Scrum and XP zo zákopov

Ako robíme Scrum

Autor:

Henrik Kniberg

Preklad:

Text: Dušan Kocúrek

Obrázky: Tibor Radačovský

Revízia: 2

C4Media, Publisher of InfoQ.com.

Táto kniha je súčasťou série kníh InfoQ Enterprise Software Development.

Pre informácie alebo objednávky tejto alebo iných InfoQ kníh prosím kontaktujte books@c4media.com.

Zodpovedný editor: Diana Plesa

Katalogizačné údaje Kongresovej knižnice:

ISBN: 978-1-4303-2264-1

Vytlačené v Spojených štátoch amerických.

Poďakovanie

Prvá verzia tejto knihy mi zabrala pri písaní iba jeden víkend, ale bol to skutočne intenzívne prežitý víkend! 150% faktor pozornosti ☺

Ďakujem mojej manželke Sophii a deťom Davovi a Jennymu za trpezlivosť počas tohto víkendu a Sophinym rodičom Eve a Jörgen za pomoc pri starostlivosti o rodinu.

Ďakujem aj všetkým mojim kolegom v spoločnosti Crisp v Štokholme a ľuďom z Yahoo skupiny scrumdevelopment za komentáre a pomoc pri zlepšení tejto publikácie.

A nakoniec sa chcem poďakovať všetkým čitateľom, ktorí mi poskytovali neustálu spätnú väzbu. Obzvlášť rád počujem, že táto kniha zaiskrila medzi Vami a naštartovala agilný softvérový vývoj.

Obsah

Predslov: Jeff Sutherland

Tímy potrebujú spoznať základy Scrumu. Ako vytvoriť a odhadnúť produktový backlog? Ako ho premeniť na backlog sprintu? Ako spravovať burndown graf a vyrátať rýchlosť vášho tímu? Henrikova kniha je starter kit základných techník, ktoré pomáhajú tímom posunúť sa od skúšky Scrumu až po správne použitie Scrumu.

Správne použitie Scrumu sa stáva dôležitejším pre tímy, ktoré chcú investovať .Ako Agile tréner v investičnej skupine som pomáhal s cieľom investovať iba v agilných spoločnostiach, ktoré praktikujú Agilné praktiky správne. Senior partner tejto skupiny sa pýta všetkých spoločností z portfólia či poznajú rýchlosť ich tímov. V tomto čase hľadajú odpovede ťažko. Budúce možnosti investícií budú požadovať aby vývojové tímy chápali ich rýchlosť produkcie softvéru.

Prečo je to dôležité? Ak tím nepozná rýchlosť, produktový vlastník nemôže vytvárať produktový plán so spoľahlivými dátumami releasov. Bez spoľahlivých dátumov releasov spoločnosť môže zlyhať a investor môžu stratiť ich peniaze!

S týmto problémom sa stretávajú spoločnosti veľké aj malé, nové aj staré, finančne zabezpečené aj nie. Počas poslednej diskusie o implementácii Scrumu v spoločnosti Google na londýnskej konferencii som sa spýtal 135 poslucháčov koľko z nich už robili Scrum. 30 odpovedalo pozitívne. Potom som sa ich spýtal či robili iteratívny vývoj podľa štandardov Nokie. Iteratívny vývoj je fundamentálny podľa Agile Manifesto - odovzdávajte funkčný softvér skoro a často. Po rokoch retrospektívy so stovkami tímov používajúcich Scrum, Nokia definovala základné požiadavky pre iteratívny vývoj:

- Iterácie musia mať pevnú časovú dĺžku, ktorá musí byť kratšia ako 6 týždňov.
- Kód na konci iterácie musí byť otestovaný QA a musí pracovať správne.

Z 30 ľudí, ktorí odpovedali že robia Scrum, iba polovica implementovala prvý princíp Agile Manifesto podľa Nokia štandardov.

Potom som sa ich spýtal, či spĺňajú Nokia štandardy pre Scrum:

- Scrum tím musí mať definovanú rolu produktový vlastník a musí vedieť kto ním je.
- Produktový vlastník musí mať produktový backlog s odhadmi vytvorenými tímom.
- Tím musí mať Burndown Graf a musí vedieť rýchlosť.
- Počas sprintu nesmie nikto mimo tímu zasahovať do prác.

Z 30 ľudí, ktorý vyvíjali pomocou Scrumu, iba 3 spĺňali test spoločnosti Nokia pre Scrum tímy. Toto sú tímy, ktoré získajú budúce investície od mojich investičných partnerov.

Hodnota Henrikovej knihy je v tom, že ak budete nasledovať praktiky, ktoré načrtol, budete mať produktový backlog, budete mať odhady backlogu, burndown graf. Budete poznať rýchlosť tímu spolu s inými ďalšími základnými praktikami pre vysoko funkčný Scrum. Budete podporovať Nokia test pre Scrum hodný investícií do Vašej práce. Ak ste start-up spoločnosť, možno dokonca získate financovanie od investičnej skupiny. Možno ste budúcnosťou softvérového vývoja a autor nasledujúcej generácie popredných softvérových produktov.

Jeff Sutherland,

Ph.D., spoluautor Scrum

Predslov: Mike Cohn

Scrum aj Extreme Programming (XP) žiadajú od tímov dosiahnuť hmatateľné výsledky práce na konci každej iterácie. Tieto iterácie sú krátke a časovo ohraničené. Zameranie sa na odovzdateľný kód v krátkych časových rámcoch znamenajú, že Scrum a XP tímy nemajú čas na teórie. Nerobia kreslenie perfektného UML modelu v nejakom case nástroji, nepíšu perfektný dokument s požiadavkami, ani nepíšu kód, ktorý by vyhovoval všetkým budúcim predstaviteľným zmenám. Namiesto toho, Scrum a XP tímy sú zamerané na dokončenie veci. Tieto tímy akceptujú, že sa mohli zmýliť na ich ceste. Zároveň ale chápu, že najlepším spôsobom ako nájsť chyby je prestať rozmýšľať o softvéri na teoretickej úrovni analýzy a dizajnu a naopak ponoriť sa, zašpiniť si ruky a začať budovať produkt.

Je to práve toto centrum záujmu ako robiť veci radšej ako teoretizovať, ktoré odlišuje túto knihu. Spôsob akým Henrik Kniberg rozmýšľa je zrejmý od samého začiatku. Neponúka dlhé popisy čo Scrum je, namiesto toho nás odporúča na jednoduché webové stránky. Naopak, Henrik ide priamo na vec a začína popisom ako jeho tím spravuje a pracuje s ich produktovým backlogom. Od tohto bodu sa presúva cez všetky ďalšie elementy a praktiky správne fungujúceho agilného projektu. Neteoretizuje. Neodkazuje. Nepoužíva poznámky pod čiarou. Nie sú potrebné. Henrikova kniha nie je filozofickým vysvetlením prečo Scrum funguje alebo prečo máte skúsiť to alebo ono. Je to popis ako správne pracujúci agilný tím pracuje.

Toto je dôvod prečo podtitul "Ako robíme Scrum" je tak výstižný. Možno to nie je spôsob ako robíte Scrum *Vy*. Kniha je o tom ako Scrum robí Henrikov tím. Možno sa pýtate, prečo by vás malo zaujímať ako iný tím robí Scrum. Malo by vás to zaujímať, pretože počúvaním príbehov sa všetci môžeme učiť ako robiť Scrum lepšie, ako to bolo urobené inými a hlavne tými, ktorí ho robia správne. Neexistuje a ani nebudú vydané "Najlepšie praktiky Scrum" pretože tímy a súvislosti projektu pretromfnú všetky iné zretele. Namiesto najlepších praktík potrebujeme

v skutočnosti poznať najlepšie praktiky a súvislosti, v ktorých boli úspešné. Prečítajte dostatočný počet príbehov úspešných tímov a ako robili veci. Takto budete pripravení na prekážky pre použití Scrumu a XP.

Henrik poskytuje dobré praktiky spolu so súvislosťami, ktoré nás učia ako lepšie robiť Scrum a XP v zákopoch našich vlastných projektov.

Mike Cohn

Autor knihy *Agile Estimating and Planning* a *User Stories Applied for Agile Software Development.*

Úvod – Hej, Scrum funguje!

Scrum funguje! Prinajmenšom pre nás (myslím tým môjho aktuálneho klienta v Štokholme, ktorého meno tu nespomeniem). Dúfam, že bude fungovať aj pre Vás. Možno Vám táto publikácia pomôže na Vašej ceste.

Toto je prvýkrát čo som videl vývojovú metodológiu (prepáč Ken, framework) fungovať hneď podľa knihy. Plug and play. Všetci sme z toho šťastný – vývojári, testeri aj manažéri. Pomohla nám dostať sa z ťažkých situácií a dovolila nám zachovať si zameranie a pohotovosť napriek množstvu turbulencií obchodu a redukcii zamestnancov.

Nemal by som hovoriť, že som bol prekvapený, no bol som. Po počiatočnom strávení niekoľko kníh na tému Scrum to vyzeralo dobre, ale až veľmi dobre nato, aby to bola pravda. Takže som bol trochu skeptický. Avšak potom, čo robím Scrum už rok, som dostatočne zaujatý (a takisto je aj väčšina ľudí v mojich tímoch) nato, aby som pokračoval v používaní Scrumu v nových projektoch kedykoľvek, keď nebude vážny dôvod proti.

1

Úvod

Vo vašej organizácii začínate používať Scrum. Alebo ste ho možno používali niekoľko mesiacov. Získali ste základy, prečítali ste knihy, možno ste dokonca získali certifikát ako Scrum Master. Gratulujem!

Ale cítite sa zmätene.

Slovami Kena Schwabera, Scrum nie je metodológia, je to *systém*.

To znamená, že Scrum Vám nepovie čo presne treba robiť. Do čerta.

Dobrou správou je, že vám presne poviem ako robím Scrum ja. Poviem vám to v boľavých až mučivých detailoch. Nuž, zlou správou je, že vám poviem *iba* spôsob ako robím Scrum *ja*. Neznamená to, že aj *vy* ho máte robiť presne týmto spôsobom. V skutočnosti aj ja ho môžem robiť inak ak narazím na rôzne situácie.

Silou a slabinou Scrumu je to, že ste povinný ho prispôsobiť vašej špecifickej situácii.

Môj aktuálny prístup k Scrumu je výsledkom ročného experimentovania so Scrumom vo vývojovom tíme približne 40 ľudí. Firma bola v ťažkej situácii s vysokým množstvom nadčasov, vážnymi problémami s kvalitou, konštantným hasením problémov, nestihnutými termínmi atď. Spoločnosť sa rozhodla použiť Scrum, ale v skutočnosti neukončila implementáciu. Toto bola moja úloha. Pre väčšinu ľudí vo vývojovom tíme v tom čase bol "Scrum" iba stále sa omieľajúce slovo bez dôsledkov v ich dennej práci.

Počas ročnej implementácie Scrumu cez všetky vrstvy spoločnosti sme skúšali rôzne veľkosti tímov (3-12 ľudí), rôzne dĺžky sprintov (2-6 týždňov), rôzne spôsoby definície "hotovo", formáty produktového a sprint backlogu (Excel, Jira, indexové kartičky), rôzne stratégie testovania, predvádzania dema, synchronizácie viacerých Scrum tímov atď. Experimentovali sme zároveň aj s XP praktikami – rôznymi spôsobmi vykonávania kontinuálnych buildov, párového programovania, vývoja riadeného testovaním a ako ich skombinovať so Scrumom.

Je to konštantný proces učenia, takže príbeh sa tu nekončí. Som presvedčený, že táto spoločnosť sa bude naďalej učiť (pokiaľ vytrvajú v retrospektívach sprintu) a získa nové poznatky ako najlepšie implementovať Scrum v konkrétnom kontexte.

Upozornenie

Tento dokument neuvádza "správny" spôsob robenia Scrumu! Prezentuje iba jeden spôsob Scrumu ako výsledok konštantného zlepšovania počas jedného roka. Možno sa rozhodnete, že sme všetko robili zle.

Všetko v tomto dokumente reflektuje moje osobné, subjektívne, názory, ktoré nie sú v žiadnom zmysle oficiálnym stanoviskom spoločnosti Crisp alebo môjho aktuálneho klienta. Z tohto dôvodu nebudem uvádzať špecifický produkt alebo ľudí.

Prečo som to napísal

Keď som sa učil o Scrume, prečítal som relevantné knihy o Scrume a agilných postupoch, brodil som sa stránkami a fórami o Scrume, získal som certifikáciu od Kena Schwabera, zahrnul som ho otázkami a strávil množstvo času diskusiami s mojimi kolegami. Najcennejšími zdrojmi informácií boli aktuálne príbehy. Príbehy zmenili Princípy a Praktiky na … nuž …. Ako To Totiž Urobíte. Pomohli mi identifikovať (a niekedy sa vyhnúť) typickým začiatočníckym chybám.

Takže toto je moja šanca vrátiť niečo naspäť. Toto je môj bojový príbeh.

Dúfam, že táto publikácia poskytne použiteľnú spätnú odozvu tým, ktorí sú v rovnakej situácii. Informujte ma prosím!

Ale čo je Scrum?

Ó, prepáčte. Ste úplným nováčikom v Scrume a XP? V tomto prípade sa pozrite na nasledujúce odkazy:

- http://agilemanifesto.org/
- http://www.mountaingoatsoftware.com/scrum
- http://www.xprogramming.com/xpmag/whatisxp.htm

Ak ste netrpezlivý, čítajte. Potrebný slovník je vysvetlený priebežne.

2

Ako pripravujeme backlog

Produktový backlog je srdcom Scrumu. Tu sa všetko začína. Backlog je v podstate prioritizovaný zoznam požiadaviek, stories, vlastností alebo čohokoľvek. Vecí, ktoré zákazník chce, popísaných terminológiou zákazníka.

Voláme ich *stories* alebo niekedy iba jednoducho *položky backlogu*.

Naše stories obsahujú nasledovné položky:

- **ID** – jedinečná identifikácia, jednoduché automaticky zvyšované číslo. Umožňuje nám nestratiť sa keď premenujeme story.
- **Názov –** krátky, popisný názov story. Napr. "Vidieť moju históriu transakcií". Jasné dosť nato, aby vývojári a produktový vlastník približne chápali o čom sa rozprávajú a jasné nato, aby jednotlivé story boli navzájom rozlíšiteľné. Normálna dĺžka je 2 – 10 slov.
- **Dôležitosť** – dôležitosť story podľa produktového vlastníka. Napr. 10 alebo 150. Čím väčšie číslo, tým dôležitejšia story.
    - Snažím sa vyhnúť pojmu priorita, pretože priorita 1 je typicky považovaná za najvyššiu prioritu. Vyzerá to zle keď sa rozhodnete že niečo iné je dôležitejšie. Aké číslo priority má táto story dostať? 0? Alebo -1?
- **Počiatočný odhad** – počiatočný odhad prácnosti implementácie úlohy v porovnaní s inými stories

urobenými tímom. Jednotkou sú story pointy, ktoré zvyčajne približne korešpondujú s "ideálnymi človekodňami"..

- Spýtajte sa tímu "ak na implementáciu tejto story máte optimálny počet ľudí (nie málo ani veľa, zvyčajne 2) a zamknete sa v miestnosti s dostatkom jedla a pracujete úplne neprerušovane, po koľkých dňoch vyjdete von z miestnosti s ukončenou, demonštrovateľnou, otestovanou a dodateľnou implementáciou?". Ak odpoveďou je "s 3 vývojármi zamknutými v miestnosti to bude trvať približne 4 dni" potom počiatočný odhad je 12 story pointov.
- Dôležitým nie je získať absolútne presné a správne čísla (teda že 2 story pointy majú trvať 2 dni), ale relatívne, správne odhady (teda že 2 bodová story by mala trvať polovicu práce 4 bodovej story.).

- **Ako robiť demo** – hrubý popis ako bude táto story demonštrovaná počas dema sprintu. Je to v podstate jednoduchá špecifikácia testu "Urob to, potom toto a toto musí nastať".
  - Ak praktikujete TDD (test-driven development) potom tento popis môže byť použitý aj ako pseudo kód pre vaše akceptačné testy.
- **Poznámky** – akékoľvek ďalšie informácie, vysvetlenia, referencie na iné zdroje atď. Často veľmi stručné.

PRODUKTOVÝ BACKLOG (príklad)					
ID	**Názov**	**Dôležitosť**	**Odhad**	**Ako demonštrovať**	**Poznámky**
1	Depozit	30	5	Prihlásiť sa, otvoriť stránku depozitu, depozit €10, ísť na stránku My balance	Potrebný UML sekvenčný diagram.

				a skontrolovať či bol stav zvýšený o €10.	Nateraz nie je potrebné sa zaoberať kódovaním.
2	Zobraziť históriu vlastných transakcií	10	8	Prihlásiť sa, kliknúť na “Transakcie”, urobiť deposit. Ísť naspäť na transakcie, skontrolovať zobrazenie nového depozitu.	Použiť stránkovanie pre obmedzenie veľkých DB dotazov. Dizajn podobný stránke pre zobrazenie užívateľov.

Experimentovali sme s množstvom iných položiek, ale nakoniec sa ukázalo, že sme použili iba 6 z vyššie uvedených položiek.

Backlog obyčajne píšeme v programe Excel so zapnutým zdieľaním, takže viacerí užívatelia ho môžu simultánne editovať. Produktový vlastník vlastní oficiálne tento dokument. Nechceme ho ale blokovať pred ostatnými užívateľmi. Stáva sa často, že vývojár potrebuje otvoriť dokument pre vyjasnenie alebo pre zmenu odhadu.

Z tohto dôvodu tento dokument nespravujeme pomocou systému správy verzií. Namiesto toho je dokument umiestnený na zdieľanom mieste. Toto je najjednoduchší spôsob ako dovoliť viacerým simultánnym editorom pracovať s dokumentom bez blokovania alebo konfliktov.

Väčšina iných artefaktov je však systémom správy verzií spravovaná.

Dodatočné vlastnosti story

Niekedy, hlavne kvôli produktovému vlastníkovi, používame ako pomoc s určením priority v produktovom backlogu aj dodatočné atribúty.

- **Kategória** – hrubá kategorizácia tejto story, napr. "back office" alebo "optimalizácia". Takto môže produktový vlastník jednoducho filtrovať všetky story "optimalizácia" a nastaviť im nízku prioritu, atď.
- **Komponenty** – obyčajne realizované ako zaškrtávacie tlačidlá v dokumente programu Excel, napr. "databáza, server, klient". Tím a produktový vlastník môžu identifikovať, ktoré technické komponenty môžu byť ovplyvnené implementáciou story. Použiteľné ak máte viacero Scrum tímov. Napr. chcete jednoduchšie ozrejmiť či story je pre tím Back Office alebo tím Klient
- **Žiadateľ** – produktový vlastník môže chcieť evidovať, ktorý zákazník alebo reprezentant zákazníka pôvodne požadoval vlastnosť, aby ho mohol oboznamovať s priebehom vývoja.
- **ID** chyby – ak máte oddelený bug tracking systém, napr. my používame Jira, je často užitočné evidovať vzťah medzi story a jednou alebo viacerými reportovanými chybami.

Ako udržiavať produktový backlog na obchodnej úrovni

Ak produktový vlastník má technické pozadie, môže pridávať stories ako "Pridať indexy do tabuľky Udalosti". Prečo by to mal chcieť? Cieľ v pozadí môže pravdepodobne znieť "zrýchliť hľadanie vo formulári Udalosti v back office".

Môže sa ukázať, že indexy neboli úzkym miestom spôsobujúcim, že formulár bol pomalý. Môže to byť niečo úplne iné. Tím vie často vysloviť *ako* niečo vyriešiť, a preto produktový vlastník by sa mal zamerať na obchodné ciele.

Ak vidím story zameranú technicky, podobnú tejto, obyčajne sa spýtam produktového vlastníka sériu otázok „Ale *prečo*?“, až kým nenájdeme skutočný cieľ. Potom prepíšeme story v zmysle identifikovaného skutočného cieľa („zrýchliť hľadanie vo formulári Udalosti v back office“). Pôvodné technické zadanie potom bude uvedené ako poznámka („Indexácia tabuľky Udalosti to môže vyriešiť“).

3

Ako sa pripravujeme na plánovanie sprintu

Fajn, deň plánovania sprintu sa teda rýchlo blíži. Čo sa zakaždým učíme je:

Lekcia: Pred mítingom plánovania sprintu sa uistite, že produktový backlog je v poriadku.

A čo *to* znamená? Že boli všetky stories perfektne a správne definované? Že všetky odhady sú správne? Že všetky priority sú pevne stanovené? Nie, nie a nie! Všetko čo to znamená je:

- Produktový backlog musí existovať! (Viete si to predstaviť?)
- Musí existovať práve jeden produktový backlog a jeden produktový vlastník (pre produkt).
- Pre všetky dôležité položky by mali mať priradené *rôzne* dôležitosti.
    - V skutočnosti je dobre ak menej dôležité položky majú rovnakú hodnotu, pretože pravdepodobne sa nevytiahnu počas mítingu plánovania sprintu.
    - Akákoľvek story, o ktorej je produktový vlastník presvedčený, že je len pravdepodobné, že bude súčasťou nasledujúceho sprintu, musí mať jedinečnú hodnotu dôležitosti.
    - Hodnotenie dôležitosti je použité iba pre triedenie položiek podľa dôležitosti.

Takže ak Položka A má dôležitosť 20 a Položka B má dôležitosť 100, potom B je jednoducho dôležitejšie ako A. *Neznamená* to, že B je 5 krát dôležitejšia než A. Ak by B malo hodnotu dôležitosti 21, stále to bude rovnaká situácia.

- Je užitočné nechať si priestor v poradí čísiel pre prípad, že Položka C sa stane dôležitejšou než A, ale menej dôležitou ako B. Samozrejme, že pre C môžete použiť číslo 20.5, ale nevyzerá to dobre, takže radšej si ponechajte medzery!

- Produktový vlastník by mal *rozumieť* každej story (normálne je jej autorom, ale v niektorých prípadoch pridávajú požiadavky aj iní ľudia, ktoré produktový vlastník môže prioritizovať). Nepotrebuje vedieť presne čo má byť implementované, ale mal by rozumieť prečo daná story existuje.

Poznámka: Aj iní ľudia ako produktový vlastník môžu pridávať stories do produktového backlogu. Nesmú ale priradzovať prioritu, toto je výhradné právo produktového vlastníka. Nesmú nastaviť ani odhad času, toto je zase výhradné právo tímu.

Iné postupy, ktoré sme vyskúšali:

- Použitie Jira (náš systém evidencie chýb) pre udržiavanie produktového backlogu. Väčšina našich produktových vlastníkov mala výhrady k veľkej potrebe klikania. Excel je pekný a jednoduchý pre priamu manipuláciu. Môžete ľahko použiť farby, presúvať položky, pridávať nové stĺpce podľa potreby, pridávať poznámky, exportovať a importovať údaje atď.

- Použitie podporných nástrojov pre agilné procesy ako napr. VersionOne, ScrumWorks, Xplanner, , atď. Zatiaľ sme neskúšali žiaden, ale pravdepodobne budeme.

4

Ako plánujeme sprint

Plánovanie sprintu je kritickým mítingom, pravdepodobne najdôležitejšou udalosťou v Scrume (podľa môjho osobného názoru). Zle urobený míting plánovania sprintu môže pokaziť celý sprint.

Účelom mítingu plánovania sprintu je poskytnúť tímu dostatok informácií aby mohli pracovať v nerušenom pokoji počas niekoľkých týždňov a poskytnúť produktovému vlastníkovi odvahu ich nechať takto pracovať.

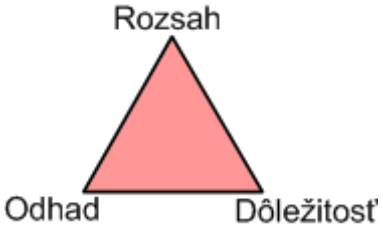
OK, toto bolo nejasné. Konkrétnym výstupom mítingu plánovania sprintu je:

- Cieľ sprintu.
- Zoznam členov tímu (a ich úroveň povinnosti, ak to nie je 100%).
- Backlog sprintu (= zoznam stories zahrnutých do sprintu).
- Definovaný dátum dema sprintu.
- Definovaný dátum a miesto pre denný scrum.

