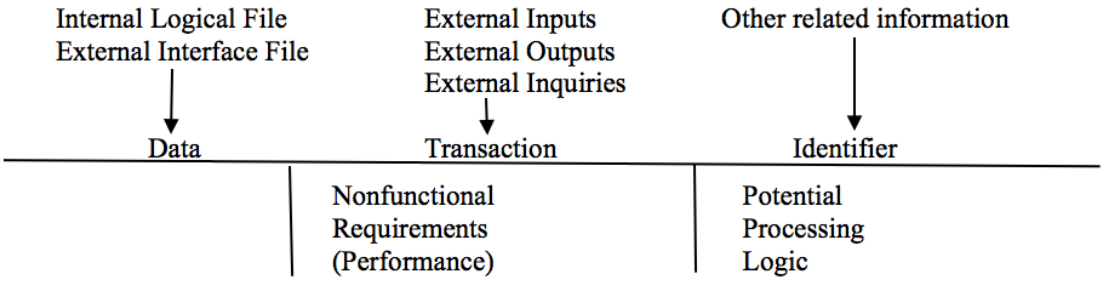

Figure 3: Function Point Counting Parameters

4

Obrázek 1 - Parametry výpočtu funkčních bodů

Hlavní výhody

- Dobře definovaný, vyspělý a standardizovaný přístup odhadování
- Nezávislost na použité metodologii vývoje nebo velikosti projektu
- Může být založena na user stories nebo dalších agilních požadavkových dokumentech
- Dostupnost průmyslových dat o produktivitě, nákladech a kvalitě

- Dostupnost výpočetních konvencí a existují certifikační požadavky pro výpočetní vzorce

Hlavní nevýhody

- Praktiky odhadování velikosti user stories touto metodou jsou relativně nové
- Existuje několik standardů; znamená nutnost identifikovat, který se používá
- Výpočty mohou výt časově náročné a je potřeba školení

4.4 Metoda Proxies

Tato metoda je formou odhadování velikosti analogií. Je založena na tom, že vyvinuté úsilí vývojáře tvořící komponentu podobnou té, kterou už dříve vytvořil, bude přibližně stejné jako u té v minulosti. K odhadování velikosti se mohou používat různé druhy balíčků nazvaných proxies. Jednotliví vývojáři si vytvářejí databázi s historií pracnosti a velikosti projektů, podle kterých jsou rozděleny použité komponenty. Výhodou je, že proxies mohou být vyvinuty pro budoucích aplikací nebo z již vyvinutých, pro které má firma historická data. Také mohou být využity k řešení problému požadavků, ke kterým není mnoho údajů.

Hlavní výhody

- Může být použito pro existující nebo budoucí aplikace, ke kterým nemáme mnoho informací
- Lze spojit s odhadováním nákladů softwaru a dalšími modely k predikci nákladů a kvality

Hlavní nevýhody

- Metoda je tak dobrá jako vytvořené balíčky

- Nefunkční, když vytvořené zástupné balíčky nejsou reprezentativními prvky vyvíjené aplikace

4.5 Metoda Halstead Vocabulary

Podle Halsteada lze na počítačový program pohlížet jako na sekvenci operátorů a jejich operandů. Délka programu, Halsteadova slovní zásoba a objem programu jsou stanoveny na základě Halsteadových metrik. Přesné výpočty jsou uvedeny zde (RistanCASE GmbH, nedatováno). Halsteadovo měřítko komplexity ukazuje, jak je složité program číst nebo psát. Jakmile jsou všechny metriky spočítány, můžeme využít velikost slovní zásoby k odhadu pracnosti a času na vývoj programu.

Hlavní výhody

- Jednoduše použitelná metoda, která může být automatizována i textovým editorem
- Nezávislost na použité metodologii vývoje nebo velikosti projektu
- Může být založena na user stories nebo dalších agilních požadavkových dokumentech

Hlavní nevýhody

- Relativně nová a neprokázaná
- Obtížné pochopení, protože je odlišná od ostatních metod používaných v softwarových kruzích

5 Závěr

V práci jsme popsali 5 nejčastějších metod odhadování velikosti v rámci agilního vývoje. Neexistuje všeobecné přesvědčení o tom, která z popsaných metod je nejlepší, záleží zejména na tom, zda jsou popsané metody správně používány. Všechny popsané metody mají své výhody i nevýhody a vhodnost jejich použití se často liší v závislosti na typu projektu či organizační úrovni. Všichni účastníci průzkumu shodně poznamenali, že dokáží metodu, kterou využívají, používat efektivně, a metoda je pro ně užitečná. Nejpoužívanější techniky v provedeném průzkumu jsou odhadování analogií a funkční body. Halstead vocabulary je velmi nová, a zároveň i slibná metoda.

Některé ze zmíněných metod odhadování jsou závislé na historických datech a databázích, což by mohl být pro některé vývojářské týmy zdánlivý problém. Existuje však mnoho volně dostupných materiálů, které lze efektivně pro odhadování projektu využít.

Jako nevýhodu či nedostatek článku vidíme nedostatečný výklad rozdílů mezi metodou odhadování analogií a metodou Proxies. Přestože se autor popisu jednotlivých metod věnoval, z článku nejsou dostatečně zřetelné rozdíly mezi nimi.

Literatura

- Berteig, Mishkin. 2014. The three fundamental principles of agile estimation – the third one will surprise you! agileadvice. [Online] 10. 6 2014. [Citace: 1. 12. 2017.] http://www.agileadvice.com/2014/06/10/agilemanagement/the-three-fundamental-principles-of-agile-estimation-the-third-one-will-surprise-you/.
- Dekkers, Carol. nedatováno. Counting Function Points for Agile / Iterative Software Development. IFPUG. [Online] nedatováno. [Citace: 25. 11. 2017.] http://www.ifpug.org/Articles/Dekkers-CountingAgileProjects.pdf.
- Fowler et al. 2013. How do you estimate on an Agile project? [Online] 2013. [Citace: 3. 12. 2017.] https://info.thoughtworks.com/rs/thoughtworks2/images/twebook-perspectives-estimation_1.pdf.
- Reifer, Donald. 2017. Size Estimation Approaches for Use with Agile Methods. InfoQ. [Online] 6. 2 2017. [Citace: 2. 12. 2017.] https://www.infoq.com/articles/size-estimation-agile.
- RistanCASE GmbH. nedatováno. Software Metrics Classification. [Online] nedatováno. [Citace: 1. 12. 2017.] https://www.ristancase.com/html/dac/manual/2.12.01-Software-Metrics-Classification.html.