Prečo sa produktový vlastník má zúčastniť

Niekedy je produktový vlastník neochotný tráviť hodiny s tímom, ktorý plánuje sprint. „Už som uviedol čo som chcel. Nemám čas sa zúčastniť plánovania sprintu. Toto je vážny problém.

Dôvodom prečo celý tím *a* produktový vlastník sa majú zúčastniť mítingu plánovania sprintu je, že každá story zahŕňa tri premenné navzájom vysoko závislé.



Rozsah a dôležitosť sú dané produktovým vlastníkom. Odhad je daný tímom. Počas mítingu plánovania sprintu sú tieto premenné jemne vyladené počas rozhovorov tvárou v tvár medzi tímom a produktový vlastníkom.

Míting obyčajne začína produktový vlastník sumarizáciou cieľa sprintu a najdôležitejších stories. Potom tím časovo odhaduje každú story, pričom začne s najdôležitejšou. Počas toho ako to robia, sa spýtajú otázku o rozsahu – „'zmazať užívateľa' zahŕňa aj zrušenie všetkých pretrvávajúcich transakcií tohto užívateľa?". V niektorých prípadoch bude pre členov tímu odpoveď prekvapujúca, často ich prinúti k prehodnoteniu ich odhadu.

V niektorých prípadoch nebude časový odhad spĺňať očakávania produktového vlastníka. Táto situácia ho môže prinútiť zmeniť dôležitosť story, alebo zmeniť rozsah, čo znova spôsobí odhadovanie času atď, atď,...

Takýto typ priamej spolupráce je základom pre Scrum a v skutočnosti pre celý agilný softvérový vývoj.

Čo ak produktový vlastník stále trvá na tom, že nemá čas sa zúčastniť mítingu plánovania sprintu? Zvyčajne používam nasledovné stratégie v uvedenom poradí:

- Skúste produktovému vlastníkovi pomôcť pochopiť prečo je jeho priama účasť kľúčová a dúfajte, že zmení svoje rozhodnutie.

- Skúste nájsť v tíme dobrovoľníka, ktorý zastúpi produktového vlastníka počas mítingu. Povedzte produktovému vlastníkovi „Pretože sa nemôžeš zúčastniť nášho mítingu, necháme Jeffa aby ťa zastupoval ako tvoj zástupca. Bude plne oprávnený zmeniť priority a rozsah stories. Odporúčam ti dohodnúť sa s ním pred mítingom. Ak sa ti Jeff nepáči ako tvoj zástupca, navrhni prosím ťa niekoho iného tak, aby bol schopný sa zúčastniť celého mítingu plánovania sprintu".
- Skúste presvedčiť manažment o potrebe určenia nového produktového vlastníka.
- Odložte spustenie sprintu pokiaľ si produktový vlastník nenájde čas zúčastniť sa mítingu. Medzitým odmietajte potvrdenia akýchkoľvek dodávok. Nechajte tím tráviť dni robením toho čo považujú za najdôležitejšie v daný deň.

Prečo sa kvalita nedohaduje

Vo vyššie uvedenom trojuholníku som sa zámerne snažil vyhnúť štvrtej premennej *kvalite*.

Pokúšam sa rozlišovať medzi *internou kvalitou* a *externou kvalitou*.

- *Externá kvalita* je to čo si užívateľ systému všime. Pomalý a neintuitívne užívateľské rozhranie je príkladom zlej externej kvality.
- *Interná kvalita* odkazuje na prípady, ktoré zvyčajne nie sú viditeľné užívateľovi, ale majú hlboký efekt na udržiavanie systému. Veci ako konzistencia systémového dizajnu, pokrytie testami, čitateľnosť kódu, refaktoring atď.

Povedané vo všeobecnosti, systém s vysokou internou kvalitou môže stále mať nízku externú kvalitu. Ale, systém s nízkou internou kvalitou len zriedka má vysokú externú kvalitu. Je veľmi ťažké vybudovať niečo pekné na zhnitých základoch.

Externú kvalitu považujem za súčasť rozsahu. V niektorých prípadoch má perfektný obchodný zmysel uvoľniť verziu systému, ktorá má neohrabané a pomalé užívateľské rozhranie a až neskôr zverejniť upravenú verziu. Nechávam to na produktovom vlastníkovi, pretože je zodpovedný za stanovenie rozsahu.

Interná kvalita ale nie je o diskusii. Je zodpovednosťou tímu udržiavať kvalitu systému za každej situácie. Neexistuje tu žiadna dohoda. Nikdy.

(Nuž, OK, *skoro* nikdy)

Takže ako objasníme rozdiel medzi problémami internej kvality a externej kvality?

Nech produktový vlastník povie: „Ok, rešpektujem váš odhad času pre 6 story pointov, ale som si istý, že môžete urobiť nejakú rýchlu zmenu tohto času na polovicu, ak sa trochu zamyslíte.“

Aha! Skúša použiť internú kvalitu ako premennú. Ako to viem? Pretože chce, aby sme redukovali odhad času bez „zaplatenia ceny“ za redukciu rozsahu. Slová „rýchla zmena“ by mali spustiť alarm vo vašej hlave...

A prečo to my nedovoľujeme?

Mojou skúsenosťou je, že obetovanie internej kvality je skoro vždy hrozný, desivý nápad. Ušetrený čas je veľmi prevážený cenou v krátkodobom aj dlhodobom období. Ak už raz bolo dovolené kaziť bázu kódu je len veľmi ťažké dosiahnuť späť kvalitu. Namiesto toho sa pokúšam viesť diskusiu k rozsahu. „Keďže je pre vás dôležité mať túto vlastnosť dostupnú skoro, môžeme redukovať rozsah a dosiahnuť tak rýchlejšiu implementáciu? Možno by sme mohli zjednodušiť spracovanie chýb a urobiť ‘Pokročilé spracovanie chýb’ samostatnou story. Alebo môžeme znížiť prioritu iných stories takže sa môžeme zamerať na túto problematickú?”

Keď sa produktový vlastník naučil, že interná kvalita nie je vecou dohody, obyčajne zistí, že celkom dobre stačí manipulovať ostatnými premennými.

Mítingy plánovania sprintu, ktoré sa len ťahajú a ťahajú ...

Najťažšou vecou na plánovaní sprintov je:

1) Ľudia neočakávajú, že budú trvať tak dlho.

2) ... ale trvajú!

Všetko v Scrume je časovo vyhradené. Milujem toto jediné, jednoduché, konzistentné pravidlo. Pokúšame sa na ňom ľpieť.

Takže čo robíme keď časovo ohraničený míting plánovania sprintu je blízko konca a stále sa nejavia príznaky cieľa sprintu alebo backlogu sprintu? Máme ho jednoducho skrátiť? Alebo ho máme natiahnuť na hodinu? Alebo ho máme prerušiť a pokračovať nasledujúci deň?

Často sa to stáva stále a stále, hlavne pre nové tímy. Takže čo máte urobiť? Neviem. A čo robíme my? Ó, hmmm, obyčajne brutálne skrátim míting. Ukončím ho. Nech sprint trpí. Presnejšie poviem tímu a produktovému vlastníkovi „takže, tento míting sa skončí za 10 minút. V skutočnosti nemáme toho veľa naplánovaného pre sprint. Alebo urobíme to čo máme, alebo si naplánujeme ďalší 4 hodinový míting plánovania sprintu zajtra od 8.00?“ Môžete hádať čo odpovedia... :o)

Pokúšal som sa naťahovať mítingy. Ničomu to nepomohlo, pretože ľudia boli unavení. Ak nevyprodukujú slušný plán sprintu za 2-8 hodiny (alebo akýkoľvek váš časový rámec), nevyprodukujú ho ani za ďalšiu hodinu. Nasledujúca možnosť je celkom OK - naplánovať nový míting nasledujúci deň. Až nato, že ľudia sú obyčajne netrpezliví, chcú pokračovať so sprintom a nie tráviť ďalšie hodiny pri plánovaní.

Takže, skráťte ho. A áno, sprint bude trpieť. Výhodou je, že tím sa naučí veľmi cennú lekciu a ďalší míting plánovania sprintu bude oveľa efektívnejší. Navyše budú ľudia menej vzdorovití keď navrhnete takú dĺžku sprintu, o ktorej by predtým uvažovali ako o veľmi dlhej.

Naučte sa udržiavať váš časový rámec, naučte sa nastaviť reálnu dĺžku časového obdobia. Týka sa to dĺžky sprintu aj mítingov.

Program mítingu plánovania sprintu

Existencia predbežného časového rozvrhu pre míting plánovania sprintu umožní redukovať riziko narušenia časového rámca.

Tu je príklad nášho typického časového rozvrhu.

Míting plánovania sprintu: 13:00 – 17:00 (10 minútová prestávka každú hodinu)

- **13:00 – 13:30.** Produktový vlastník preberie cieľ sprintu a zosumarizuje produktový backlog. Je určený dátum a miesto dema.
- **13:30 – 15:00**. Tím urobí časový odhad a rozdelí položky ak je to potrebné. Produktový vlastník mení prioritu, ak je to potrebné. Položky sú ozrejmované. Pre všetky vysoko prioritné položky je vyplnený údaj 'Ako urobiť demo'.
- **15:00 – 16:00**. Tím si vyberie stories do sprintu. Prepočíta rýchlosť ako kontrolu reality.
- **16:00 – 17:00**. Určí sa čas a miesto pre denné scrum (ak sa líšia od posledného sprintu). Tím nasledovne rozdelí stories na úlohy.

Časový rozvrh nie je striktne dodržiavaný. Scrum master môže predĺžiť alebo skrátiť časové rámce podľa potreby ako sa vyvíja míting.

Definujeme dĺžku sprintu

Jedným z výstupov mítingu plánovania sprintu je definovaný dátum dema sprintu. To znamená, že sa musíte rozhodnúť o dĺžke sprintu.

Takže aká je dobrá dĺžka sprintu?

Nuž, krátke sprinty sú dobré. Dovoľujú spoločnosti byť „agilnou“, teda meniť často smerovanie. Krátke sprinty = krátky cyklus spätnej väzby = častejšie dodávky = častejšia zákaznícka odozva = menej času stráveného zlým smerovaním = rýchlejšie a vylepšené učenie, atď.

Ale aj dlhé sprinty sú dobré. Tím má viac času vybudovať svoju hybnosť, majú viac priestoru pre zotavenie sa z problémov, pričom stále pracujú na cieli sprintu, neunavíte sa termínmi mítingov plánovania sprintov, demami atď.

Vo všeobecnosti povedané, produktoví vlastníci majú radšej krátke sprinty, vývojári dlhé sprinty. Dĺžka sprintu je teda kompromisom. Dosť sme s tým experimentovali a našli sme si našu obľúbenú dĺžku sprintu: 3 týždne. Väčšina našich tímov (ale nie všetky) pracujú v 3 týždňových sprintoch. Sú dosť krátke nato, aby poskytovali firemnú agilitu, dosť dlhé nato, aby tím dosiahol tok a zregenerovanie sa z problémov, ktoré sa objavili počas sprintu.

Na jednej veci sme sa zhodli: *experimentujte* s dĺžkou sprintu na začiatku. Nemrhajte veľa času na *analýzy*, radšej si zvoľte dobrú dĺžku, ktorú skúsite počas jedného dvoch sprintov. Neskôr ju môžete zmeniť, ak to bude potrebné.

Ale, keď ste sa už raz dohodli na dĺžke, ktorá vám najviac vyhovuje, *zabetónujte* ju na dlhšie časové obdobie. Po niekoľkých mesiacoch experimentovania sme zistili, že 3 týždne sú dobré. Takže teraz robíme 3 týždňové sprinty, bodka. Niekedy sa zdajú dosť dlhé, niekedy naopak krátke. Ale udržaním rovnakej dĺžky sprintu sa táto dĺžka stala korporátnym tlkotom srdca, ktorá sadla všetkým. Neexistujú žiadne argumenty o dátumoch releasov a podobne pretože všetci vedia, že release je každé 3 týždne, bodka.

Definovanie cieľa sprintu

Stáva sa to zakaždým. V nejakom bode počas mítingu plánovania sprintu sa spýtam „takže aký je cieľ tohto

sprintu?“ a každý na mňa zíza prázdnym pohľadom a produktový vlastník zvraští čelo a poškriabe si bradu.

Z nejakého dôvodu je *ťažké* nájsť cieľ sprintu. Zistil som ale, že sa skutočne oplatí nejaký nastaviť. Radšej z polovice mizerný než vôbec žiaden. Cieľom môže byť „zarobiť peniaze“ alebo „ukončiť tri najdôležitejšie stories“ alebo „urobiť dojem na CEO“ alebo „urobiť systém dostatočne dobrý na publikovanie medzi beta skupinu“ alebo „pridať základy pre podporu back office“ alebo čokoľvek iné. Dôležitou vecou je, že musí byť v obchodných pojmoch, nie technických. Znamená to v pojmoch, ktorým rozumejú aj ľudia mimo tímu.

Cieľ sprintu musí odpovedať na základnú otázku: „Prečo robíme tento sprint? Prečo by sme radšej nemali ísť všetci na dovolenku?“ V skutočnosti jedným spôsobom ako vymámiť cieľ sprintu od produktového vlastníka je doslova položiť túto otázku.

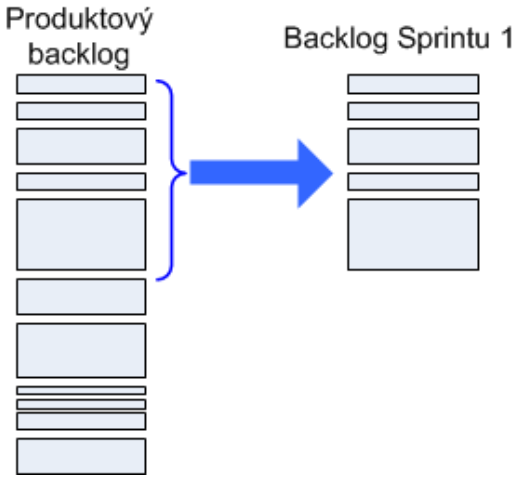
Cieľom musí byť niečo, čo ešte nebolo dosiahnuté. „Urobiť dojem na CEO“ je možno dobrým cieľom, ale nie vtedy, keď už je zaujatý systémom v jeho aktuálnom stave. V tomto prípade všetci môžu ísť domov a cieľ sprintu bude naďalej dosiahnutý.

Cieľ sprintu môže vyzerať dosť smiešne a vyumelkovane počas plánovania sprintu, ale často je použitý uprostred sprintu, keď ľudia začínajú byť zmätení čo majú robiť. Ak máte viac Scrum tímov (tak ako my) pracujúcich na rôznych produktoch, je veľmi užitočné urobiť zoznam cieľov sprintov jednotlivých tímov na jednej wiki stránke (alebo kdekoľvek) a dať tento zoznam na význačné miesto tak, aby (nielen top manažment) vedel čo spoločnosť robí – a prečo!

Rozhodovanie, ktoré stories budú zahrnuté do sprintu

Jednou z hlavných aktivít mítingu plánovania sprintu je rozhodnutie, ktoré zo stories budú zahrnuté do sprintu.

Presnejšie, ktoré stories z produktového backlogu budú skopírované do backlogu sprintu.



Pozrite sa na prechádzajúci obrázok. Každý obdĺžnik reprezentuje story, utriedené podľa dôležitosti. Najdôležitejšia story je navrchu zoznamu. Veľkosť každého štvorca reprezentuje veľkosť tejto story (teda čas odhadnutý v story pointoch). Výška celého modrého stĺpca reprezentuje tímom *predpokladanú rýchlosť*, teda koľko story pointov je tím presvedčený, že ukončí počas nasledujúceho sprintu.

Backlog sprintu napravo je momentkou stories z produktového backlogu. Reprezentuje zoznam stories, ktoré sa tím zaviaže urobiť v tomto sprinte.

Tím rozhoduje koľko stories bude zahrnutých do sprintu. Nie produktový vlastník ani ktokoľvek iný.

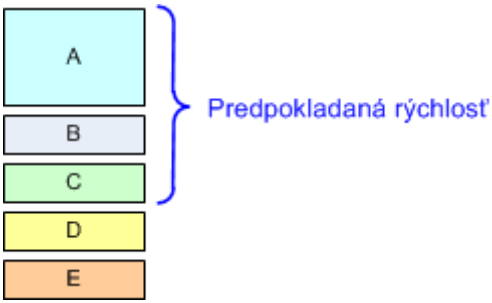
To prináša dve otázky:

1. Ako sa tím rozhodne, ktoré stories budú zahrnuté do sprintu?
2. Ako môže produktový vlastník ovplyvniť ich rozhodnutie?

Začnem druhou otázkou.

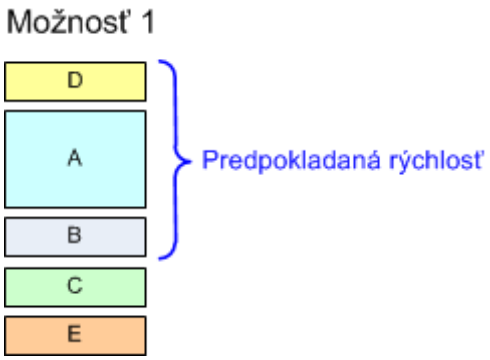
Ako môže produktový vlastník ovplyvniť, ktoré stories urobiť v sprinte?

Povedzme, že máme nasledujúcu situáciu počas mítingu plánovania sprintu.



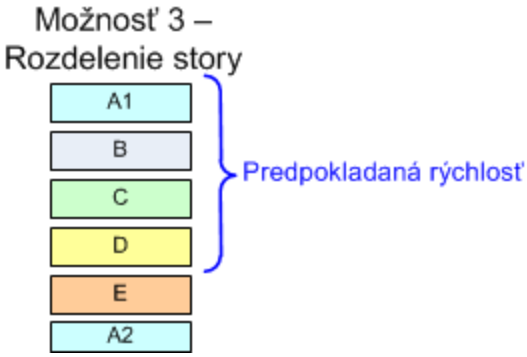
Produktový vlastník je sklamaný, že story D nie je zahrnutá do sprintu. Aké sú jeho možnosti počas mítingu plánovaná sprintu?

Jednou je zmena priority. Ak dá položke D vyššiu dôležitosť, tím bude zaviazaný pridať ju do sprintu ako prvú (v tomto prípade preskočí story C).



Druhou možnosťou je zmena rozsahu – redukciou rozsahu story A až kým tím neuverí, že story D sa zmestí do sprintu.

Treťou možnosťou je rozdelenie story. Produktový vlastník sa môže rozhodnúť, že existujú aspekty story A, ktoré v skutočnosti nie sú tak dôležité, takže rozdelí story A na dve story A1 a A2 s rôznymi úrovňami dôležitosti.



Ako môžete vidieť, napriek tomu, že produktový vlastník nemôže ovplyvňovať predpokladanú rýchlosť, existuje mnoho spôsobov, ktorými môže ovplyvniť aké stories budú v sprinte.

Ako sa tím rozhodne, ktoré stories zahrnúť do sprintu?

Používame dve techniky:

1. Pocit
2. Výpočet rýchlosti

Odhadovanie použitím dobrého pocitu

- **Scrum master**: „Páni, môžeme dokončiť v tomto sprinte story A?" (ukazujúc na najdôležitejšiu položku v produktovom backlogu)
- **Lisa**: „Uhh. Samozrejme že môžeme. Máme 3 týždne a toto je celkom jednoduchá vlastnosť."
- **Scrum master**: „OK, čo ak pridáme aj story B?" (ukazuje na druhú najdôležitejšiu položku)
- **Tom & Lisa jednohlasne**: „Nič sa tým nezmení"
- **Scrum master**: „OK, a čo tak story A a B a C?"
- **Sam (produktovému vlastníkovi)**: „Zahŕňa story C pokročilejšie spracovanie chýb?"
- **Produktový vlastník**: „Nie, toto môžete nateraz preskočiť, implementujte iba základné spracovanie chýb."
- **Sam**: „Potom C môžeme zahrnúť."
- **Scrum master**: „OK, čo ak pridáme story D?"
- **Lisa**: „Hmmm...."
- **Tom**: „Myslím si, že to môžeme urobiť."
- **Scrum master**: „Presvedčený na 90%? 50%?"
- **Lisa & Tom**: „Skoro 90%".
- **Scrum master**: „OK, D je zahrnutá. Čo ak pridáme story E?"
- **Sam**: „Možno."
- **Scrum master**: „90%? 50%?"
- **Sam**: „Povedal by som, že viac k 50%".
- **Lisa**: „Nie som si istá."
- **Scrum master**: „OK, tak ju nechajme. Potvrdzujeme A, B, C, a D. Ak budeme môcť. Tak ukončíme aj E, ale nikto s tým nemôže rátať, takže ju necháme mimo sprintu. Čo vy nato?"

- **Každý**: „OK!"

Pocit funguje pre malé tímy a krátke sprinty.

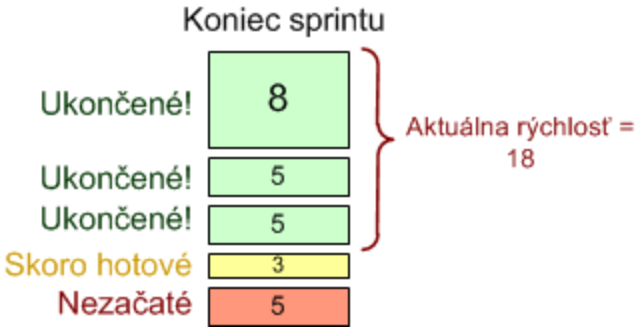
Odhad použitím výpočtov rýchlosti

Táto technika pozostáva z dvoch krokov:

1. Určte *predpokladanú rýchlosť*
2. Vypočítajte koľko stories môžete zahrnúť bez prekročenia odhadovanej rýchlosti

Rýchlosť je miera "množstva dokončenej práce", pričom každá položka je vážená za podmienok počiatočného odhadu.

Obrázok nižšie ukazuje príklad *odhadovanej rýchlosti* na začiatku sprintu a *aktuálnej rýchlosti* na konci tohto sprintu. Každý obdĺžnik je story, číslo vnútri je počiatočný odhad tejto story.



Všimnite si, že aktuálna rýchlosť je založená na *počiatočných* odhadoch každej story. Akékoľvek zmeny odhadov story urobené počas sprintu sú ignorované.

Už počujem vaše protesty: „Je to použiteľné? Vysoká alebo nízka rýchlosť môže závisieť na celom množstve faktorov! Hlúpi programátori, nesprávne počiatočné odhady, neplánované prerušenia počas sprintu, atď!"

Súhlasím, je to hrubé číslo. Ale stále to je použiteľné číslo, hlavne v porovnaní s ničím. Poskytuje vám niektoré vážne fakty: „bez ohľadu na dôvody, tu je aproximovaný rozdiel medzi tým čo sme si mysleli, že sme schopní a tým, čo sme skutočne ukončili.“.

A čo story, ktoré sú *skoro* dokončené počas sprintu? Prečo nezahrnieme aspoň čiastočné body do našej aktuálnej rýchlosti? Nuž, aby sme zdôraznili fakt, že Scrum (a fakticky celý agilný softwarový vývoj a lean manufacturing vo všeobecnosti) je o úplnom dokončení vecí, dodaní, hotovo! Hodnota napoly dokončených vecí je nula (fakticky môže byť negatívna). Prečítajte si o tom viac v „Managing the Design Factory“ Donalda Reinertsena alebo jednu z kníh autorov Poppendieck.

Takže pomocou akého kúzla urobíme odhad rýchlosti?

Jedným jednoduchým spôsobom odhadu rýchlosti je pozrieť sa na históriu tímu. Aká bola ich rýchlosť počas niekoľkých predchádzajúcich sprintov? Potom usúďte, že rýchlosť bude zhruba rovnaká aj v ďalšom sprinte.

Táto technika je známa ako *včerajšie počasie*. Je vhodná pre tímy, ktoré už dokončili niekoľko sprintov (takže je dostupná štatistika) a aj nasledujúci sprint budú robiť rovnako, s rovnakou veľkosťou tímu a rovnakými pracovnými podmienkami, atď. Samozrejme nie vždy je možný tento postup.

Sofistikovanejším variantom je jednoduchý prepočet zdrojov. Povedzme, že plánujeme 3 týždňové sprinty (15 pracovných dní) s 4 členným tímom. Lisa bude na dovolenke 2 dni. Dave je dostupný len na 50% a bude na dovolenke 1 deň. Zhrnutím dohromady...

…vyjde 50 dostupných človekodní v tomto sprinte.

Je to naša odhadovaná rýchlosť? Nie! Keďže našou jednotkou odhadu je *story point* ktoré, v našom prípade, korešpondujú zhruba „ideálnym človekodňom“. Ideálny človekodeň je veľmi efektívny, nerušený deň, ktorý je vzácny. Mimo to musíme rátať s vecami ako neplánovaná práca pridaná do sprintu, choroby ľudí atď

Takže naša predpokladaná rýchlosť bude určite menej ako 50. Ale o koľko menej? Budeme používať pojem „faktor pozornosti“.

PREDPOKLADANÁ RÝCHLOSŤ TOHTO SPRINTU

(DOSTUPNÉ ČLOVEKODNI) x (FAKTOR POZORNOSTI) = (PREDPOKLADANÁ RÝCHLOSŤ)

Faktor pozornosti je predpoklad ako je tím sústredený. Nízky faktor pozornosti znamená, že tím očakáva mnoho rušenia alebo očakáva, že odhad je optimistický.

Najlepším spôsobom ako rozpoznať rozumný faktor pozornosti je pozrieť sa na posledný sprint (alebo ešte lepšie na pár posledných sprintov).

FAKTOR POZORNOSTI POSLEDNÉHO SPRINTU

$$(\text{FAKTOR POZORNOSTI}) = \frac{(\text{AKTUÁLNA RÝCHLOSŤ})}{(\text{DOSTUPNÉ ČLOVEKODNI})}$$

Aktuálna rýchlosť je sumou počiatočných odhadov všetkých stories ukončených počas posledného sprintu.

Povedzme, že počas posledného sprintu bolo ukončených 18 story pointov použitím 3 členného tímu pozostávajúceho z Toma, Lisy a Sama pracujúcich 3 týždne, čo je celkom 45 človekodní. A teraz sa pokúšame odhadnúť predpokladanú rýchlosť nasledujúceho sprintu. Aby to bolo komplikovanejšie, nový člen, Dave, sa pripojí k tímu v tomto sprinte. Ak vezmeme do úvahy dovolenku a zdroje, v nasledujúcom sprinte máme 50 človekodní.

FAKTOR POZORNOSTI POSLEDNÉHO SPRINTU

$$40\% = \frac{18 \text{ STORY POINTOV}}{45 \text{ ČLOVEKODNI}}$$

PREDPOKLADANÁ RÝCHLOSŤ TOHTO SPRINTU

50 ČLOVEKODNI x 40% = 20 STORY POINTOV

Takže naša očakávaná rýchlosť pre nasledujúci sprint je 20 story pointov. To znamená, že tím môže pridať do sprintu stories až kým nedosiahnu približne 20 story pointov.

V tomto prípade tím môže vybrať 4 horné story s celkovo 19 story pointami alebo 5 stories s celkovo 24 story pointami. Povedzme, že si vyberú 4 story keďže to je najbližšie k 20 story pointom. Ak si nie ste istí, zvoľte menej stories.

Pretože tieto 4 stories majú celkovo 19 story pointov, ich celková očakávaná rýchlosť pre tento sprint je 19.

„Včerajšie počasie“ je praktická technika, ale použite ju s dávkou spoločného porozumenia. Ak posledný sprint bol neobvykle zlým sprintom, pretože väčšina tímu bola celý týždeň chorá, potom je bezpečnejšie predpokladať, že nabudúce nebudete mať také nešťastie a pre nasledujúci sprint môžete predpokladať vyšší faktor pozornosti. Ak tím práve nainštaloval nový, rýchlosťou blesku fungujúci systém kontinuálneho prekladu, tak pravdepodobne taktiež môžete zvýšiť faktor pozornosti. Ak sa do sprintu zapojil nový človek, potrebujete znížiť faktor pozornosti, aby ste vzali do úvahy potrebu jeho vzdelávania atď. Kedykoľvek to je možné, pozrite sa niekoľko sprintov dozadu a spriemerujte si čísla, aby ste dostali viac spoľahlivé predpoklady.

Čo ak tím je úplne nový takže nemáte žiadnu štatistiku? Pozrite sa na faktor pozornosti iného tímu s podobnými podmienkami.

Čo ak nemáte žiaden iný tím, na ktorý by ste sa pozreli? Hádajte faktor pozornosti. Dobrá správa je, že vaša domnienka bude použitá iba v prvom sprinte. Po ňom už budete mať štatistiku a môžete priebežne merať a zlepovať váš faktor pozornosti a predpoklad rýchlosti.

„Štandardný“ faktor pozornosti, ktorý používam, je obyčajne 70%, pretože to je hodnota, ktorú dosiahla väčšina našich tímov.

Ktorú techniku odhadu používame?

Spomínal som vyššie niekoľko techník – „dobrý pocit“, výpočet rýchlosti podľa „včerajšieho počasia“ a výpočet rýchlosti založený na dostupných človekodňoch a predpokladaného faktoru pozornosti.

Takže ktorú techniku používame?

Zvyčajne kombinujeme do určitej miery všetky tieto techniky. Netrvá to dlho.

Pozrieme sa na faktor pozornosti a aktuálnu rýchlosť z posledného sprintu. Pozrieme sa na celkovú dostupnosť zdrojov v tomto sprinte a odhadneme faktor pozornosti. Diskutujeme o rozdieloch medzi týmito dvoma faktormi pozornosti a urobíme úpravy ako to je potrebné.

Keď už máme predbežný zoznam stories, ktoré budú v sprinte, urobím kontrolu „dobrým pocitom“. Spýtam sa tímu, aby nateraz ignorovali čísla a aby sa zamysleli či tieto pocity či je to reálne množstvo pre tento sprint. Ak cítia, že je toho veľa, odoberieme jednu, dve story. A naopak.

Na konci dňa je cieľ sa jednoducho rozhodnúť ktoré stories budú v sprinte. Faktor pozornosti, dostupnosť zdrojov a predpokladaná rýchlosť sú iba prostriedkom pre dosiahnutie tohto cieľa.

Prečo použiť indexové karty

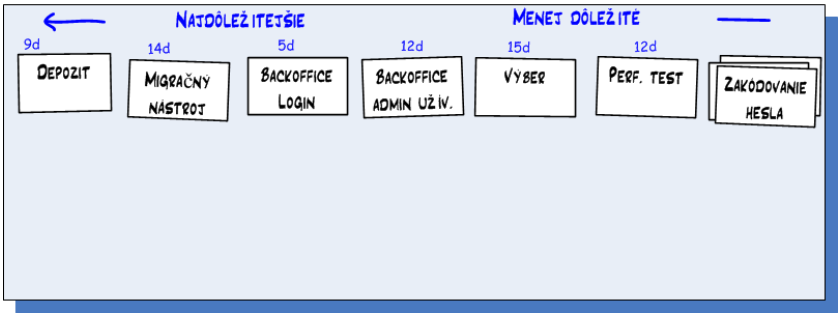
Väčšina porady plánovania sprintu je strávená so stories v backlogu. Ich odhadovanie, zmena priorít, ich vyjasnenie, delenie na menšie atď...

Ako to robíme prakticky?

Nuž, tímy zvyčajne zvyknú naštartovať projektor, zobrazia backlog v Exceli a jeden z nich (obyčajne produktový vlastník alebo scrum master) chytia klávesnicu, zamrmlú každú story a otvoria diskusiu. Ako tím a produktový vlastník diskutujú priority a detaily, človek za klávesnicou zaktualizuje story priamo v Exceli.

Znie to dobre? Nuž nie je to tak. Obyčajne sa to pokazí. A čo je horšie, tím normálne ani *nezistí*, že sa to pokazilo pokiaľ nedosiahnu koniec mítingu a nezistia, že ešte stále neprešli cez celý zoznam stories!

Riešenie, ktoré funguje najlepšie je vytvoriť *indexové karty* a *umiestniť ich na stenu* (alebo na veľký stôl).

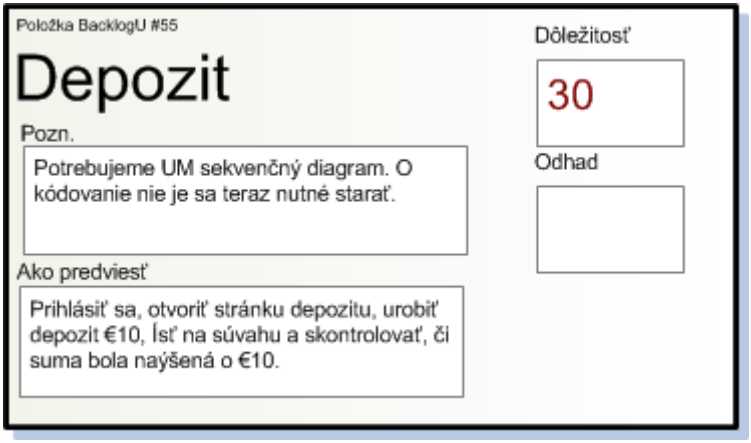


Toto je perfektné užívateľské rozhranie v porovnaní s počítačom a projektorom pretože:

- Ľudia stoja a chodia okolo => vnímajú a upozorňujú dlhšie.
- Každý sa cíti viac zainteresovaný (viac než len chlapík za klávesnicou).
- Súčasne môže byť editovaných viacero stories.
- Zmena priority je triviálna – iba ďalej posuniete indexovú kartu.

- Na konci mítingu môžu byť indexové karty prinesené priamo do tímovej miestnosti a použité na tabuli úloh umiestnenej na stene (pozri str. 55 „Ako tvoríme backlogy sprintov“)

Môžete ich napísať rukou alebo (ako to zvykneme robiť my) použiť jednoduchý skript pre vygenerovanie indexových kariet, ktoré je možné tlačiť priamo z produktového backlogu.



PS – skript je dostupný na mojom blogu

http://blog.crisp.se/henrikkniberg.

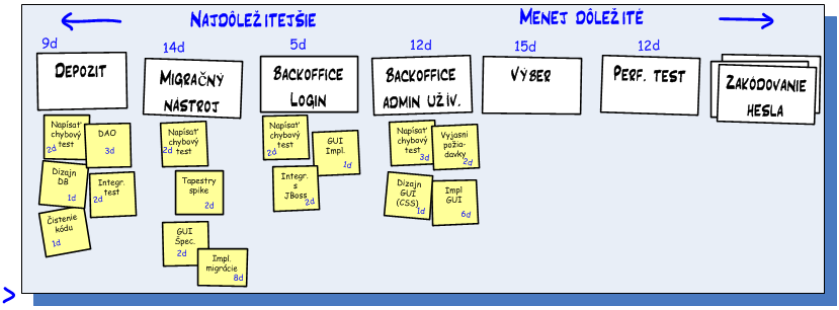
Dôležité: Po mítingu plánovania sprintu náš Scrum Master ručne zaktualizuje produktový backlog v Exceli so zohľadnením všetkých zmien urobených na fyzické indexové karty. Áno, je to tak trochu administratívna nepríjemnosť, ale zistili sme, že je to perfektne akceptovateľné so zohľadnením o koľko je míting plánovania sprintu s fyzickými indexovými kartami efektívnejší.

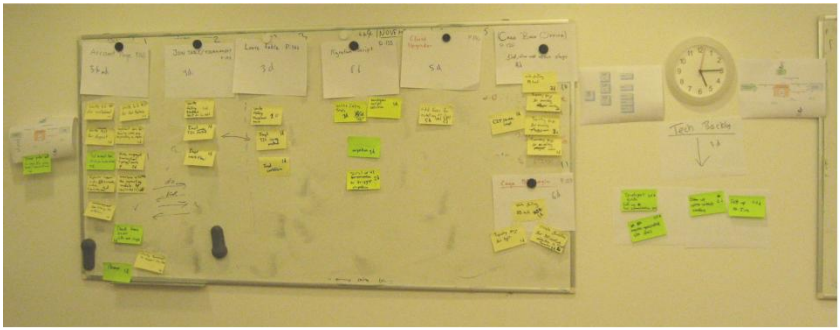
Jedna poznámka o atribúte „Dôležitosť“. Je to dôležitosť špecifikovaná v produktovom backlogu v Exceli v čase tlače. Je jednoduchšie triediť karty ak dôležitosť máte na karte (normálne umiestňujeme najdôležitejšie položky naľavo, menej dôležité napravo). Ak sú už ale karty na stene, môžete ignorovať hodnotenie dôležitosti a namiesto toho použite fyzické usporiadanie na stene ako relatívnu

indikáciu hodnotenia dôležitosti. Ak produktový vlastník vymení dve karty, nestrácajte čas aktualizáciou dôležitosti na papieri. Uistite sa iba, že po mítingu bola v produktovom backlogu v Exceli zaktualizovaná dôležitosť.

Je obyčajne ľahšie odhadnúť čas (a presnejšie), ak je story rozdelená na úlohy. V skutočnosti používame slovo „aktivita“, pretože „task“ vo švédštine znamená niečo úplne iné :o) S našimi indexovými kartami je aj toto pekné a jednoduché. Môžete rozdeliť tím na páry a každý pár môže deliť story paralelne.

Fyzicky to robíme pridaním malých post-it papierikov pod každú story, každý post-it zobrazuje jednu úlohu v danej story.





Produktový backlog v Exceli neaktualizujeme so zohľadnením rozdelenia do úloh z dvoch dôvodov:

- Rozdelenie do úloh je prechodné, napr. ak sú často menené a spresnené počas sprintu, takže je často

nepríjemné udržiavať synchronizovaný produktový backlog v Exceli.

- Produktový vlastník nemusí byť zúčastnený v tejto úrovni detailov.

Podobne ako indexová karta story aj úlohy môžu byť priamo použité v backlogu sprintu (pozrite str. 55 „Ako tvoríme backlogy sprintov“).

Definícia „hotovo“

Je dôležité aby sa produktový vlastník a tím dohodol na jasnej definícii „hotovo“. Je story hotová keď je celý kód zapísaný do repozitára? Alebo je hotová, keď bol umiestnený do testovacieho prostredia a overená tímom integračných testov? Kedykoľvek je to možné používame pre definíciu hotovo „pripravený pre umiestnenie do produkcie“, ale niekedy musíme pracovať s definíciou „umiestnené na testovací server a pripravené pre akceptačné testy“

Na začiatku sme zvykli mať detailný kontrolný zoznam. Teraz často iba povieme „story je hotová, keď tak povie tester v Scrum tíme“. Je potom na testerovi uistiť sa, že zámer produktového vlastníka je pochopený tímom a že položka je dostatočne „hotová“ pre splnenie akceptačnej definície pre stav hotovo.

Pochopili sme, že sa nedá so všetkými stories zaobchádzať rovnako. So story nazvanou „Formulár Hľadanie užívateľov“ bude nakladané inak ako so story „Operačný manuál“. V druhom prípade definícia „hotovo“ môže jednoducho znamenať „akceptované operačným tímom“. To je dôvod prečo je všeobecnejší zmysel často lepší než formálny kontrolný zoznam.

Ak sa často spletiete pri definícii hotovo (čo sa nám stávalo na začiatku), potom by ste mali mať položku „definícia hotovo“ na každej individuálnej story.

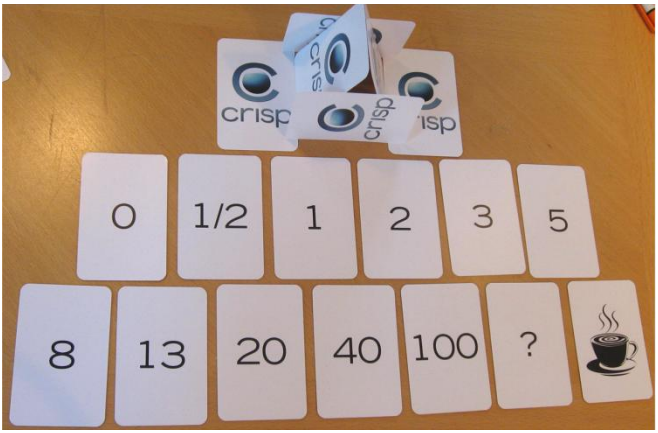
Odhadovanie času použitím plánovacieho pokru

Odhadovanie je tímovou aktivitou – každý člen tímu je obyčajne zahrnutý do odhadovania každej story. Prečo?

- Na začiatku plánovania normálne nevieme presne kto bude implementovať ktorú časť ktorých stories.
- Stories normálne zasahujú viac ľudí a rôzne typy znalostí (dizajn užívateľského rozhrania, kódovanie, testovanie atď).
- Aby sme stanovili odhad, člen tímu potrebuje nejaké pochopenie, ktorá story je o čom. To zvýši pravdepodobnosť, že členovia tímu si môžu pomôcť navzájom pochopiť o čom story je. Zároveň to zvýši pravdepodobnosť, že dôležité otázky o story sa objavia skoro.
- Pýtaním sa každého na odhad story často objavíme nezrovnalosti keď dvaja rôzni členovia tímu majú veľmi rozdielne odhady pre tú istú story. Na túto situáciu je dobré prísť a diskutovať o nej radšej skôr ako neskôr.

Ak sa spýtate tímu, aby poskytol odhad, bežne prvými, ktorí vyhŕknu, budú ľudia, ktorí najlepšie chápu story. Nanešťastie to veľmi silno ovplyvní odhad každého iného.

Je tu však excelentná technika, ktorá tomu predchádza – nazýva sa plánovací poker (myslím, že ju vymyslel Mike Cohn).



Každý člen tímu dostane kôpku 13 kariet ako je to na obrázku vyššie. Kedykoľvek je story odhadovaná, každý člen tímu si vyberie kartu, ktorá reprezentuje jeho odhad času (v story pointoch) a umiestni ju na stôl číslom dole. Keď ich už všetci položili, karty sa naraz odkryjú. Takto je každý člen tímu nútený premýšľať za seba a nie sa prikláňať k odhadu niekoho iného.

Ak sú veľké rozdiely medzi dvoma odhadmi, tím diskutuje rozdiely a skúša si vytvoriť spoločný obraz o akú prácu v story pôjde. Môžu urobiť rozdelenie na úlohy. Potom tímu odhaduje opäť. Tento cyklus je opakovaný pokiaľ časový odhad sa nezbližuje, teda pokiaľ všetky odhady nie sú pre danú story *približne* rovnaké.

Je dôležité pripomenúť členom tímu, že odhadujú celkové množstvo práce zahrnutej v story. Nie iba „jeho“ časť práce. Tester nesmie odhadovať iba množstvo práce s testami.

Upozorňujem, že sekvencia čísiel je nelineárna. Napr. žiadna hodnota nie je medzi 40 a 100. Prečo?

Je to preto, aby sa predišlo falošným pocitom presnosti pre odhady veľkých časov. Ak je story odhadovaná na približne 20 story pointov, nie je relevantné diskutovať o tom, či má byť 20 alebo 18 alebo 21. Všetko čo vieme je, že je to veľká story a že je ťažké ju odhadnúť- Takže 20 je náš približný odhad.

Chcete presnejší odhad? Rozdeľte story na menšie stories a namiesto toho odhadnite menšie!

A nie, nemôžete podvádzať kombinovaním 5 a 2 aby ste dostali 7. Musíte si vybrať 5 alebo 8, neexistuje žiadna 7.

Upozornenie na niektoré špeciálne karty:

- 0 = „táto story je už skoro hotová alebo „táto story je skoro nič, niekoľko minút práce ”.
- ? = „Nemám absolútne predstavu. Nič.“
- Šálka kávy = „Nevládzem premýšľať. Dajme si krátku prestávku.“

Vyjasnenie stories

Najhoršie čo sa môže stať je, keď tím hrdo demonštruje počas dema sprintu novú vlastnosť a produktový vlastník sa zamračí a povie „nuž toto je pekné, ale *nie je to to čo som chcel!“*.

Ako sa uistíte, že chápanie produktového vlastníka zodpovedá chápaniu tímu? Alebo že každý člen tímu chápe každú story rovnako? Nuž, nemôžete sa uistiť. Existujú ale jednoduché techniky pre identifikáciu najočividnejších nepochopení. Najjednoduchšou technikou je sa jednoducho uistiť, že všetky položky každej story sú vyplnené (alebo presnejšie, pre každú story, ktorá má dôležitosť dostatočne vysokú nato, aby bola zvažovaná do tohto sprintu).

Príklad 1:

Tím a produktový vlastník sú spokojní s plánom sprintu a pripravení ukončiť míting. Scrum master povie „Počkajte chvíľu, táto story nazvaná ‘pridať užívateľa’ - nie je tu odhad. Odhadnime ju!” Po pár kolách plánovacieho pokru sa tím zhodol na 20 story pointoch. Nato produktový vlastník zúrivo vstal s krikom “Čožeeeee?!”. Po niekoľkých minútach horúcej diskusie sa vyjasnilo, že tím nepochopil rozsah ‘pridať užívateľa’; tím si myslel, že to znamená „pekné webové rozhranie pre pridanie, odstránenie, mazanie a hľadanie užívateľov”, zatiaľ čo produktový vlastník tým myslel ‘manuálne pridanie

užívateľov pomocou SQL smerom k DB'. Opäť urobili odhad a zhodli sa na 5 story pointoch.

Príklad 2:

Tím a produktový vlastník sú spokojní s plánom sprintu a pripravení ukončiť míting. Scrum master povie „Počkajte chvíľu, táto story nazvaná 'pridať užívateľa', ako by mala byť demonštrovaná?" Nato prebehlo nejaké mumlanie a po minúte niekto vstal a povedal "najprv sa prihlásime na webovú stránku a potom …" a produktový vlastník preruší "Prihlásiť sa k webovej stránke?! Nie, nie, nie, táto funkčnosť vôbec nesmie byť súčasťou webovej stránky, mal by to byť smiešny malý SQL skript určený iba pre technických administrátorov".

Popis story „ako demonštrovať" by mal (a aj musí) byť veľmi *stručný*! Inak neukončíte míting plánovania sprintu načas. Je to v podstate popis na vyššej úrovni ako vykonať najtypickejší testovací scenár manuálne: „Urob to, potom to a potom to skontroluj".

Zistil som, že tento jednoduchý popis *často* odkryje dôležité nedorozumenia o rozsahu story. Je dobré to zistiť skôr, že?

Rozdelenie story na menšie stories

Stories nemajú byť ani veľmi malé ani veľmi veľké (ak sa bavíme o odhadoch). Ak máte niekoľko 0,5 bodových stories, stanete sa pravdepodobne obeťou mikromanažmentu. Na druhej strane, 40 bodová story sa stáva vysoko rizikovou, že bude dokončená iba *sčasti*, čo neprinesie žiadnu hodnotu spoločnosti a iba zvýši administráciu. Okrem toho, ak je vaša odhadovaná rýchlosť 70 a vaše dve najprioritnejšie stories sú každá ohodnotené 40 story pointami, plánovanie sa stane ťažšie. Musíte si zvoliť medzi pod plánovaním (zahrnie sa iba jedna story) alebo preplánovaním (zahrnú sa obidve story).

Zistil som, že skoro vždy je možné rozdeliť veľkú story na menšie. Iba sa uistite, že menšie stories stále reprezentujú niečo uskutočniteľné s obchodnou hodnotou.

Normálne sa snažíme mať stories hodnotené 2-8 človekodňami. Naša rýchlosť je obyčajne okolo 40-60 pre typický tím, takže to nám dáva okolo 10 stories na sprint. Niekedy 5 a niekedy až 15. Je to manažovateľný počet indexových kariet.

Rozdelenie stories na úlohy

Počkajte chvíľu, aký rozdiel je medzi „úlohami“ a „stories“? Otázka je namieste.

Rozdiel je celkom jednoduchý. Stories sú výsledky, o ktoré sa stará produktový vlastník. Úlohy nie sú o výsledku, alebo veci o ktoré sa produktový vlastník nestará.

Príklad rozdelenia story na menšie stories:

Príklad rozdelenia story na úlohy:

Zaujímavé postrehy:

- Nové Scrum tímu sa zdráhajú tráviť čas delením kopy stories na priamo úlohy. Niektorí to pociťujú ako prístup typu vodopád.
- Pre jasne pochopiteľné stories je to jednoduché urobiť to hneď.
- Takéto delenie často odkryje dodatočnú prácu čo zapríčiní, že odhad času sa zvýši a tým aj dá realistickejší plán sprintu.
- Tento typ rozdelenia robí denné scrum mítingy efektívnejšími (pozri str. 84 „Ako robíme denné scrumy“
- Ak aj je rozdelenie nepresné a zmení sa keď sa začne práca, vyššie uvedené výhody pretrvávajú.

Takže sa pokúsime vytvoriť čas plánovania sprintu dosť dlhý nato, aby sme to zahrnuli, ale ak čas vyprší, skončíme. (pozri nižšie „Kde nakresliť čiaru“)

Poznámka – praktikujeme TDD (vývoj riadený testami) čo znamená, že prvá úloha pre väčšinu story je „napísať test chyby“ a poslednou úlohou je „refaktorovať“ (= zlepšiť čitateľnosť kódu a odstrániť duplikácie).

Definícia času a miesta denného scrumu

Jedným často zabudnutým výstupom mítingu plánovania sprintu je „definovaný čas a miesto denného scrumu“. Bez toho váš sprint začne zle. Prvý denný scrum je v podstate štart, kde sa všetci rozhodujú kde začať pracovať.

Preferujem ranné mítingy. Ale musím potvrdiť, že sme v skutočnosti neskúsili urobiť denné scrumy poobede alebo uprostred dňa.

Nevýhody poobedňajších scrumov: keď prídete ráno do práce, musíte si pamätať čo ste včera povedali ľuďom o tom, čo budete dnes robiť.

Nevýhody ranných scrumov: keď prídete ráno do práce, musíte si pamätať čo ste robili včera tak, že o tom môžete reportovať.

Môj názor je, že prvá nevýhoda je horšia, pretože najdôležitejšia vec je čo *budem robiť*, nie čo som *urobil*.

Naša štandardná procedúra je vybrať si najskorší čas kedy nikto z tímu nemrmle. Obyčajne 9:00, 9:30 alebo 10:00. Najdôležitejšia vec je, že je to čas ktorý všetci v tíme môžu bezvýhradne akceptovať.

Kde nakresliť čiaru

Ok, takže čas beží. Zo všetkých vecí, ktoré chceme urobiť počas plánovania sprintu, ktoré odrežeme ak nebudeme mať dosť času?

Ja používam nasledujúci zoznam:

Priorita 1: Cieľ sprintu a dátum dema. Toto je to najmenej čo potrebujete pre štart sprintu. Tím má cieľ, dátum konca a môžu pracovať priamo z produktového backlogu. Smrdí to, áno, a musíte vážne zvážiť načasovanie mítingu plánovania sprintu, ale ak skutočne potrebujete naštartovať sprint potom to aj pravdepodobne urobíte. Ak mám povedať pravdu, nikdy som nenaštartoval sprint s tak málo informáciami.

Priorita 2: Zoznam stories, ktoré tím akceptoval pre tento sprint.

Priorita 3: Odhad je vyplnený pre každú story sprintu.

Priorita 4: „Ako demonštrovať" vyplnené v každej story sprintu.

Priorita 5: Výpočet rýchlosti a zdrojov, ako kontrola pre váš plán sprintu. Obsahuje zoznam členov tímu a ich povinnosti (inak nebudete vedieť vyrátať rýchlosť).

Priorita 6: Špecifický čas a miesto pre denný scrum. Trvá to iba moment rozhodnúť sa, ale ak nemáte čas, Scrum master sa o tom môže rozhodnúť po mítingu a poslať všetkým emailom.

Priorita 7: Rozdelenie stories na úlohy. Toto rozdelenie môže byť urobené aj denne počas denných scrumov, ale to môže narušiť tok sprintu.

Technické stories

Je tu aj komplexná otázka: technické stories. Alebo položky netýkajúce sa funkčnosti alebo akokoľvek ich chcete nazvať.

Veci, ktoré majú byť urobené, ale nie sú dodávané, ani priamo vo vzťahu k špecifickým stories, a ani priama hodnota pre produktového vlastníka.

My ich voláme „technické stories".

Napríklad:

- **Inštalovať kontinuálny build server**
    - Prečo to musí byť urobené: pretože to ušetrí obrovské množstvo času pre vývojárov a redukuje riziko big-bang integračných problémov na konci iterácie.
- **Napísať celkový dizajn systému**
    - Prečo to musí byť urobené: Pretože vývojári stále zabúdajú na celkový dizajn, a preto píšu nekonzistentný kód. Je tu potreba mať prehľadný dokument pre udržanie všetkých na rovnakej úrovni vedomostí.
- **Refaktoring DAO vrstvy**
    - Prečo to musí byť urobené: Pretože vrstva DAO sa skutočne skomplikovala a stojí každého čas kvôli popleteniu sa a zbytočným chybám. Vyčistenie kódu ušetrí všetkým čas a zlepší robustnosť systému.

- **Upgrade systému Jira** (bug tracker)
    - Prečo to musí byť urobené: Aktuálna verzia je veľmi chybová a pomalá, upgrade ušetrí všetkým čas.

Sú to stories v normálnom slova zmysle? Alebo sú to úlohy, ktoré nie sú spojené so žiadnou story? Kto ich prioritizuje? Týka sa to produktového vlastníka?

Experimentovali sme s rôznymi spôsobmi nakladania s technickými stories. Skúšali sme ich považovať za prvotriedne stories podobne ako akékoľvek iné. Nebolo to dobré, keď produktový vlastník prioritizoval produktový backlog, bolo to ako keby porovnával jablká s pomarančmi. V skutočnosti, pre zjavné dôvody, technickým stories bola často priradená nízka priorita s motiváciou ako „som si istý, že kontinuálny buildovací server je dôležitý, ale mohli by sme najskôr vybudovať nejaké vlastnosti vedúce k zisku? Potom môžete pridať vaše technické bonbóniky, ok?

V niektorých prípadoch má produktový vlastník pravdu, ale nie často. Usúdili sme, že produktový vlastník nie je vždy kvalifikovaný urobiť dohodu.

Takže čo robíme:

1) Pokúste sa vyhnúť technickým stories. Tvrdo hľadajte spôsob ako transformovať technickú story na normálnu story s merateľnou obchodnou hodnotou. Týmto spôsobom má produktový vlastník lepšiu šancu správne sa dohodnúť.

2) Ak nemôžeme transformovať technickú story na normálnu story, zistite, či táto práca nemôže byť urobená ako úloha v inej story. Napr. „refaktorovať DAO vrstvu“, môže byť úlohou v story „editácia užívateľa“, pretože vyžaduje DAO vrstvu.

3) Ak vyššie uvedené postupy zlyhajú, definujte ju ako technickú story a udržujte oddelený zoznam takýchto stories. Produktový vlastník ich môže vidieť, ale nie upravovať. Použite „faktor pozornosti“ a „odhadovanú rýchlosť“

pre dohadovanie sa s produktovým vlastníkom a nechajte si nejaký čas na implementáciu technických stories.

Príklad (dialóg podobný tomuto sa vyskytol počas jedného z našich mítingov plánovania sprintov).

- **Tím:** „Máme tu nejaké interné technické záležitosti, ktoré musia byť dokončené. Dáme tomu 10% nášho času, teda redukujeme faktor pozornosti zo 75% na 65%. Je to ok?“
- **Produktový vlastník:** „Do pekla nie. Nemáme čas!“
- **Tím:** „Nuž, pozri sa na posledný sprint (všetky hlavy sa otočili na rýchlosť načarbanú na tabuli). Naša odhadovaná rýchlosť bola 80 a naša aktuálna rýchlosť je 30, správne?“
- **PO:** „Správne! Takže nemáme čas robiť interné technické záležitosti! Potrebujeme nové vlastnosti!“
- **Tím:** „Nuž, *dôvod* prečo sa naša rýchlosť zhoršila bol, že sme strávili príliš veľa času pokusmi dať dokopy konzistentné releasy pre testovanie“.
- **PO:** „Áno a?“
- **Tím:** „Naša rýchlosť bude naďalej podobné zlá ak s tým niečo neurobíme.“
- **PO:** „Áno a?“
- **Tím:** „Takže navrhujeme odstrániť z tohto sprintu približne 10% pre nastavenie servera kontinuálneho buildu a iných vecí, ktoré odstránia bolesti integrácie. Toto pravdepodobne zvýši našu rýchlosť sprintu *najmenej o 20%* pre každý ďalší nasledujúci sprint, navždy!“
- **PO:** „Ó, skutočne? Prečo sme to potom neurobili v poslednom sprinte?“
- **Tím:** „Hmmm, pretože si to po nás nechcel...“

- **PO:** „Oh, um, nuž dobre, znie to ako dobrý nápad to urobiť práve teraz!“

Samozrejme, že iná možnosť je držať produktového vlastníka mimo alebo mu dať nemožný faktor pozornosti. Ale neexistuje žiadne ospravedlnenie *neskúsiť* dosiahnuť najprv konsenzus.

Ak produktový vlastník je kompetentný a rozumný partner (a my sme také šťastie mali), odporúčam aby bol informovaný čo najviac a nechať ho robiť celkové priority. Transparentnosť je jednou zo základných hodnôt Scrumu, že?

Systém sledovania chýb verzus produktový backlog

Máme tu zložitý problém. Excel je dobrým formátom pre produktový backlog. Stále ale potrebujeme systém sledovania chýb a Excel na to nie je dobrý. My používame Jiru.

Tak ako vznášame problémy z Jiri na míting plánovania sprintu? Znamená to, že ich neignorujeme a nezameriavame sa iba na stories.

Skúšali sme viacero stratégií:

1) Produktový vlastník vytlačí položky z Jiri s najvyššou prioritou, prinesie ich na míting plánovania sprintu, dá ich na stenu spolu s inými stories (tým implicitne špecifikuje prioritu týchto chýb v porovnaní s inými stories).

2) Produktový vlastník vytvorí stories, ktoré referujú na chyby v Jire. Napr. „Opraviť najkritickejšie chyby tlače, Jira-124, Jira, Jira-124, Jira-126, a Jira-180“.

3) S opravami chýb sa predpokladá, že sú mimo sprintu, teda, že tím udržiava faktor pozornosti dosť nízko (napr. 50%), aby sa uistil, že má čas pre opravu chýb. Jednoducho sa predpokladá, že tím

strávi určité množstvo času opravou chýb reportovaných v Jire.

4) Uložte produktový backlog v Jire (zbavte sa Excelu). Narábajte s chybami ako s inými stories.

Neuzavreli sme, ktorá stratégia je pre nás najlepšia; v skutočnosti sa to líši tím od tímu a sprint od sprintu. Kvôli lenivosti zvyknem postupovať podľa prvej stratégie. Je pekná a jednoduchá.

Míting plánovania sprintu konečne skončil!

Noo, nikdy som si nemyslel, že by táto kapitola o mítingoch plánovania sprintu mohla byť tak dlhá! Myslím, že to reflektuje môj názor, že míting plánovania sprintu je najdôležitejší v Scrume. Venujte veľa úsilia, aby ste ho urobili správne a ostatné pôjde oveľa ľahšie.

Míting plánovania sprintu je úspešný ak všetci (všetci členovia tímu a aj produktový vlastník) odídu z mítingu s úsmevom, a aj nasledujúce ráno sa prebudia s úsmevom a aj ich prvý denný scrum urobia s úsmevom.

Potom, samozrejme, všetko možné môže ísť dole vodou, ale prinajmenšom nemôžete za to viniť plán sprintu :o)

5

Ako komunikujeme o sprintoch

Je dôležité aby celá spoločnosť bola informovaná o tom, čo sa deje. Inak sa ľudia budú sťažovať, alebo ešte horšie, dospejú k nesprávnym predpokladom o tom, čo sa deje.

Používame „stránku Informácie o sprinte.

Jackass team, sprint 15

Cieľ sprintu
- Release Beta!

Backlog sprintu (odhad v zátvorkách)
- Depozit (3)
- Migračný nástroj (8)
- Backoffice prihlásenie (5)
- Backoffice Administrácia užívateľov (5)

Predpokladaná rýchlosť: 21

Plán
- Perióda sprintu: 6.11.2006 do 24.11.2006
- Denný scrum: 9:30 – 9:45, v tímovej miestnosti
- Demo sprintu: 24.11.2006, 13:00, v bufete

Tím
- Jim
- Erica (scrum master)
- Tom (75%)
- Eva
- John

Niekedy pripojíme aj informáciu aj o tom, ako bude každá story demonštrovaná.

Čo najskôr po mítingu plánovania sprintu vytvorí Scrum master túto stránku, dá ju na wiki a pošle spam do celej spoločnosti.

```
Subject: Jackass sprint 15 started

Hi all! The Jackass team has now started sprint
15. Our goal is to demonstrate a beta-ready
release on nov 24.

See the sprint info page for details:
http://wiki.mycompany.com/jackass/sprint15
```

Na našej wiki máme aj „dashboard“, ktorý odkazuje na všetky trvajúce sprinty.

Korporátny Dashboard

Aktuálne sprinty

- Team X sprint 15
- Team Y sprint 12
- Team Z sprint 1

Navyše, Scrum master vytlačí stránku s informáciami o sprinte a dá ju na stenu pred tímovou miestnosťou. Ktokoľvek okoloidúci sa môže pozrieť na stránku s informáciami o sprinte aby zistil, čo tím robí. Keďže stránka obsahuje aj čas a miesto pre denné scrumy a demo sprintu, vie tak, kde môže zistiť viac.

Keď sa sprint blíži ku koncu, Scrum master každému pripomenie nadchádzajúce demo.

```
Subjekt: Zajtra je demo Jackass tímu o 13:00 V bufete.

Ahojte všetci! Chcem Vás pozvať sa zúčastniť nášho dema sprintu
zajtra o 13:00 v bufete (piatok). Predvedieme beta release.

Viac na informačnej stránke sprintu:
http://wiki.mycompany.com/jackass/sprint15
```

So všetkým tým, nikto sa nemôže vyhovoriť, že nevedel čo sa deje.

6

Ako tvoríme backlogy sprintov

Dokázali ste to až tak ďaleko? Páni, dobrá práca.

Takže teraz, keď sme ukončili míting plánovania sprintu a oznámili sme svetu viac o našom novom nablyšťanom sprinte, je čas pre scrum mastra vytvoriť backlog sprintu. Je potrebné to urobiť *po* mítingu plánovania sprintu, ale *pred* prvým denným scrumom.

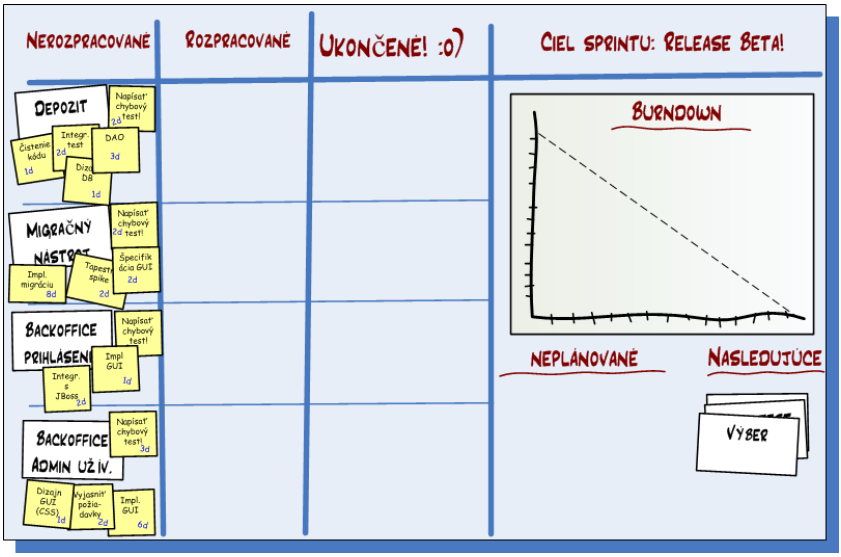
Formát backlogu sprintu

Experimentovali sme s rôznymi formátmi pre backlog sprintu vrátane Jira, Excel a fyzickej tabule úloh na stene. Na začiatku sme najviac používali Excel. Existuje mnoho voľne dostupných Excelovských šablón pre backlogy sprintov, vrátane automatickej tvorby burn-down grafov a podobných vecí. Mohol by som veľa rozprávať o úprave našich backlogov sprintu založených na Exceli. Ale nebudem. Nechcem tu uviesť ani príklad.

Namiesto toho popíšem detailne čo sme našli ako najefektívnejší formát pre backlog sprintu – tabuľu úloh umiestnenú na stene!

Nájdite veľkú nepoužitú stenu alebo obsahujúcu nepotrebné veci ako napr. logo spoločnosti, staré diagramy alebo škaredé kresby. Vyčistite stenu (spýtajte sa na povolenie iba ak musíte). Zlepte dohromady veľké kusy papierov (najmenej 2x2 metre alebo 3x2 metre pre veľké tímy).

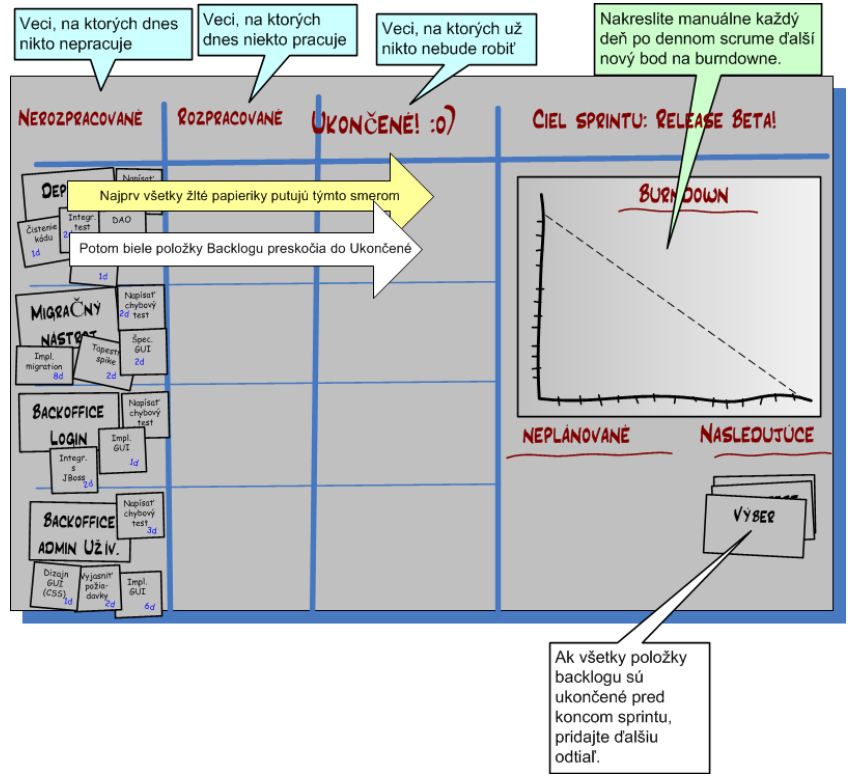
Potom urobte toto:



Samozrejme môžete použiť aj tabuľu. Je to ale mrhanie času. Ak je to možné, ponechajte si tabule pre kresby dizajnu a použite iné tabule ako tabule úloh.

Poznámka – ak používate post-ity pre úlohy, nezabudnite ich prilepiť skutočnou páskou, inak ich nájdete jedného dňa na podlahe.

Ako funguje tabuľa úloh

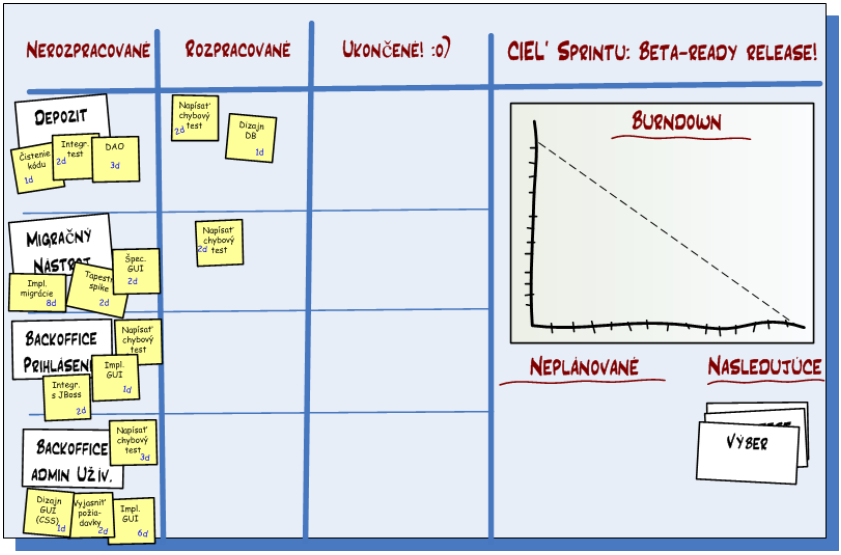


Samozrejme môžete použiť aj iné dodatočné stĺpce. „Čaká na integračné testy“ alebo „Zrušené“. Ale pred tým, než skomplikujete stav, hlboko sa zamyslite. Je to skutočne, *skutočne*, potrebné?

Zistil som, že táto jednoduchosť je extrémne hodnotná takže ja dodatočne komplikujem iba ak náklady *neurobenia* sú veľmi veľké.

Príklad 1 – po prvom dennom scrume

Po prvom dennom scrume môže tabuľa úloh vyzerať podobne:

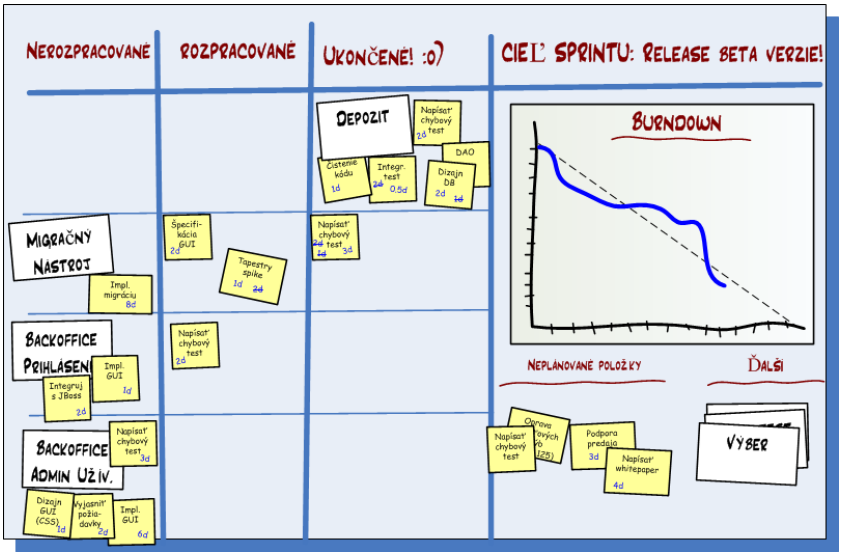


Ako vidíte, tri úlohy boli „rozpracované“, teda tím bude dnes pracovať na týchto položkách.

Niekedy pre veľké tímy úloha ostane visieť v stave „rozpracované“ pretože si nikto nepamätá, kto na nej pracoval. Keď sa to v tíme stáva často, obyčajne predstavia postupy ako označiť všetky rozpracované úlohy s menom osoby kto ju rozpracoval.

Príklad 2 – po niekoľkých dňoch

O niekoľko dní neskôr môže tabuľa vyzerať takto:



Ako môžete vidieť, že sme ukončili story „depozit“ (teda, že bola zapísaná do repozitára zdrojových kódov, otestovaná, refaktorovaná atď). Nástroj pre migráciu je čiastočne ukončený, prihlásenie k back office je naštartované a administrácia užívateľov nie je naštartované.

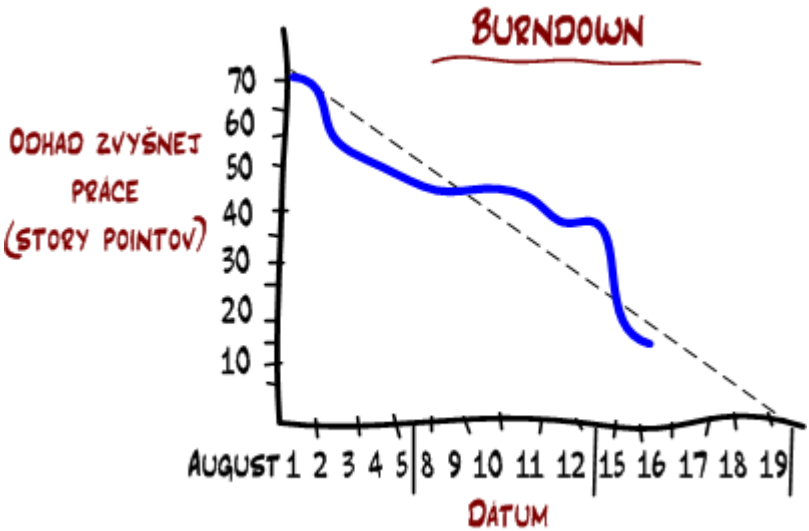
Ako môžete vidieť vpravo dole, máme 4 neplánované položky. Je užitočné si to zapamätať ak robíte retrospektívu sprintu.

Tu je príklad skutočného backlogu sprintu blízko ukončenia sprintu. Počas priebehu sprintu bude trochu neprehľadný, ale to je v poriadku, keďže má len krátku životnosť. Každý nový sprint vytvoríme nový, prázdny backlog sprintu.


NOT CHECKED OUT
CHECKED OUT
UNPLANNED ITEMS

Ako funguje burndown graf

Zamerajme sa na burndown graf:



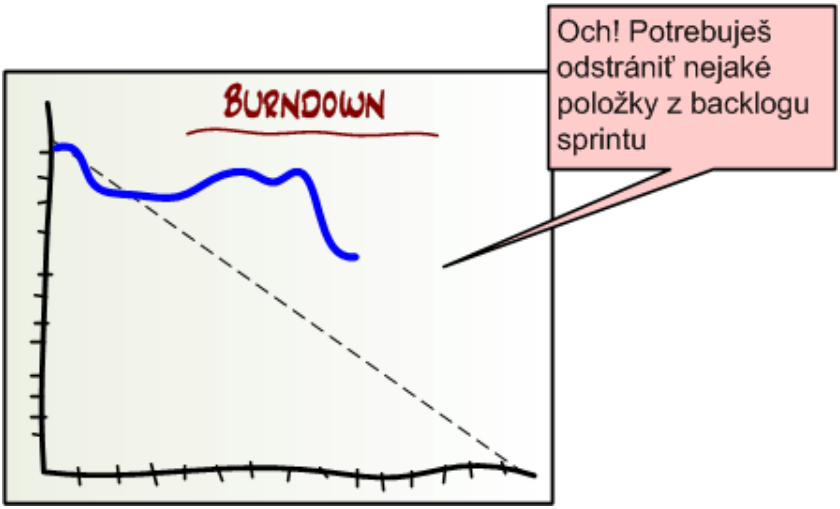
Tento graf ukazuje:

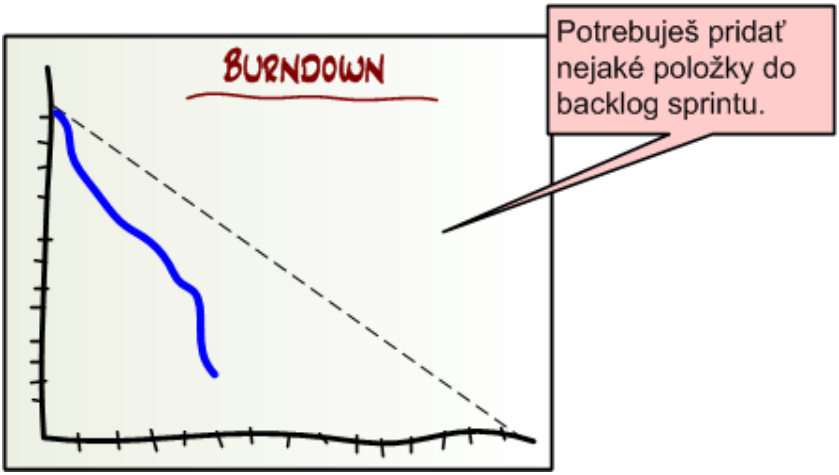
- Prvý deň sprintu, 1. Augusta, tím predpokladal, že ostáva pracovať na 70 story pointoch. Bola to *predpokladaná* rýchlosť pre celý sprint.
- 16. Augusta tím predpokladá, že ostáva približne 16 story pointov. Prerušovaná čiara trendu ukazuje, že sú približne na správnej ceste, teda že všetko ukončia do konca sprintu.

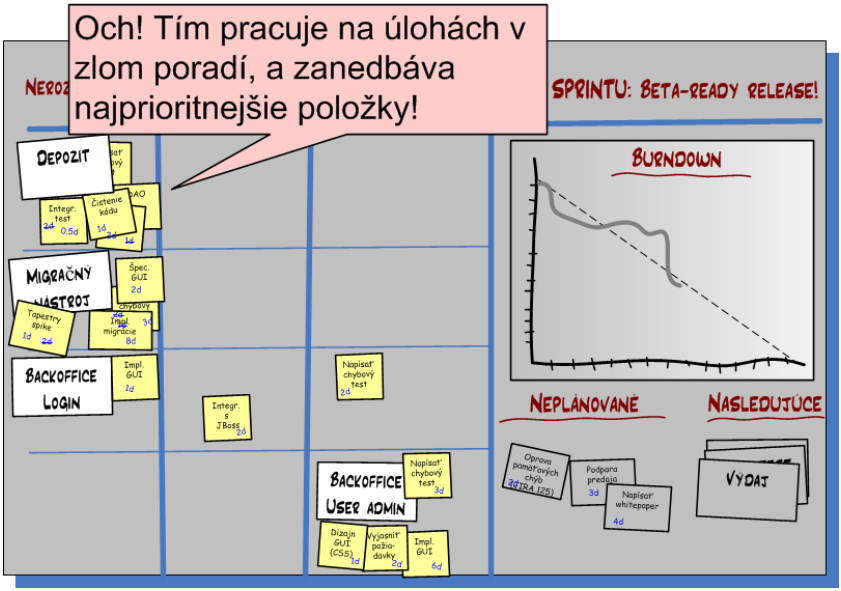
Na osi X preskakujeme víkendy, keďže práca je len málokedy robená cez víkendy. Kreslili sme aj víkendy, ale toto spravilo burndown trochu mätúcim, keďže ho to urobí „plochejším“ cez víkendy, čo môže vyzerať ako varovné znamenie.

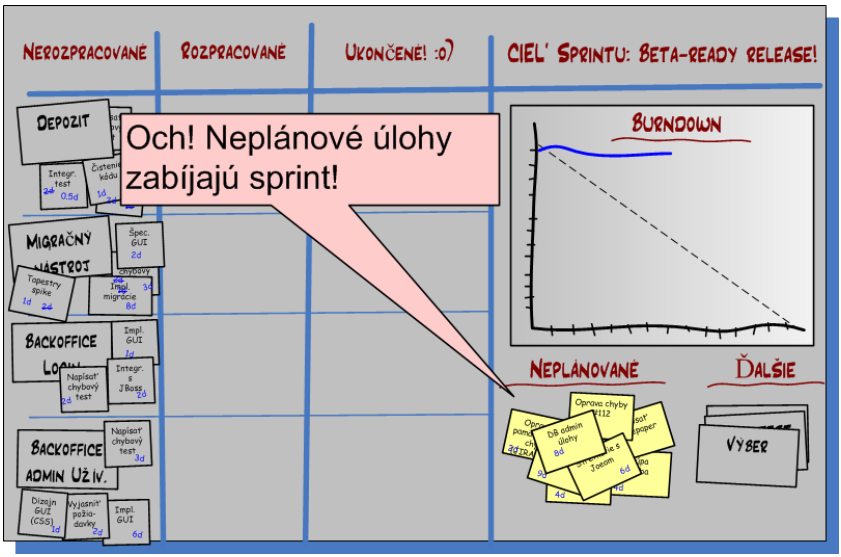
Varovné znamenie tabule úloh

Letmý pohľad na tabuľu úloh by mal poskytnúť hocikomu indikáciu ako dobre postupuje sprint. Scrum master je zodpovedný za uistenie sa, že tím sa zachová na základe varovných signálov ako napríklad:






Och! Tím pracuje na úlohách v zlom poradí, a zanedbáva najprioritnejšie položky!
SPRINTU: BETA-READY RELEASE!
DEPOZIT
Integr. test 0.5d
Čistenie kódu
MIGRAČNÝ NÁSTROJ
Špec. GUI 2d
Tapestry spike 1d
Impl. migrácie 8d
BACKOFFICE LOGIN
Impl. GUI 1d
Integr. s JBoss
Napísať chybový test 2d
BACKOFFICE USER ADMIN
Napísať chybový test 3d
Dizajn GUI (CSS) 1d
Vyjasniť požiadavky 2d
Impl. GUI 6d
BURNDOWN
NEPLÁNOVANÉ
NASLEDUJÚCE
Podpora predaja 3d
Napísať whitepaper 4d
VÝDAJ


NEROZPRACOVANÉ
ROZPRACOVANÉ
UKONČENÉ! :o)
CIEĽ SPRINTU: BETA-READY RELEASE!
Och! Neplánové úlohy zabíjajú sprint!
DEPOZIT
Integr. test 0.5d
MIGRAČNÝ NÁSTROJ
Špec. GUI 2d
Tapestry spike 1d
Impl. migrácie 8d
BACKOFFICE LOGIN
Impl. GUI 1d
Napísať chybový test 2d
Integr. s JBoss 2d
BACKOFFICE ADMIN UŽ IV.
Napísať chybový test 3d
Dizajn GUI (CSS) 1d
Vyjasniť požiadavky 2d
Impl. GUI 6d
BURNDOWN
NEPLÁNOVANÉ
ĎALŠIE
Oprava chyby 112
VÝBER

Hej, a čo sledovateľnosť?!

Najlepšiu sledovateľnosť, ktorú môžem ponúknuť v tomto modeli, je sfotiť tabuľu úloh každý deň. Ak musíte. Niekedy som to robil, ale nikdy som nepotreboval vytiahnuť tieto fotky.

Ak je sledovateľnosť pre vás dôležitá, potom možno tabuľa úloh nie je pre vás riešením.

Ale odporúčam vám skutočne sa pokúsiť odhadnúť aktuálnu hodnotu detailnej sledovateľnosti sprintu. Keď už je sprint hotový a fungujúci kód bol doručený a dokumentácia uložená, zaoberá sa niekto skutočne koľko stories bolo hotových v piatom dni sprintu? Skutočne sa niekto zaujíma aký časový odhad bol pre úlohu „napísať chybný test pre Depoziť“?

Odhadovanie dní verzus hodín

Vo väčšine kníh a článkov o Scrume zistíte, že úlohy sú časovo odhadované v hodinách, nie dňoch. Zvykli sme to takto robiť. Náš všeobecný vzorec bol: 1 efektívny človekodeň = 6 efektívnych človekohodín.

Teraz sme to prestali robiť, prinajmenšom vo väčšine našich tímov z nasledujúcich dôvodov:

- Odhady v človekohodinách boli veľmi jemné, viedlo to k podpore veľmi malých 1-2 hodinových úloh a následnému mikromanažmentu.
- Ukázalo sa, že každý premýšľal aj tak v človekodňoch a potom ich vynásobil 6 pred zapísaním človekohodín. „Hmmm, táto úloha bude trvať asi deň. Oh, musím zapísať človekohodiny. Teda napíšem 6.“
- Dve rôzne jednotky sa pletú. „Bol tento odhad v človekodňoch alebo človekohodinách?“

Takže teraz používame človekodni ako základ pre všetky časové odhady (aj keď ich voláme story pointy). Naša najnižšia hodnota je 0,5, teda každá hodnota, ktorá je

menšia ako 0,5 je buď odstránená, skombinovaná s nejakou inou úlohou alebo sa ponechá s odhadom 0,5 (žiadne veľké poškodenie s ľahkým nadhodnotením).

Pekné a jednoduché.

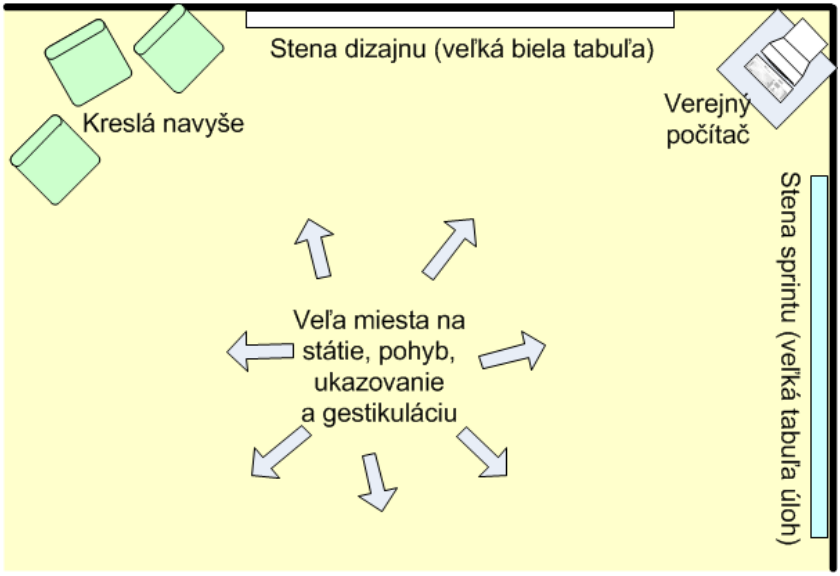
7

Ako usporiadať tímovú miestnosť

Roh dizajnu

Všimol som si, že väčšina z najzaujímavejších a najhodnotnejších diskusií o dizajne vznikli spontánne pred tabuľou úloh.

Z tohto dôvodu sa pokúšame pripraviť tento priestor ako explicitný „roh dizajnu“.



Toto je skutočne celkom šikovné. Neexistuje žiadna lepšia cesta ako získať prehľad o systéme ako stáť v rohu dizajnu a vidieť obe steny, potom sa pozrieť na počítač a skúsiť poslednú verziu systému (ak ste dosť šťastný a máte kontinuálny build, pozri str. 107 „Ako kombinujeme Scrum s XP“).

„Stena dizajnu“ je iba veľká biela tabuľa obsahujúca najdôležitejšie kresby dizajnu a výstupy najdôležitejšej dokumentácie dizajnu (sekvenčný diagram, GUI prototypy, doménové modely atď).

Hore: denný scrum prebieha vo vyššie uvedenom rohu.

Hmmm.... ten burndown vyzerá podozrivo pekne a priamo. Ale tím trval na tom, že je skutočný :o)

Nechajte tím sedieť spolu!

Keď príde na sedenie a rozloženie stolov je jedna vec, ktorá musí byť veľmi zdôraznená.

Nechajte tím sedieť spolu!

Kvôli objasnenie čo hovorím:

Nechajte tím sedieť spolu!

Ľudia sú neochotní sa presúvať. Prinajmenšom na miestach, kde som pracoval. Nechcú vyberať všetky tie veci, odpájať počítač, presúvať všetky ich rárohy na nový stôl a zapájať všetko odznova, Čím kratšia vzdialenosť, tým väčšia neochota. „Ale šéf, aký to má zmysel posunúť sa iba 5 metrov?“

Nuž, keď budujete efektívne Scrum tímy, neexistuje žiadna iná alternatíva. Dajte tím dokopy. Aj keby ste museli osobne vyhrážať každému, preniesť celé ich náčinie a vyčistiť ich staré kávové škvrny. Ak nie je miesto pre tím, urobte ho. Niekde. Aj keby ste museli umiestniť tím do suterénu. Presuňte stoly, podplaťte manažéra, robte čo treba. Ale dajte tím dokopy.

Keď dáte tím dokopy, odmena bude okamžitá. Už po prvom sprinte sa tím zhodne, že to bol dobrý nápad sa dať dokopy (taká je moja osobná skúsenosť, neznamená to, že váš tím bude tvrdohlavý to potvrdiť).

Čo „spolu“ znamená? Ako majú byť uložené stoly? Nuž, nemám žiaden podstatný názor na optimálne rozloženie stolov. A ak by som aj mal, predpokladám, že väčšina tímov nemá ten luxus rozhodovať ako presne budú ich stoly rozložené. Obvykle sú fyzické obmedzenia – susedný tím, dvere toalety, veľký automat v strede miestnosti, čokoľvek.

„Spolu“ znamená:

- **Počuteľnosť:** Hocikto v tíme môže hovoriť k hocikomu inému bez kriku alebo opustenia stola.

- **Viditeľnosť:** Každý v tíme vidí každého. Všetci vidia tabuľu úloh. Nie je potrebné byť tak blízko, aby sa dala *čítať*, ale prinajmenšom ju *vidieť*.
- **Izolácia:** Ak celý váš tím náhle vstal a spontánne sa angažuje v živej diskusii o dizajne, nie je nikto mimo tímu blízko dosť nato, aby bol rušený. A naopak.

„Izolácia“ neznamená, že tím je úplne izolovaný. V prostredí s kójami bude stačiť ak váš tím bude mať vlastnú kóju a dosť vysoké steny kóje pre odfiltrovanie *väčšiny* hluku od mimo tímových elementov.

A čo ak máte distribuovaný tím? Potom nemáte šťastie. Použite toľko technickej pomoci koľko môžete, aby ste mohli minimalizovať škody – videokonferencie, web kamery, nástroje pre zdieľanie plochy, atď.

Držte produktového vlastníka v zálive

Produktový vlastník musí byť blízko dosť nato, aby tím mohol k nemu zablúdiť a spýtať sa ho niečo a on môže zablúdiť k tabuli úloh. Ale nemal by sedieť s tímom. Prečo? Pretože sa môže stať, že nebude schopný sa zastaviť pred zasahovaním do detailov a tím sa „nezlepí“ správne (teda, že dosiahne tesný, samokontrolovateľný, hyperproduktívny stav).

Budem čestný, je to špekulácia. V skutočnosti som nevidel prípad, kedy by produktový vlastník sedel s tímom, takže nemám žiaden empirický dôvod povedať, že to je zlý nápad. Iba dobrý pocit a to, čo som počul od iných scrum mastrov.

Držte manažérov a trénerov v zálive

Je pre mňa ťažké písať o tom, pretože som aj manažér aj tréner...

Bolo mojou prácou pracovať s tímom tak blízko ako to len bolo možné. Vytvoril som tímy, presúval som sa medzi nimi, programoval v pároch s ľuďmi, trénoval scrum mastrov, organizoval mítingy plánovania sprintov atď. Spätne si väčšina ľudí myslí, že to bola Dobrá Vec, pretože mám skúsenosť s agilným vývojom softvéru.

Ale potom som bol tiež (hraje hudba Darth Vader) šéf vývoja, prakticky manažérska rola. Čo znamená, že vstupom do tímu sa to stane automaticky menej samoriaditeľné. „Dočerta, šéf je tu, pravdepodobne má veľa nápadov čo by sme mali robiť a kto má čo robiť. Nechám ho rozprávať."

Môj názor je takýto. Ak ste Scrum tréner (a pravdepodobne aj manažér) buďte zainteresovaný čo najviac. Ale iba na limitovaný čas, potom odíďte a nechajte tím sa zlepiť a riadiť sa samostatne. Skontrolujte tím raz za čas (nie často) zúčastnením sa počas dema sprintov a prezeraním sa na tabuľu úloh a počúvaním v ranných scrumoch. Ak vidíte priestor na zlepšenie, zoberte scrum mastra stranou a poučte ho. *Nie* pred tímom. Ďalší dobrý nápad je zúčastniť sa retrospektív (pozri str. 79 Ako robíme retrospektívy sprintov) ak váš tím vám verí dosť nato, že vaša prítomnosť ich neumlčí.

Pre podporu dobre fungujúcich Scrum tímov sa uistite, že majú všetko čo potrebujú, potom sa vzdiaľte ďaleko od ich cesty (okrem dema sprintov).

8

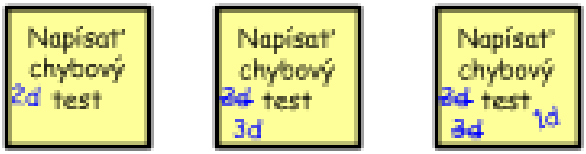
Ako robíme denné scrumy

Naše denné scrumy sú ako z knižky. Začínajú sa presne načas, každý deň na rovnakom mieste. Na začiatku sme išli robiť plánovanie sprintu do samostatnej miestnosti (tam sú tie dni, keď sme používali elektronické backlogy sprintov), ale teraz robíme denný scrum v tímovej miestnosti priamo pred tabuľou úloh. Nič nemôže byť lepšie.

Bežne počas mítingov stojíme, pretože to znižuje riziko prekročenia 15 minút.

Ako aktualizujeme tabuľu úloh

Tabuľu úloh bežne aktualizujeme počas denného scrumu. Zatiaľ čo každá osoba popisuje čo robil včera a bude robiť dnes, presúva karty po tabuli úloh. Keď popisuje neplánovanú položku, vytvorí pre túto položku kartu. Keď aktualizuje časový odhad, na kartu napíše nový odhad a preškrtne starý. Niekedy činnosti s kartami robí Scrum master zatiaľ čo ľudia rozprávajú.



Niektoré tímy majú politiku, že každá osoba by mala zaktualizovať tabuľu úloh *pred* každým mítingom.Toto tiež funguje. Len sa rozhodnite pre politiku a dodržujte ju.

Bez ohľadu na formát vášho backlogu sprintu, pokúste sa zainteresovať *celý tím* do udržania aktuálneho backlogu sprintu. Skúšali sme robiť sprint, keď Scrum master bol jediný udržiavateľ backlogu sprintu a musel obísť každý deň ľudí a spýtať sa ich na odhady zostávajúceho času.

Nevýhody sú:

- Scrum master trávi príliš veľa času administráciou namiesto podpory tímu a odstraňovania prekážok.
- Členovia tímu nie sú oboznámení o stave sprintu, keďže backlog sprintu nie je niečo o čo sa majú starať. Tento nedostatok spätnej väzby redukuje celkovú agilitu a centrum záujmu tímu.

Aktualizácia backlogu sprintu každým užívateľom by mal byť jednoduchý, ak je backlog sprintu navrhnutý správne.

Okamžite po dennom scrum mítingu niekto zosumarizuje všetky odhady časov (samozrejme pri tom ignoruje stĺpce „hotovo“) a nakreslí nový bod na burndown sprintu.

Riešenie oneskorencov

Niektoré tímy majú plechovku mincí a účtov. Keď ste sa oneskorili, aj keď len jednu minútu, pridáte do plechovky pevnú čiastku. Nepadnú nijaké otázky. Aj keď zavoláte pred mítingom, že prídete neskôr, aj tak musíte zaplatiť.

Môžete sa tomu vyhnúť iba ak máte dobré ospravedlnenie, ako napr. lekárske ošetrenie alebo vaša vlastná svadba alebo niečo podobné.

Peniaze z plechovky sú použité na spoločenské udalosti. Napr. na nákup hamburgerov keď máme noci hier :o)

Toto funguje dobre. Ale je to potrebné iba pre tímy kde ľudia často meškajú. Niektoré tímy nepotrebujú tento typ schémy.

Riešenie „Neviem čo mám dnes robiť“

Nie je zriedkavé niekoho počuť „Včera som robil bla bla bla, ale dnes nemám ani hmlistú predstavu čo robiť“ Čo teraz?

Povedzme, že Joe a Lisa sú tí, ktorí nevedia čo dnes robiť.

Ak som Scrum master chcem sa iba pohnúť ďalej a nechať rozprávať ďalšieho, ale urobte si poznámku, ktorí ľudia nemajú čo robiť. Potom čo všetci odpovedali, prejdeme s celým tímom tabuľu úloh, zhora dole a skontrolujeme či je všetko zladené, či všetci vedia čo každá položka znamená, atď. Vyzývam ľudí, aby pridali viac kariet. Potom sa vrátim späť k tým ľuďom, ktorí nevedeli čo robiť „teraz keď sme prešli cez tabuľu úloh, máte nejaký návrh o tom, čo budete dnes robiť“? Dúfam, že to budú vedieť.

Ak nie, ja uvažujem, či tu nie je nejaká možnosť párového programovanie. Povedzme že Niklas ide dnes implementovať GUI pre back office administrátora. V tomto prípade zdvorilo navrhnem, že možno Joe alebo Lisa by mohli na tom programovať v páre s Niklasom. A to zvyčajne funguje.

A ak to nefunguje, tu je nasledujúci trik.

Scrum master: „Ok, kto nám chce predviesť pripravený beta release?“ (predpokladám, že to bol cieľ sprintu)

Tím: zmätené ticho

Scrum master: „Nie sme hotoví?“

Tím: „hmmm, ... nie“

Scrum master: „Oh, do čerta. Prečo nie? Čo nám ešte ostáva?“

Tím: „Nuž nemáme ani testovací server, na ktorom by to bežalo a build skript je chybný“

Scrum master: „Aha.(pridal dve kartičky na tabuľu. Joe a Lisa, ako nám s tým môžete dnes pomôcť?“

Joe: „Uhm... tuším, že skúsim tu nájsť nejaký testovací server“.

Lisa: „...a ja sa pokúsim opraviť ten buildovací skript“.

Ak máte šťastie, niekto skutočne predvedie pripravený beta release, ktorý ste chceli. Paráda! Dosiahli ste Váš cieľ sprintu. Ale čo ak ste uprostred sprintu? Kľud. Zagratulujte tímu k dobrej práci, stiahnite jednu alebo dve ďalšie stories zo sekcie „Nasledujúce“ v pravom dolnom rohu tabule úloh a presuňte ich to stĺpca „nerozpracované“ naľavo. Potom zopakujte denný scrum. Oznámte produktovému vlastníkovi, že ste pridali nejaké položky do sprintu.

Ale čo ak tím ešte nedosiahol cieľ a Joe a Lisa stále odmietajú prísť s niečím rozumným. Obyčajne zvažujem jednu z nasledujúcich stratégií (ani jedna z nich nie je veľmi pekná, ale je to posledná možnosť):

- **Hanba:** „Nuž ak nemáte žiaden nápad ako pomôcť tímu, odporúčam Vám ísť domov alebo čítať knihu alebo čokoľvek. Alebo si iba seďte niekde tu pokiaľ niekto vás nezavolá pomôcť.“.
- **Stará škola:** Jednoducho priraďte úlohu.
- **Tlak okolia:** Povedzte „nechajte si čas Joe a Lisa, my všetci tu iba postojíme pokiaľ neprídete s niečím čo nám pomôže dosiahnuť cieľ.“
- **Nevoľníctvo:** Povedzte: „Nuž môžete dnes pomôcť tímu nepriamo ak budete sluhami. Noste kávu, dajte ľuďom masáž, vysypte nejaké koše, uvarte niečo na obed alebo urobte čokoľvek budeme počas dňa chcieť.“ Môžete byť prekvapený ako rýchlo Joe a Lisa prídu s použiteľnými technickými úlohami :o)

Ak vás nejaká osoba často núti zájsť tak ďaleko, potom by ste mali zobrať túto osobu stranou a dať jej vážne školenie. Ak problém naďalej pretrváva, potrebujete prehodnotiť, či táto osoba je alebo nie je dôležitá pre tím.

Ak *nie* je veľmi dôležitá, pokúste sa ju vybrať z tímu.

Ak *je* dôležitá, skúste ho spojiť s niekým iným, kto môže vystupovať ako „sprievodca“. Joe môže byť dobrý vývojár a architekt, iba v skutočnosti preferuje iných ľudí, ktorím by povedal čo robí. Dobre. Dajte Niklasovi povinnosť byť permanentným sprievodcom. Alebo sa o neho postarajte sami. Ak je Joe dosť dôležitý pre váš tím, stojí to za pokus. Mali sme podobné prípady a viac menej to fungovalo.

9

Ako robíme demá sprintov

Demo sprintu (alebo recenzia sprintu ako to niektorí ľudia volajú) je dôležitá časť Scrumu, ktorú majú ľudia tendenciu podceňovať.

„Ó, skutočne *musíme* robiť demo? Nie je veľmi čo ukazovať!“

„Nemáme čas na prípravu &%$# dema!“

„Nemám čas sa zúčastniť dema iných tímov!“

Prečo naliehame, aby všetky sprinty končili demom

Správne urobené demo sprintu, aj keď to vyzerá nedramaticky, má hlboký efekt:

- Tím získa česť za ich výkon. *Cítia* sa dobre.
- Iní ľudia sa učia čo tím robí.
- Demo priťahuje životne dôležitú spätnú väzbu od zákazníkov.
- Demá sú (alebo by mali byť) spoločenskou udalosťou, kde rôzne tímy majú priestor na interakciu a možnosť diskutovať o ich práci. Je to hodnotné.
- Robenie dema núti tím *ukončiť veci* a uvoľniť ich (aj keď to je len testovacie prostredie). Bez dema sme stále mali kopy na 99% dokončených vecí. S demom máme možno menej dokončených vecí, ale tieto položky sú *skutočne dokončené*, čo je

(v našom prípade) oveľa lepšie než mať kopu vecí, ktoré sú iba *nejako ukončené* a znečisťujú nasledujúci sprint.

Ak tím je viac alebo menej nútený robiť demo sprintu, aj keď im nedochádza, že skutočne funguje, demo bude trápne. Tím bude koktať a robiť chyby počas dema a potlesk na konci bude len z polovice srdca. Ľudia budú súcitiť s tímom, niektorí budú nahnevaní, že premárnili čas týmto mizerným demom.

To bolí. Ale efekt je ako horko chutnajúci liek. *Nasledujúci sprint* tím sa skutočne pokúsi *dokončiť* veci! Budú cítiť „nuž, v nasledujúcom sprinte možno môžeme predviesť iba 2 veci namiesto 5, ale tentoraz to bude FUNGOVAŤ!“. Tím vie, že budú musieť robiť demo za každých okolností, čo značne zvýši šancu, že bude čo predvádzať. Videl som to stať sa niekoľkokrát.

Kontrolný zoznam dema sprintu

- Uistite sa, že ste jasne prezentovali cieľ sprintu. Ak na deme sú ľudia, ktorí nevedia nič o produkte, popíšte ho počas niekoľkých minút.
- Nestrávte veľa času prípravou dema, hlavne nie na okázalých prezentáciách. Odstráňte nezmysly a sústreďte sa na demonštráciu aktuálneho funkčného kódu.
- Udržte si vysoké tempo, sústreďte vašu prípravu radšej na rýchle ako nádherné demo.
- Smerujte demo orientované na obchod, vynechajte technické detaily. Sústreďte sa radšej na „čo sme urobili“ než „ako sme to urobili“.
- Ak je to možné, nechajte poslucháčov si vyskúšať produkt.
- Nepredvádzajte množstvo menších opráv chýb a triviálnych vlastností. Spomeňte ich, ale nepredvádzajte ich, pretože to trvá dlho a odvádza pozornosť od dôležitejších stories.

Vyrovnanie sa s „nedemonštrovateľnými“ vecami

Člen tímu: „Nebudem demonštrovať túto položku, pretože sa to nedá. Story je ‘Zlepšiť škálovateľnosť tak, aby systém mohol spracovať 10 000 simultánnych užívateľov’. Nemôžem preboha pozvať 10 000 užívateľov, aby sme to ukázali alebo môžem?“

Scrum master: „A ukončil si to?“.

Člen tímu. „Samozrejme, že áno“.

Scrum master: „Ako to vieš?“

Člen tímu: „Nastavil som systém v prostredí testovania výkonnosti, naštartoval som 8 load serverov a trápil som systém so simultánnymi požiadavkami“.

Scrum master: „Ale máš akúkoľvek indikáciu toho, že systém zvládne 10 000 užívateľov“.

Člen tímu: „Áno. Testovacie stroje sú úbohé, teraz môžu spracovať 50 000 simultánnych požiadaviek počas môjho testu.“

Scrum master: „Ako to vieš?“

Člen tímu (frustrovaný): „Nuž mám tento report! Sám to vidíš, ukazuje ako bol test nastavený a koľko požiadaviek bolo odoslaných!“

Scrum master: „Ó nádhera! Potom toto je tvoje „demo“. Ukáž iba tento report a prejdi ho s poslucháčmi. Lepšie ako nič, že?“.

Člen tímu: „A je to dosť? Ale treba to ešte dotiahnuť“.

Scrum master: „Ok, ale netráv na tom veľa času. Nemusí to byť pekné, iba informatívne“.

10

Ako robíme retrospektívy sprintov

Prečo trváme na tom, aby všetky tímy robili retrospektívu

Najdôležitejším na retrospektívach je, aby *sa robili*.

Z nejakého dôvodu tímy nie vždy inklinujú k retrospektívam. Bez nenásilného pobádania väčšina tímov často preskočí retrospektívu a namiesto toho pôjde na ďalší sprint. Možno je to kultúrna záležitosť tu vo Švédsku, ale nie som si istý.

Ale, vyzerá to tak, že všetci sa zhodujú na tom, že retrospektíva je neobyčajne užitočná. V skutočnosti by som povedal, že retrospektíva je druhá najužitočnejšia udalosť v Scrume (prvou je míting plánovania sprintu), pretože to je vaša *najlepšia šanca sa zlepšiť*!

Samozrejme, že nepotrebujete prísť na dobré nápady na mítingu retrospektívy, to môžete aj doma vo vani! Ale bude tím akceptovať váš nápad? Možno, ale pravdepodobnosť prijatia tímom je oveľa vyššia keď nápad vznikne „z tímu“, teda že vznikne počas retrospektívy keď všetkým je dovolené prispieť a diskutovať o nápadoch.

Bez retrospektívy zistíte, že tímy robia stále tie isté chyby znova a znova.

Ako organizujeme retrospektívy

Vo všeobecnosti sa formát mení, ale obyčajne ju robíme takto:

- Vyhradíme 1-3 hodiny v závislosti na tom, koľko diskusie očakávame.
- Účastníci: produktový vlastník, celý tím a ja.
- Presunieme sa do uzavretej miestnosti, s príjemnou sedačkou v rohu, na terasu alebo podobné miesto. Kde nebudeme rušení počas diskusie.
- Zvyčajne retrospektívu nerobíme v tímovej miestnosti, pretože pozornosť ľudí zvykne ochabovať.
- Niekto je ustanovený zapisovateľom.
- Scrum master ukáže backlog sprintu a s pomocou tímu zosumarizuje sprint. Dôležité udalosti a rozhodnutia, atď.
- Urobíme „kolá“. Každá osoba dostane šancu povedať, bez prerušenia, čo si myslia, že bolo dobré, čo by mohlo byť lepšie a čo by radšej robili inak v nasledujúcom sprinte.
- Skontrolujeme predpokladanú rýchlosť voči aktuálnej. Ak je tu veľký rozdiel, pokúsime sa analyzovať prečo.
- Keď čas už skoro vypršal, Scrum master sa pokúsi zosumarizovať konkrétne návrhy čo môžeme urobiť lepšie v nasledujúcom sprinte.

Naše retrospektívy vo všeobecnosti nie sú veľmi štruktúrované. Téma na pozadí je stále rovnaká: „čo môžeme urobiť lepšie v nasledujúcom sprinte“.

Tu je príklad tabule z poslednej retrospektívy:

Tri stĺpce:

- **Dobré***:* Ak by sme mohli zopakovať sprint rovnako, tieto veci by sme robili rovnako
- **Mohlo by byť lepšie***:* Ak by sme mohli zopakovať sprint rovnako, tieto veci by sme robili inak.
- **Návrhy***:* Konkrétne nápady ako sa môžeme zlepšiť v budúcnosti.

Takže kým stĺpce 1 a 2 pozerajú do minulosti, stĺpec 3 sa pozerá do budúcnosti.

Potom čo tím prebral brainstormingom všetky tieto kartičky, použili „bodové hlasovanie“ pre rozpoznanie zlepšení, na ktoré je potreba sa v nasledujúcom sprinte zamerať. Každý člen tímu dostal 3 magnety a bol pozvaný na hlasovanie za zlepšenia, ktoré by bol rád aby ich tím preferoval počas nasledujúceho sprintu. Každý člen tímu môže rozdeliť magnety ako chce, dokonca aj umiestniť všetky na jeden problém.

Na základe toho vyberú 5 zlepšení procesu, na ktoré sa zamerajú a ktoré preveria počas nasledujúcej retrospektívy.

Je dôležité to neprehnať. Zamerajte sa len na niekoľko zlepšení na sprint.

Zdieľanie naučených lekcií medzi tímami

Informácie, ktoré sa objavia počas retrospektívy sprintu sú zvyčajne veľmi hodnotné. Prežíva tento tím ťažké časy pretože ich obchodný manažér stále núti programátorov zúčastniť sa na obchodných stretnutiach ako „technických expertov“? Toto je dôležitá informácia. Možno iné tímy mali rovnaký problém. Mali by sme náš manažment viac poučiť o našich produktoch tak, aby mohli sami robiť podporu obchodu?

Retrospektíva sprintu nie je len o tom, ako tím môže robiť lepšiu robotu počas nasledujúceho sprintu, ale má širšie dopady než len to.

Naša stratégia je veľmi jednoduchá. Jedna osoba (v tomto prípade ja) sa zúčastňuje všetkých retrospektív a zdieľam vedomostí. Úplne jednoducho.

Alternatívou môže byť, aby každý Scrum tím publikoval správu z retrospektívy sprintu. Vyskúšali sme to, ale prišli sme na to, že len málo ľudí číta takéto správy a dokonca niektorí sú proti nim. Takže sme to urobili jednoduchším spôsobom.

Dôležité pravidlá pre osobu zdieľajúcu znalosti:

- Mal by byť dobrý poslucháč.
- Ak je počas retrospektívy panuje ticho, mal by byť pripravený sa spýtať jednoduché, ale správne zamerané otázky, ktoré by simulovali diskusiu v skupine. Napr. „ak by ste mali vrátiť čas a znova urobiť rovnaký sprint 1 odo dňa 1, čo by ste urobili inak?“
- Musí byť ochotný stráviť čas návštevami všetkých retrospektív vo všetkých tímoch.
- Musí byť nejakým spôsobom autorita, takže môže vystupovať s návrhmi zlepšení, ktoré sú mimo kontrolu tímu

Toto celkom funguje, ale môžu existovať aj iné prístupy, ktoré fungujú oveľa lepšie. V tomto prípade ma poučte.

Zmeniť alebo nezmeniť

Povedzme, že tím vyvodil „v tíme komunikujeme veľmi málo, takže si stúpame po prstoch na nohách a špiníme sa dizajnom iných“.

Čo máme s tým robiť? Zaviesť denné dizajn mítingy? Vyhľadať nové nástroje pre ľahšiu komunikáciu? Pridať viac wiki stránok? Nuž, možno. Ale možno nie.

Zistili sme, že v mnohých prípadoch je už len jasná identifikácia problému dostatočná na automatické vyriešenie v nasledujúcom sprinte. Hlavne keď zapíšete retrospektívu sprintu na stenu tímovej miestnosti (čo stále zabudneme urobiť, naša hanba!). Každá zmena, ktorú zavediete má nejakým spôsobom svoju cenu, takže pred zmenami sa uvážte či nerobiť nič a dúfať, že problém zmizne (alebo sa zmenší) samovoľne.

Vyššie uvedený príklad („v tíme komunikujeme veľmi málo...“) je typickým príkladom niečoho, kde najlepším spôsobom je nerobiť vôbec nič.

Ak budete zavádzať zmeny zakaždým, keď niekto sa na niečo sťažuje, ľudia sa budú zdráhať objavovať malé problémové oblasti, čo môže byť desivé.

Príklad toho, čo sa môže objaviť počas retrospektívy

Tu sú typické príklady toho, čo sa objaví počas plánovania sprintu a typické akcie.

„Mali by sme viac času stráviť rozdelením stories na položky a úlohy“

Toto je typické. Každý deň počas denného scrumu členovia tímu sami zistia, že hovoria „Skutočne neviem, čo mám dnes robiť“. Takže po každom dennom scrume strávite čas

hľadaním konkrétnych úloh. Obyčajne je efektívnejšie to urobiť dopredu.

Typické akcie: žiadne. Tím sa s týmto vysporiada sám počas nasledujúceho plánovania sprintu. Ak sa to stáva opakovane, zväčšite čas pre plánovanie sprintu.

„Príliš veľa vonkajších rušení“

Typické akcie:

• Požiadajte tím, aby redukoval svoj faktor pozornosti tak, aby ich plán bol reálnejší.

• Požiadajte tím, aby si zaevidoval rušenia pred nasledujúcim sprintom. Kto ich rušil, ako dlho to trvalo. Bude tak jednoduchšie vyriešiť problém neskôr.

• Požiadajte tím, aby všetky rušenia poslali k scrum mastrovi alebo produktovému vlastníkovi.

• Požiadajte tím, aby označil jednu osobu ako „ brankára“, nasmerujte všetky rušenia k nemu, takže zvyšok tímu sa môže sústrediť. Môže to byť Scrum master alebo rotujúca pozícia.

„Nadmerne sme sa zaviazali a iba polovica vecí je hotová“

Typické akcie: žiadne. Tím pravdepodobne sa v nasledujúcom sprinte nadmerne nezaviaže. Alebo nie až tak zle.

„Naše prostredie v kancelárii je veľmi hlučné a zapratané“

Typické akcie:

- pokúste sa vytvoriť lepšie prostredie alebo presuňte tím mimo. Prenajmite si hotelovú izbu. Čokoľvek. (Pozri str. 66 „Ako usporiadať tímovú miestnosť“).
- Ak to nie je možné, povedzte tímu, aby nasledujúci sprint znížil svoj faktor pozornosti a jasne špecifikujte, že je to kvôli hluku a zapratanému prostrediu. Dúfajte, že toto prinúti produktového

vlastníka a kvôli tomu začať obťažovať vyšší manažment.

Našťastie som nikdy nebol ohrozený presunúť tím mimo. Ale urobím to, ak budem musieť :o)

11

Čas medzi sprintmi

V skutočnom živote nemôžete stále šprintovať. Potrebujete medzi sprintami oddychovať. Ak stále šprintujete, v skutočnosti iba beháte.

Podobné je to aj v Scrume a vo vývoji softvéru vo všeobecnosti. Sprinty su dosť intenzívne. Ako vývojár nikdy skutočne nechcete lenivieť, každý deň musíte stáť v tej pokašlanej miestnosti a povedať každému čo ste včera dosiahli. Niekoľko bude inklinovať vysloviť „Strávil som väčšinu dňa s nohami pod stolom prezeraním blogov a usrkávaním kapučína“.

Navyše k aktuálnej téme samotnej, je tu ďalší dobrý dôvod mať odpočinok medzi sprintmi. Po deme sprintu a retrospektíve aj tím aj produktový vlastník budú plní informácií a nápadov na spracovanie. Ak sa hneď rozbehnú a začnú plánovať nasledujúci sprint, je šanca, že nikto nebude mať šancu na spracovanie akejkoľvek informácie alebo sa nepoučí, produktový vlastník nebude mať čas nastaviť priority po deme sprintu atď.

Zle:

Pondelok

09-10: Sprint 1 demo
10-11: Sprint 1 retrospektíva

13-16: Sprint 2 plánovanie

Pokúšame sa zaviesť istým spôsobom medzeru pred odštartovaním nového sprintu (presnejšie, periódu *po*

retrospektíve sprintu a *pred* nasledujúcim mítingom plánovania sprintu). Ale nie vždy sme uspeli.

A naposledy, snažíme sa, aby sa retrospektíva sprintu a nasledujúci míting plánovania sprintu nevyskytovali v rovnaký deň. Každý musí mať najmenej dobrú bezsprintovú noc pred odštartovaním nasledujúceho sprintu.

Lepšie:

Pondelok	Utorok
09-10: Sprint 1 demo 10-11: Sprint 1 retrospektíva	9-13: Sprint 2 plánovanie

Ešte lepšie:

Piatok	Sobota	Nedeľa	Pondelok
09-10: Sprint 1 demo 10-11: Sprint 1 retrospektíva			9-13: Sprint 2 plánovanie

Jedným spôsobom ako to urobiť sú „lab dni“ (alebo akokoľvek ich nazývate). Teda dni keď sa vývojári môžu zaoberať čímkoľvek chcú (OK, súhlasím, inšpirované Google). Napríklad čítaním o posledných nástrojoch a API, štúdiom na certifikáciu, diskutovaním o hlúpostiach s kolegami, programovaním hoby projektov atď.

Našim cieľom je mať lab dni medzi každým sprintom. Týmto spôsobom získate prirodzený odpočinok medzi sprintmi a budete mať tím, ktorý dostáva realistickú možnosť si udržiavať aktuálne vedomosti. Navyše je to pekný a atraktívny zamestnanecký benefit.

Najlepšie?

Štvrtok	Piatok	Sobota	Nedeľa	Pondelok
09-10: Sprint 1 demo 10-11: Sprint 1 retrospektíva	LAB DAY			9-13: Sprint 2 plánovanie

V súčasnosti mávame lab dni raz za mesiac. Presnejšie prvý piatok každý mesiac. Prečo nie medzi sprintmi? Nuž, pretože som cítil, že je dôležité, aby celá spoločnosť mala lab dni v rovnakom čase. Inak to ľudia nebudú brať vážne. A keďže my (zatiaľ) nemáme sprinty cez všetky produkty, musel som vybrať namiesto toho nezávislý interval lab dní.

Možno jedného dňa budeme musieť synchronizovať sprinty vo všetkých produktoch (teda rovnaký začiatok a koniec sprintu pre všetky produkty a tímy). V tomto prípade určite dáme lab dni medzi každý sprint.

12

Ako plánujeme release a kontrakty s pevnou cenou

Niekedy potrebujeme naplánovať dopredu viac než jeden sprint súčasne. Typicky v spojení s pevnou cenou kontraktu, kde *musíme* plánovať dopredu alebo riskovať podpis niečoho, čo nemôžeme odovzdať načas.

Typické plánovanie releasu je pre nás pokusom zodpovedať otázku „*kedy, najneskôr,* budeme schopní odovzdať verziu 1.0 tohto nového systému“.

Ak sa *skutočne* chcete naučiť plánovanie releasu, odporúčam vám preskočiť túto kapitolu a namiesto toho si kúpiť knihu od Mike Cohna „Agile Estimating and Planning“. Skutočne by som si prial prečítať túto knihu skôr (prečítal som si ju *až* keď sme prišli na tieto veci sami...). Moja verzia plánovania releasu je tak trochu zjednodušená, ale môže slúžiť ako dobrý štartovací bod.

Definujte hranice akceptácie

Rozšírením produktového backlogu je, že produktový vlastník definuje zoznam *akceptačných hraníc*, ktoré jednoducho klasifikujú čo znamenajú úrovne dôležitosti v produktovom backlogu v pojmoch kontraktu.

Tu je príklad pravidiel akceptačných hraníc:

- Všetky položky s dôležitosťou >= 100 *musia* byť zahrnuté do verzie 1.0 alebo inak sme odsúdený na smrť.

- Všetky položky s dôležitosťou 50 - 99 *by mali byť* zahrnuté do verzie 1.0, ale *môžeme* byť schopní pokračovať s prácou na nich v rýchlom nasledujúcom release.
- Položky s dôležitosťou 25 - 49 sú požadované, ale môžu byť urobené v nasledujúcom release 1.1.
- Položky s dôležitosťou < 25 sú špekuláciou a nemusia byť nikdy potrebné.

A tu je príklad produktového backlogu, vyfarbeného podľa vyššie uvedených pravidiel..

Dôležitosť	Názov
130	Banán
120	Jablko
115	Pomaranč
110	Guáve
100	Hruška
95	Hrozno
80	Orech
70	Lieskovec
60	Cibuľa
40	Grapefruit
35	Papája
10	Čučoriedka
10	Broskyňa

Červená = *musí* byť zahrnutá vo verzii 1.0 (banán – hruška)

Žltá = *mala by* byť zahrnutá vo verzii 1.0 (hrozno– cibuľa)

Zelená = môže byť ukončená neskôr (grapefruit – broskyňa)

Takže, keď dodáme všetko od banánu po cibuľu načas, sme v bezpečí. Ak sa čas kráti, *môžeme* pokračovať preskočením hrozna, orechu, lieskovca alebo cibule. Všetko pod cibuľou je bonus.

Odhadujte čas najdôležitejších položiek

Aby sa dalo urobiť plánovanie release, produktový vlastník potrebuje odhady, prinajmenšom pre stories, ktoré sú zahrnuté v kontrakte. Podobne ako pri plánovaní sprintu, je to spoločné úsilie medzi produktovým vlastníkom a tímom – tím odhaduje, produktový vlastník popisuje položky a odpovedá na otázky.

Odhad času je hodnotný ak sa blíži k správnym hodnotám, menej hodnotný ak je mimo, povedzme že o faktor 30%, a úplne nanič keď nemá žiadne spojenie s realitou.

Tu je môj príspevok k hodnote odhadu času v spojení s tým, kto ho ráta a ako veľa času strávil na jeho príprave.

Všetko čo sme povedali bolo, že:

- Nechajte *tím* robiť odhady.
- Nenechajte ich to robiť dlho.
- Uistite sa, že chápe, že odhady času je *hrubým odhadom*, nie *záväzok*.

Zvyčajne produktový vlastník dostane celý tím do miestnosti, uvedie nejaké novinky a povie im, že cieľom tohto mítingu je urobiť odhad času pre najdôležitejších 20 (alebo akýchkoľvek) stories v produktovom backlogu. Prejde každú story a potom nechá tím pracovať. Produktový vlastník ostane v miestnosti, aby odpovedal na otázky a objasnil rozsah každej položky podľa potreby. Podobne ako pri plánovaní sprintu, položka „ako robiť demo“ je veľmi užitočným spôsobom ako znížiť riziko nedorozumenia.

Tento míting musí byť striktne časovo ohraničený, inak má tím tendenciu stráviť príliš veľa času odhadovaním len niekoľkých stories.

Ak produktový vlastník chce na tom stráviť viac času, môže jednoducho naplánovať ďalší míting. Tím sa musí uistiť, že dopad týchto stretnutí na ich aktuálne sprinty je jasne viditeľný pre produktového vlastníka, takže rozumie, že ich práca na odhadovaní času nie je zadarmo.

Tu je príklad ako môže skončiť odhadovanie času (v story pointoch):

Dôležitosť	Názov	Odhad
130	Banán	12
120	Jablko	9
115	Pomaranč	20
110	Guave	8
100	Hruška	20
95	Hrozno	12
80	Orech	10
70	Lieskovec	8
60	Cibuľa	10
40	Grepfruit	14
35	Papája	4
10	Borovnica	
10	Broskyňa	

Odhad rýchlosti

Ok, takže máme hrubý časový odhad pre najdôležitejšie stories.

Nasledujúcim krokom je odhadnúť našu priemernú rýchlosť za sprint.

To znamená, že potrebujeme rozhodnúť hodnotu faktoru pozornosti. Pozrite sa na stranu 28 „Ako sa tím rozhoduje, ktorú story zahrnúť do sprintu“.

Faktor pozornosti je v princípe „koľko času tímu je stráveného na práve potvrdených stories“. Nikdy to nie je 100%, pretože tímy strácajú čas robením neplánovaných

položiek, prepínajú svoj kontext, pomáhajú iným tímom, čítajú emaily, opravujú pokazený počítač, argumentujú v kuchyni o politike atď.

Povedzme, že rozhodneme, že faktor pozornosti pre tím bude 50% (celkom málo, normálne máme okolo 70%). A povedzme, že dĺžka nášho sprintu bude 3 týždne (15 dní) a veľkosť nášho tímu je 6.

Teda každý sprint je dlhý 90 človekodní, ale dá sa očakávať úplných a hodnotných 45 človekodní (kvôli nášmu 50% faktoru pozornosti).

Takže naša predpokladaná rýchlosť je 45 story pointov.

Ak každá story ma odhad času 5 dní (čo nemajú), potom tento tím môže roztočiť približne 9 story pointov za sprint.

Zložte to dokopy do plánu releasu

Teraz keď máme odhady časov a rýchlosť (45), môžeme ľahko rozdeliť produktový backlog do sprintov:

Dôležitosť	Názov	Odhad
		Sprint 1
130	Banán	12
120	Jablko	9
115	Pomaranč	20
		Sprint 2
110	Guave	8
100	Hruška	20
95	Hrozno	12
		Sprint 3
80	Orech	10
70	Lieskovec	8

60	Cibuľa	10
40	Grep	14

Sprint 4

35	Papája	4
10	Čučoriedka	
10	Broskyňa	

Každý sprint zahŕňa toľko stories, koľko je to možné bez prekročenia odhadovanej rýchlosti 45.

Teraz vidíme, že pravdepodobne budeme potrebovať 3 sprinty pre dokončenie „musíme mať“ a „mali by sme mať“.

3 sprinty = 9 kalendárnych týždňov = 2 kalendárne mesiace. Je to koncový termín, ktorý sme sľúbili zákazníkovi? To závisí od povahy kontraktu, ako je zafixovaný rozsah atď. Obyčajne pridávame značný priestor ako ochranu pred zlými časovým odhadmi, neočakávanými problémami, nepredpokladanými vlastnosťami atď. Takže v tomto prípade sa môžeme dohodnúť na dátume dodania o 3 mesiace, s 1 mesiacom „rezervy“.

Pekné je, že môžeme demonštrovať niečo použiteľné zákazníkovi každé 3 mesiace a pozvať ho zmeniť požiadavky zatiaľ čo pracujeme (samozrejme záleží na tom ako kontrakt vyzerá).

Prispôsobenie plánu releasov

Realita sa neprispôsobí plánu, takže sa musí obísť.

Po každom sprinte sa pozrieme na aktuálnu rýchlosť daného sprintu. Ak sa aktuálna rýchlosť veľmi líšila od očakávanej rýchlosti, zrevidujeme pre budúce sprinty predpokladanú rýchlosť a upravíme plán releasov. Ak nám to spôsobí problémy, produktový vlastník môže sa začať dohadovať so zákazníkom alebo začneme kontrolovať ako

redukovať rozsah bez porušenia kontraktu. A možno on a tím prídu s nejakým spôsobom ako zvýšiť rýchlosť alebo faktor pozornosti odstránením závažných prekážok, ktoré boli identifikované počas sprintu.

Produktový vlastník môže zavolať zákazníka a povedať „ahoj, sme trochu pozadu voči plánu, ale verím, že stihneme koncový termín ak odstránime vlastnosť „integrovaný Pacman“, ktorá zaberie veľa času. Môžeme ho pridať do nasledujúceho releasu 3 týždne po prvom release, ak chcete“.

Možno to nie je dobrá správa pre zákazníka, ale prinajmenšom sme čestní a dávame zákazníkovi možnosť – máme dodať najdôležitejšie veci načas alebo dodáme všetko neskôr. Obyčajne to nie je ťažká voľba :o)

13

Ako kombinujeme Scrum s XP

V skutočnosti to nie je kontroverzné vyhlásenie ak povieme, že Scrum a XP (Extrémne programovanie) môžu byť úspešne skombinované. Väčšina vecí, čo som videl na Internete podporujú podobné hypotézy, takže nebudem tráviť čas argumentovaním prečo.

Ale spomeniem jednu vec. Scrum sa zameriava na manažment a na praktiky organizácie zatiaľ čo XP sa zameriava väčšinou na aktuálne programovacie praktiky. To je dôvodom, prečo môžu spolu fungovať – zameriavajú sa na rôzne okruhy a navzájom sa dopĺňajú.

Týmto pridávam svoj hlas existujúcim empirickým dôkazom, že Scrum a XP môžu byť úspešne skombinované!

Zvýrazním niektoré z najhodnotnejších XP praktík a to, ako sa aplikujú v dennodennej práci. Nie všetky naše tímy sú vedené k adaptácii všetkých týchto praktík, ale experimentovali sme s väčšinou stránok kombinácie XP/Scrum. Niektoré XP praktiky sú adresované Scrumom a môže sa zdať, že sa prekrývajú, napr. „Celý Tím“, „Sedieť spolu!, Stories“ a „Plánovacia hra“- V týchto prípadoch sa jednoducho prikloníme k Scrumu.

Párové programovanie

Začali sme s ním len v poslednej dobe v jednom z našich tímov. Funguje to celkom dobre. Väčšina z našich tímov stále veľmi nerobí párové programovanie, ale skúsili sme to

v jednom tíme počas niekoľkých sprintov. Som inšpirovaný skúsiť takýto tréning vo viacerých tímoch.

Nejaké závery o párovom programovaní:

- Párové programovanie zlepšuje kvalitu kódu.
- Párové programovanie zlepšuje tímovú pozornosť (napr. keď chlapík za vami povie „hej, je táto vec skutočné potrebná pre tento sprint?“).
- Prekvapujúco mnoho vývojárov, ktorí sú silne proti párovému programovaniu ho v skutočnosti nikdy nevyskúšali. Rýchlo si ho obľúbili, keď si ho vyskúšali.
- Párové programovanie je vyčerpávajúce a nemalo by byť robené počas celého dňa.
- Častá zmena párov je dobrá.
- Párové programovanie zlepšuje šírenie vedomostí v skupine. Prekvapivo rýchlo.
- Niektorím ľuďom párové programovanie nevyhovuje. Nevyhadzujte excelentného programátora iba preto, že mu párové programovanie nevyhovuje.
- Kontrola kódu je dobrou alternatívou k párovému programovaniu.
- „Navigátor“ (chlapík, ktorý nepoužíva klávesnicu) by mal mať pre seba tiež počítač. Nie pre vývoj, ale pre pomoc, prezeranie dokumentácie keď „vodič“ (chlapík s klávesnicou) sa zasekne atď.
- Nenúťte ľudí do párového programovania. Povzbudzujte ľudí, poskytnite im správne nástroje, ale nechajte ich experimentovať.

Vývoj riadený testami (Test-driven development - TDD)

Amen! Pre mňa je to dôležitejšie než Scrum a XP dokopy, Môžete mi zobrať dom a moju televíziu, ale nezastavujte ma pre robením TDD! Ak nemáte radi TDD, potom ma nenechajte v budove, pretože sa pokúsim ho zatiahnuť či už tak alebo onak ☺.

Tu je 10 sekundový prehľad o TDD:

> *Vývoj riadený testami znamená, že napíšete automatizovaný test, potom napíšte trochu kódu tak, aby tento test prešiel, potom zrefaktorujete kód pre zvýšenie čitateľnosti a odstránenie duplikátov. Prepláchnuť a opakovať.*

Dôsledky vývoja riadeného testami:

- TDD je *ťažké*. Programátorovi to trvá nejaký čas. V skutočnosti nezáleží koľko sa ho učíte a viete a predvádzate – v mnohých prípadoch je jediným spôsobom pre programátora robiť TDD spárovať ho s niekým iným, kto už je dobrý v TDD. Ak už raz programátor sa zžije s TDD, obyčajne bude podstatne ovplyvnený a už nikdy nebude chcieť pracovať iným spôsobom.
- TDD má nesmierne pozitívny efekt na systémový dizajn.
- Trvá to nejaký čas rozbehnúť a fungovať s TDD efektívne v novom produkte, hlavne black-box integračné testy, ale návratnosť investícií je *rýchla*.
- Uistite sa, že investujete čas potrebný pre zjednodušenie písania testov. Znamená to získanie správnych nástrojov, učenie ľudí, vytváranie správnych pomocných tried ale aj bázových tried atď.

Používame nasledujúce nástroje pre vývoj riadený testami:

- jUnit / httpUnit / jWebUnit. Odporúčame TestNG and Selenium.

- HSQLDB ako embeded in-memory databázu pre testovacie účely.
- Jetty ako embedded in-memory web kontajner pre testovacie účely.
- Cobertura pre metriky pokrytia testami.
- Spring framework pre prepojenie rôznych typov testovacích príslušenstiev (s mock-mi, bez nich, s externou databázou, s pamäťovou databázou, atď).

V našich najsofistikovanejších produktoch (z perspektívy TDD) máme automatizované black-box akceptačné testy. Tieto testy štartujú celý systém v pamäti, vrátane databáz a web serverov, a pristupujú k systému iba prostredníctvom verejných rozhraní (napr. HTTP).

Toto vytvára neobyčajne rýchle cykly vývoj-build-test. Zároveň vytvára aj záchrannú sieť, poskytujúc vývojárom dostatočnú dôveru pre častý refaktoring, čo znamená, že dizajn ostáva čistý a jednoduchý napriek rastu systému.

TDD na novom kóde

Robíme TDD pre všetky naše nové projekty, aj keď to znamená dlhšiu prípravu iniciálneho projektu (keďže potrebujeme viac nástrojov a podporu pre ovládanie testov, atď.) Je to síce tak trochu neintelektuálna činnosť, ale benefity sú tak veľké, že neexistuje žiadne to nie je ospravedlnenie prečo nerobiť TDD.

TDD na starom kóde

TD je ťažké, ale skúšať robiť TDD na kóde, ktorý nebol vytváraný pomocou TDD od začiatku .... too je *skutočne ťažké*! Prečo? Nuž, mohol by som o tom napísať mnoho stránok o tejto téme, takže radšej prestanem. Nechám si to pre moju ďalšiu publikáciu „TDD zo zákopov“ :o)

Strávili sme množstvo času skúšaním automatizácie integračného testovania v jednom z našich komplexnejších systémov, kódovou základňou, ktorá už nejakú tu chvíľu existovala a bola v dosť zlom stave a kompletne chýbajúcimi testami.

Pre každý release systému máme tím vyhradených testerov, ktorí budú vykonávať celú kopu komplexných regresných a výkonnostných testov. Regresné testy boli zväčša manuálnou prácou. Toto značne spomaľovalo náš vývojový a release cyklus. Našim cieľom bolo zautomatizovanie týchto testov. Aj napriek nárazom našich hláv do steny počas niekoľkých mesiacov, nepohli sme sa ďalej.

Po tomto sme zmenili prístup. Objasnili sme si fakt, že s manuálnymi regresnými testami sme zlyhali a namiesto toho sme sa začali pýtať sa samých seba „Ako môžeme urobiť manuálne testovanie menej časovo náročné?“

Bol to systém pre hry a zistili sme, že množstvo času testovacieho tímu bolo stráveného robením celkom jednoduchých nastavovacích úloh alebo čakaním na začiatok naplánovaného turnaja. Takže sme nato vytvorili nástroje. Malé, ľahko prístupné skratky a skripty, ktoré robili všetku ofrflanú prácu a nechali testerov sa sústrediť na aktuálne testovanie.

Táto snaha sa skutočné vyplatí! V skutočnosti práve toto sme mali urobiť od samého začiatku. Boli sme hladní po automatizácii testovania, ktoré sme zabudli robiť krok za krokom, kde prvý krok bol vytvoriť vec, ktorá urobí *manuálne* testovanie efektívnejším.

Poučenie z lekcie: Ak ste skončili s tým, že musíte robiť manuálne regresné testovanie a chcete to odstrániť zautomatizovaním, nerobte to (pokiaľ to skutočne nie je jednoduché). Namiesto toho vytvorte veci, ktoré robia manuálne regresné testovanie jednoduchším. *Potom* zvážte automatizáciu aktuálneho testovania.

Inkrementálny dizajn

Znamená to udržiavať dizajn jednoduchým od začiatku a priebežne ho radšej zlepšovať než skúšať ho robiť správnym od samého začiatku a potom ho zmraziť.

Urobili sme toho dosť, teda strávili sme dostatočný čas refaktorovaním a zlepšovaním existujúceho dizajnu a iba občas sme strávili čas robením veľkých dizajnov. Samozrejme, že sme niekedy narazili, napríklad pripustením neistého dizajnu sme sa „ponorili“ veľmi

hlboko takže refaktoring sa stal veľkým projektom. Ale tak či tak sme boli spokojní.

Priebežné zlepšovanie dizajnu je väčšinou automatickým vedľajším efektom robenia TDD.

Priebežná integrácia

Väčšina našich produktov má celkom sofistikované integračné nastavenie založené na Maven alebo QuickBuild. Je to extrémne hodnotné a šetriace čas. Je to definitívne riešenie starého problému „hej, ale pracuje to na mojom stroji“. Náš server priebežného buildu figuruje ako „sudca“ alebo referenčným bodom, podľa ktorého stanovuje zdravie všetkých našich báz kódu. Zakaždým keď niekto zapíše niečo do systému správy verzií, server kontinuálneho buildu sa prebudí, vytvorí všetko od podlahy na zdieľanom serveri a spustí všetky testy. Ak sa niečo pokazí, pošle email oznamujúci celému tímu to, že build zlyhal, obsahujúc aj informácie, ktorý kód presne poškodil build, linku na testovacie reporty atď.

Každú noc kontinuálny buildovací server úplne prebuilduje produkt a publikuje binárky, dokumentáciu, testovacie reporty, reporty pokrytia testami, reporty závislostí atď. do nášho interného portálu dokumentácie. Niektoré produkty sú automaticky deployované do testovacieho prostredia.

Nastaviť to znamenalo *množstvo práce,* ale vyplatilo sa to.

Kolektívne vlastníctvo kódu

Podporujeme kolektívne vlastníctvo kódu, ale nie všetky tímy ho adaptovali. Zistili sme, že párové programovanie s častou rotáciou párov automaticky vedie k vyššej úrovni kolektívneho vlastníctva kódu. Takéto tímy dokázali, že sú veľmi robustné, napr. ich sprinty neodumrú iba preto, že nejaký kľúčový človek je chorý.

Informatívne pracovné prostredie

Všetky tímy majú prístup k tabuliam a k prázdnemu priestoru na stenách a aj to dobre využívajú. Vo väčšine miestností nájdete steny pokryté s rôznymi druhmi informácií a produkte a projekte. Najväčším problémom je starý odpad kumulujúci sa na stenách, ale možno by sme mali zaviesť rolu „upratovačka“ v každom tíme.

Podporujeme používanie tabúľ, ale nie všetky tímy to nateraz chcú. Pozri str. 66 „Ako usporiadať tímovú miestnosť“.

Štandard kódovania

V poslednej dobe sme začali definovať štandard kódovania. Je to veľmi použiteľné, prial by som si to mať skôr. Nezaberie to skoro žiaden čas, len začnite jednoducho a nechajte ho rásť. Zapíšte veci, ktoré nie sú zrejmé všetkým a odkážte na existujúce materiály kedykoľvek je to možné.

Väčšina programátorov má vlastný, rozdielny, štýl kódovania. Malé detaily ako ošetrenie výnimiek, ako komentujú kód, kedy vracajú null atď. V niektorých prípadoch rozdiely nie sú podstatné, v iných môžu viesť k vážnym nekonzistenciám systémového dizajnu a ťažko čitateľnému kódu. Štandard písania kódu je veľmi použiteľný, pokiaľ sa zamerajú na podstatné veci.

Tu je niekoľko príkladov z nášho štandardu písania kódu:

- Môžete porušiť akékoľvek pravidlá, ale uistite sa, že je na to dobrý dôvod a zdokumentujte to.
- Štandardné použite konvencie Sun kódu: http://java.sun.com/docs/codeconv/html/CodeConvTOC.doc.html
- Nikdy, nikdy nezachytávajte výnimky bez logovania stavu zásobníka ale znova vyvolania výnimky. log.debug() je v poriadku, iba nestraťte zásobník.

- Použite setter-based dependency injection pre vzájomné oddelenie tried (samozrejme okrem prípadu kedy to je to potrebné).
- Vyhnite sa skratkám. Všeobecne známe skratky ako DAO sú v poriadku.
- Metódy, ktoré vracajú kolekcie alebo polia nesmú vracať null. Namiesto null je potrebné vrátiť prázdnu kolekciu alebo prázdne pole.

Udržateľné tempo / zaktivizovanie práce

Mnoho kníh o agilnom softvérovom vývoji vyhlasuje, že zväčšenie práce nadčas je v softvérovom vývoji kontraproduktívna.

Po niekoľkých neochotných experimentoch to môžem iba odporúčať celým srdcom!

Pred rokom jeden z našich tímov (najväčší tím) pracoval nadčasy do zbláznenia. Kvalita existujúceho kódu bola bezútešná a strávili množstvo času hasením ohňa. Testovací tím (ktorý tiež pracoval v nadčasoch) nemal šancu robiť akúkoľvek serióznu kontrolu kvality. Naši užívatelia boli nazlostení.

Po niekoľkých mesiacoch sme znížili množstvo pracovných hodín na prijateľnú úroveň. Ľudia pracovali normálny počet pracovných hodín (samozrejme okrem projektových kritických okamihoch) A, prekvapujúco, produktivita a kvalita sa výrazne zlepšila.

Samozrejme, redukcia pracovných hodín nebola *jediným* aspektom vedúcim k zlepšeniu, ale sme presvedčení, že mala na ňom najväčší podiel.

14

Ako testujeme

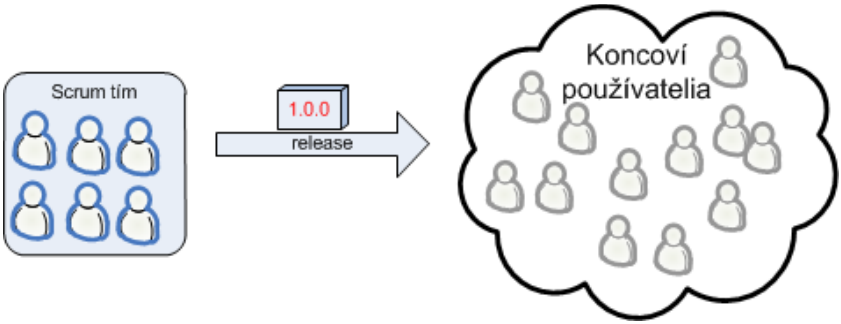
Toto je najťažšia časť. Neviem či najťažšia časť Scrumu alebo vôbec najťažšia časť softvérového vývoja vo všeobecnosti.

Testovanie je časťou, ktorá sa bude pravdepodobne líšiť vo väčšine organizácií. V závislosti na tom, koľko testerov máte, ako sú testy zautomatizované, aký typ systému máte (server a webová aplikácia alebo softvér šírený v krabici?), veľkosť release cyklov, ako je softvér kritický (blogovací server verzus systém riadenia letov), atď.

Experimentovali sme dosť so spôsobom testovania v Scrume. Skúsim popísať čo sme doteraz robili a čo sme sa naučili.

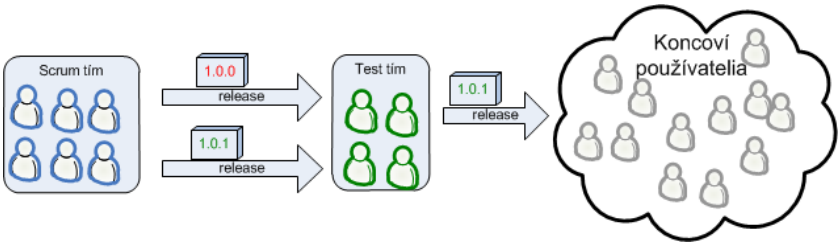
Nemôžete sa zbaviť akceptačnej testovacej fázy

V ideálnom svete Scrumu sprint končí v potenciálne deployovateľnej verzii vášho systému. Takže iba urobiť deploy, že?



Chyba.

Naša skúsenosť je, že toto nefunguje. Budú tu odporné chyby. Ak kvalita má pre vás akúkoľvek hodnotu, nejaký druh manuálnej akceptačnej fázy je potrebný. To je vtedy, keď dedikovaní testeri, ktorí *nie sú* súčasťou tímu búšia do systému s takými typmi testov, na ktoré Scrum tím nemyslel, alebo nemal čas ich robiť, alebo nemali ani hardvér, na ktorom ich robiť. Testeri pristupujú k systému presne rovnako ako koncoví užívatelia čo znamená, že musia byť urobené manuálne (predpokladajúc, že systém je určený pre ľudí).



Testovací tím nájde chyby, Scrum tím bude musieť urobiť opravné releasy a skôr alebo neskôr (radšej skôr) budete radšej schopní zverejniť opravenú verziu 1.0.1 pre koncových užívateľov ako roztrasenú verziu 1.0.0.

Keď hovorím „akceptačná testovacia fáza“ myslím tým celé obdobie testovania, ladenia a znova zverejnenia až kým verzia nie je vhodná pre produkčný release.

Minimalizácia akceptačnej testovacej fázy

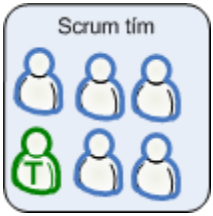
Akceptačná testovacia fáza zraňuje. Je vnímaná jasne neagilne. Aj keď sa jej nemôžeme zbaviť, môžeme sa pokúsiť ju zminimalizovať. Presnejšie, minimalizovať množstvo *času* potrebného pre akceptačnú testovaciu fázu. Môže to byť urobené:

- Maximalizujte kvalitu kódu odovzdaného Scrum tímom
- Maximalizujte efektivitu manuálnej práce počas testovania (nájdite lepších testerov, dajte im najlepšie nástroje, uistite sa, aby najviac zdržujúce úlohy mohli byť zautomatizované)

Takže ako maximalizujeme kvalitu kódu dodanú Scrum tímom? Nuž, je mnoho spôsobov. Tu sú dva, ktoré pre nás fungovali najlepšie:

- Zahrňte testerov do Scrum tímu
- Počas sprintu robte menej

Zvýšte kvalitu zahrnutím testerov do Scrum tímu



Áno, počujem obidve námietky:

- "Veď to je zrejmé! Scrum tímy musia byť multi-disciplínne!"
- "Scrum tímy musia byť bezrolové! Nemôžeme mať niekoho, kto je *iba* tester!"

Dovoľte mi vysvetliť to. Pod pojmom „tester" v tomto prípade myslím „Niekoho, koho primárnou vedomosťou je testovanie", a nie" niekto, koho rolou je robiť iba testovanie".

Vývojári sú často hrozní testeri. *Hlavne* vývojári testujúci ich vlastný kód.

Testeri sú tí, čo „podpisujú“

Testeri nie sú „iba“ členmi tímu, tester je dôležitá profesia. Je to niekto, kto podpisuje. Nič nie je uznané za „hotové“ v sprinte pokiaľ *on* nepovie, že to je hotové. Zistil som, že vývojári často povedia o niečom, že je hotové aj keď v skutočnosti to hotové nie je. Aj keď máte jasnú definíciu „hotovo“ (čo by ste skutočne mali, pozri str. 37 - Definícia „hotovo“), vývojári na ňu skoro zabudnú. My programátori sme netrpezliví ľudia a chceme prejsť na nasledujúcu položku čo najskôr.

Takže ako p. T (náš tester) vie že niečo je hotové? Nuž, najskôr musí (prekvapenie) *to otestovať*! V mnohých prípadoch zistí, že niečo vývojárom označené za „hotové“, v skutočnosti to ani nebolo *možné otestovať* !

Pretože to nebolo publikované na testovací server alebo to nebolo možné naštartovať alebo čokoľvek. Keď p. T otestoval vlastnosť, mal by prejsť cez kontrolný zoznam „hotovo“ (ak nejaký máte) s vývojárom. Napr. ak definícia „hotovo“ určuje, že má existovať správa o release, potom p. T musí overiť jej existenciu. Ak existuje nejaká formálnejšia špecifikácia vlastnosti (v našom prípade výnimočné) tak p. T ju musí overiť tiež. Atď.

Pekným momentom je, že tím má v tomto momente chlapíka schopného perfektne zorganizovať demo sprintu.

Čo robí tester keď nemá čo testovať?

Táto otázka sa neustále vynára. Pán T.:“ Hej scrum master, nemám teraz čo testovať, čo teda *mám* teraz robiť?“. Bude trvať týždeň než tím skompletizuje prvú story, takže čo má robiť tester počas *tohto* času?

Nuž, ako prvé by sa mal *pripraviť na testovanie*. To znamená napísanie špecifikácie testovania, prípravu testovacieho prostredia atď. Takže keď vývojár má už niečo pripravené na testovanie, nemusí čakať. Pán T. môže začať testovať.

Ak tím robí TDD, potom ľudia trávia čas písaním testovacieho kódu od samého začiatku, od dňa 1. Tester by mal programovať v páre s vývojármi píšucimi testovací

kód. Ak tester nemôže programovať, stále sa môže zúčastniť párového programovania s vývojármi, ale mal by navigovať vývojára počas písania. Dobrý tester prichádza s inými druhmi testov ako vývojár, takže sa môžu navzájom dopĺňať.

Ak tím nerobí TDD, alebo ak nie je dosť testovacích prípadov pre vyplnenie času testerov, mal by jednoducho robiť čokoľvek čo pomôže tímu dosiahnuť cieľ sprintu. Podobne ako ktokoľvek z tímu. Ak tester môže programovať, potom je to skvelé. Ak nie, váš tím musí identifikovať všetky neprogramátorské úlohy, ktoré by mali byť ukončené počas sprintu.

Keď spodrobňujete stories na úlohy počas plánovania sprintu, tím má tendenciu sa sústreďovať na *programátorské úlohy*. Ale obyčajne je množstvo neprogramátorských úloh, ktoré majú byť dokončené v sprinte. Ak strávite čas pokusom *identifikovať neprogramátorské úlohy* počas fázy plánovania sprintu, je tu šanca, že p. T bude schopný prispieť, aj keď nevie programovať a práve nie je čo programovať.

Príklady neprogramátorských úloh, ktoré sú často potrebné:

- Príprava testovacieho prostredia.
- Vyjasnenie požiadaviek.
- Diskusia detailov deploymentu s operations.
- Napísať deployment dokumentáciu (poznámky releasu, RFC alebo čokoľvek čo vaša organizácia robí).
- Kontaktovanie externých zdrojov (napr. GUI dizajnérov).
- Zlepšenie buildovacích skriptov.
- Ďalšie spodrobnenie stories na úlohy.
- Identifikácia kľúčových otázok vývojárov a získanie odpovedí.

Na druhej strane, čo urobím ak p. T sa stane úzkym miestom? Povedzme, že máme posledný deň sprintu

a náhle niekoľko vecí je ukončených a p. T nemá šancu všetko overiť. Čo urobíme? Nuž môžeme každého z tímu urobiť asistentom p. T. On rozhodne, ktoré veci urobí sám a ktoré deleguje na zvyšok tímu. O tom je multi-disciplínny tím!

Takže áno, p. T *má* špeciálnu rolu v tíme, ale stále je schopný robiť inú prácu, a iní členovia tímu sú schopní robiť ich prácu.

Zvýšenie kvality menej robením v sprinte

Týmto sa vraciame do mítingu plánovania sprintu. Jednoducho povedané, nepchajte príliš veľa stories do sprintu! Ak máte problémy s kvalitou alebo dlhé cykly akceptačných testov, potom robte menej počas sprintu! Toto automaticky vedie k zvýšeniu kvality, kratším cyklom akceptačných testov a nadlho k vyššej produktivite pretože tím sa môže celý čas sústrediť skôr na nové veci než na opravy starých, ktoré nefungujú.

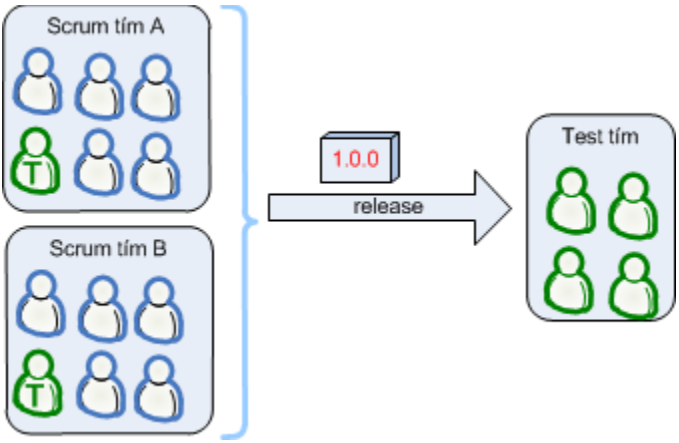
Je skoro vždy lacnejšie budovať menej, ale stabilnejšie, ako budovať veľa a potom v panike naprávať.

Má byť akceptačné testovanie súčasťou sprintu?

V tomto váhame. Niektoré naše tímy zahŕňajú akceptačné testovanie do sprintu. Väčšina tímov to však nerobí, z dvoch dôvodov:

- Sprint je časovo obmedzený. Akceptačné testovanie (použitím mojej definície, ktorá zahŕňa ladenie a opätovné releasovanie) je veľmi ťažké pre time-box. Čo ak uplynie čas a stále máte kritickú chybu? Vypustíte release do produkcie s kritickou chybou? Budete čakať na ďalší sprint? V obidvoch prípadoch je to neakceptovateľné. Preto nechávame akceptačné testovanie mimo.
- Ak máte viacero Scrum tímov pracujúcich na rovnakom produkte, manuálne akceptačné

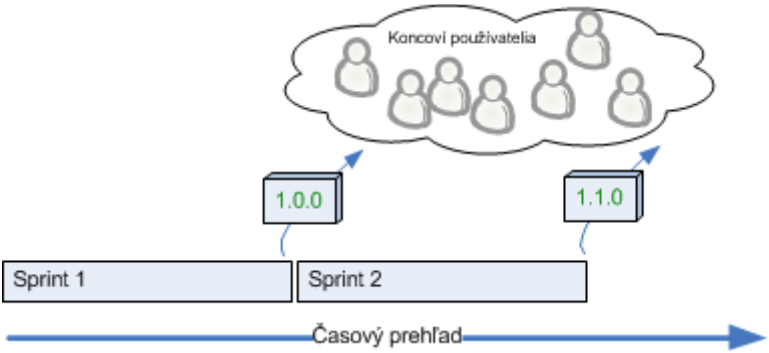
testovanie musí kombinovať výsledok práce všetkých tímov. Ak tímy robia manuálne akceptačné testy v sprinte, stále budete potrebovať tím pre otestovanie konečného releasu, ktorý je integrovaným buildom všetkých tímov.



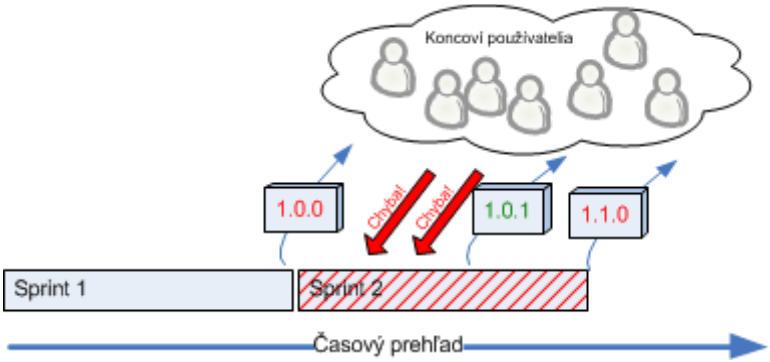
Toto nie je perfektným riešením, ale vo väčšine prípadov dosť dobré pre nás.

Cykly sprintu verzus cykly akceptačných testov

V perfektnom McScrum svete nebudete potrebovať fázy akceptačných testov, pretože každý Scrum tím uvoľňuje novú verziu pripravenú na produkciu každý sprint.



Nuž, tu je realistickejší obrázok:

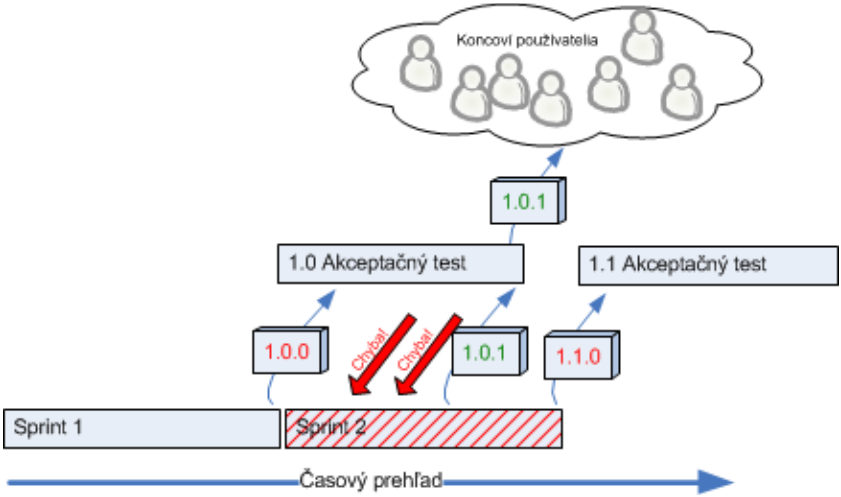


Po sprinte 1 je uvoľnená chybná verzia. Počas sprintu 2 sa začali objavovať reporty chýb, tím strávi väčšinu času ladením a je nútený urobiť release 1.0.1 uprostred sprintu. Na konci sprintu 2 uvoľnia verziu s novými vlastnosťami 1.1.0, ktorá je samozrejme oveľa chybovejšia, keďže mali menej času to urobiť správne vďaka všetkým tým prerušeniam počas posledného releasu. Atď, atď.

Diagonálne červené čiary v sprinte 2 symbolizujú chaos.

Nie je to veľmi pekné, že? Nuž, smutnou vecou je, že problém zostáva aj keď máte akceptačný testovací tím. Jediný rozdiel je, že väčšina reportov chýb bude pochádzať od testovacieho tímu a nie od nahnevaných koncových užívateľov.

To je veľký rozdiel z obchodnej perspektívy, ale pre vývojárov je to rovnaké. Aj keď testeri sú obyčajne menej agresívni ako koncoví užívatelia. Zvyčajne.



Na riešenie tohto problému sme nenašli žiadne jednoduché riešenie. Experimentovali sme s množstvom modelov.

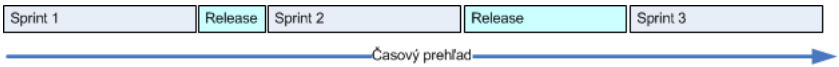
Najprv, opäť, maximalizujte kvalitu kódu, ktorú Scrum tím uvoľňuje. Cena skorého vyhľadania a opravy chýb je extrémne nízka v porovnaní s cenou neskoršieho vyhľadania a opravy chýb.

Ale faktom ostáva, aj keď môžeme minimalizovať počet chýb, stále budú chyby oznamované po ukončení sprintu. Ako sa s tým vysporiadame?

Prístup 1: „Nezačínajte nové veci pokiaľ sú staré v produkcii“

Znie to pekne, že?

Už niekoľkokrát sme boli blízko adoptovaniu tohto prístupu a nakreslili sme chutné modely ako to dosiahneme. Ale zakaždým sme si to rozmysleli, keď sme zvážili nevýhody. Museli by sme pridať časovo neohraničené release periódy medzi sprintmi, kde by sme iba testovali a ladili až by sme urobili produkčný release.



Nepáčilo sa nám mať takéto periódy, hlavne preto, že by to narušilo regulárny cyklus sprintov. Nemohli by sme

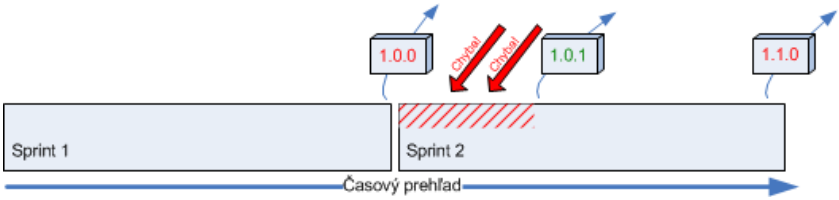
povedať, že „každé 3 týždne začíname nový sprint“. Okrem toho by to aj tak nevyriešilo problém úplne. Aj keby sme mali release periódu, aj tak budú urgentné chyby oznamované z času na čas, a my musíme byť pripravení ich riešiť.

Prístup 2: “OK pre odštartovanie tvorby nových vecí, ale uprednostňujte staré veci do produkcie”

Toto je nami preferovaný prístup. Prinajmenšom teraz.

Základom je, že ak ukončíme sprint, prejdeme na nasledujúci. Ale očakávame nejaký čas strávený opravami chýb z posledného sprintu. Ak je nasledujúci sprint príliš zničený, pretože sme strávili veľa času opravami chýb z predošlého sprintu, vyčíslime prečo sa to stalo a ako môžeme zvýšiť kvalitu. Uistime sa, že sprinty sú dosť dlhé pre prežitie dostatočného času stráveného opravami chýb predošlého sprintu.

Postupne, počas niekoľkých mesiacov, množstvo času stráveného opravami chýb z predošlých sprintov sa zníži. Navyše sme boli schopní mať menej zainteresovaných ľudí keď sa našli chyby, takže *celý* tím nemusel byť vyrušený každou chybou. Tak sme dnes na akceptovateľnejšej úrovni.



Počas mítingov plánovania sprintov nastavujeme faktor pozornosti dostatočne nízky nato, aby sme mohli očakávať opravy chýb z posledného sprintu. Časom sa tímy naučili odhadovať ho celkom presne. Veľmi nám pomohla metrika rýchlosti (viď str. 27 - Ako sa tím rozhodne, ktoré stories zahrnúť do sprintu?)

Zlý prístup– „zamerať sa na tvorbu nových vecí“

To v skutočnosti znamená „zamerajte sa na tvorbu *radšej* nových vecí *než dostať staré veci do produkcie*“. Kto by to takto chcel? Nuž, my sme robili túto chybu dosť často na začiatku a som si istý, že ani mnoho iných spoločností to robí takisto. Je to choroba zo stresu. Mnoho manažérov tomu v skutočnosti nerozumie, že aj keď je kód hotový, ste ešte ďaleko od produkčného releasu. Prinajmenšom pre komplexný systém. Takže manažér (alebo produktový vlastník) žiada od tímu pridávanie nových vecí, zatiaľ čo pozadie kódu už skoro-pripraveného-na-release sa stáva zložitejším a zložitejším, spomaľujúcim čokoľvek iné.

Nepredstihnite najpomalšiu časť vo vašej reťazi

Povedzme, že akceptačné testy sú vašim najslabším článkom. Máte veľmi málo testerov, alebo obdobie akceptačných testov je dlhá, pretože kvalita kódu je bezútešná.

Povedzme že váš tím akceptačného testovania je schopný overiť 3 vlastnosti za týždeň (nie, „počet vlastností za týždeň“ nepoužívame ako metriku). A povedzme, že vývojári sú schopní vyvinúť 6 nových vlastností za týždeň.

Bude to nútiť manažérov a produktových vlastníkov (alebo aj tím) pre načasovanie vývoja na 6 vlastností za týždeň.

Nerobte to! Realita vás dostihne a bude to bolieť.

Namiesto toho, načasujte 3 nové vlastnosti za týždeň a strávte zvyšok času zmenšením úzkych miest testovania. Napr.:

- Nechajte niekoľkých vývojárov pracovať ako testerov (óo, budú vás za to milovať...).
- Implementujte nástroje a skripty, ktoré robia testovanie ľahším.
- Pridajte viac automaticky testovaného kódu.

- Zväčšite dĺžku sprintu a zahrňte akceptačné testy do sprintu.
- Definujte niektoré sprinty ako „testovacie sprinty“ kde celý tím pracuje ako testovací tím.
- Najmite viac testerov (aj keď to znamená odstránenie vývojárov)

Vyskúšali sme všetky tieto riešenia (okrem posledného). Najlepšie dlhodobé riešenie je samozrejme bod 2 a 3.

Retrospektíva je dobré fórum pre identifikáciu najslabších článkov v reťazci.

Späť do reality

Pravdepodobne som vám poskytol pocit, že máme testerov v každom Scrum tíme, že máme veľké akceptačné testovacie tímy pre každý produkt uvoľnený každý sprint atď, atď.

Nie, nemáme.

Niekedy sme to tak robili a videli sme pozitívne efekty. Ale stále sme ďaleko od procesu poistenia akceptovateľnej kvality a ešte stále sa máme čo učiť.

15

Ako zvládame viacero Scrum tímov

Množstvo vecí je ťažších keď máte viacero Scrum tímov pracujúcich na rovnakom produkte. Tento problém je univerzálny a nemá nič spoločné so Scrumom. Viac vývojárov = viac komplikácií.

Aj s týmto sme (ako vždy) experimentovali. Navyše sme mali tím zložený z približne 40 ľudí pracujúcich na rovnakom produkte.

Kľúčové otázky sú:

- Koľko tímov vytvoriť?
- Ako alokovať ľudí do tímov?

Koľko tímov vytvoriť

Keď je práca s viacerými Scrum tímami taká ťažká, prečo sa máme vlastne trápiť? Prečo nedať jednoducho všetkých to rovnakého tímu?

Najväčší Scrum tím, ktorý sme kedy mali bolo 11 ľudí. Fungoval, ale nie veľmi dobre. Denné scrumy mali tendenciu sa naťahovať na viac ako 15 minút. Členovia tímu nevedeli čo iní v tíme robia, takže sa niekedy splietli. Pre scrum mastra bolo ťažké udržať všetkých zameraných na cieľ a bolo ťažké nájsť čas pre adresovanie všetkých reportovaných prekážok.

Alternatívou je rozdelenie na dva tímy. Ale je to lepšie? Nie nevyhnutne.

Ak tím má skúsenosť a je spokojný so Scrumom, a je logický spôsob rozdelenia roadmapy na dve rôzne cesty, a tieto dve cesty sa navzájom neovplyvňujú v rovnakom zdrojovom kóde, potom by som povedal, že je to dobrý nápad rozdeliť tím. Inak by som zostal pri jednom tíme, aj napriek nevýhode veľkých tímov.

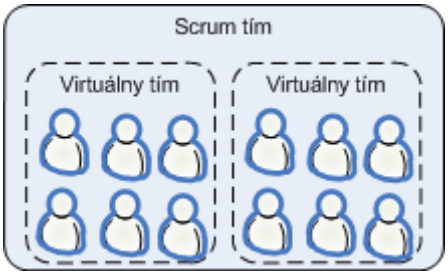
Moja skúsenosť je, že je lepšie mať menej tímov, ktoré sú dosť veľké než mať mnoho malých tímov, ktoré sa ovplyvňujú navzájom. Malé tímy urobte iba ak sa navzájom neovplyvňujú!

Virtuálne tímy

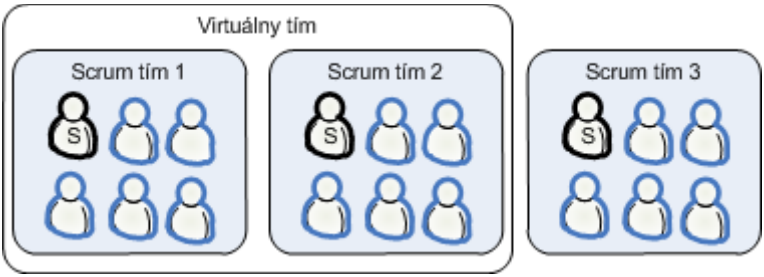
Ako viete či ste urobili správne alebo nesprávne rozhodnutie s rešpektom k zváženiu „veľký tím“ verzus „mnoho tímov“?

Ak máte oči a uši otvorené, možno si všimnete formu „virtuálnych tímov“.

Príklad 1: Zvolili ste si jeden veľký tím. Ale keď začnete sledovať kto sa s kým rozpráva počas sprintu, zistíte, že sa tím rozdelil na dva menšie tímy.



Príklad 2*:* Rozhodli ste sa mať tri menšie tímy. Ale keď začnete zisťovať kto s kým komunikuje, zistíte, že tím 1 a tím 2 medzi sebou celý čas, zatiaľ čo tím 3 pracuje v izolácii.



Čo to znamená? Že vaša stratégia rozdelenia tímov bola zlá? Áno, ak vyzerá, že virtuálne tímy sú druhom permanentnými. Nie, aj virtuálne tímy vyzerajú byť dočasnými.

Pozrite sa opäť na príklad 1. Ak dva virtuálne tímy majú tendenciu sa raz za čas meniť (teda ľudia sa presúvajú medzi tímami) potom ste pravdepodobne urobili správne rozhodnutie mať ich ako jeden Scrum tím. Ak dva tímy zostanú rovnakými počas celého sprintu, pravdepodobne ich budete chcieť v nasledujúcom sprinte rozdeliť na dva reálne scrum tímy.

Teraz sa pozrite na príklad 2. Ak tím 1 a tím 2 navzájom komunikujú (a nie s tímom 3) počas celého sprintu, pravdepodobne budete chcieť skombinovať tím 1 a tím 2 do jedného Scrum tímu v nasledujúcom sprinte. Ak tím 1 a tím 2 komunikujú navzájom počas prvej polovice sprintu a potom tím 1 a tím 3 počas druhej polovice, potom by ste mali zvážiť skombinovať do jedného tímu všetky tri tímy alebo ich nechať ako tri tímy. Spýtajte sa túto otázku počas retrospektívy sprintu a nechajte sa tímy rozhodnúť samé.

Rozdelenie tímu je jednou z najťažších častí Scrumu. Nerozmýšľajte o tom veľmi hlboko alebo príliš neoptimalizujme. Experimentujte, dávajte pozor na virtuálne tímy a uistite sa, že sa tejto téme budete venovať počas retrospektív. Skôr alebo neskôr nájdete správne riešenie pre partikulárne situácie. Dôležitou vecou je, že tímy sú spokojné a príliš o seba nezakopávajú.

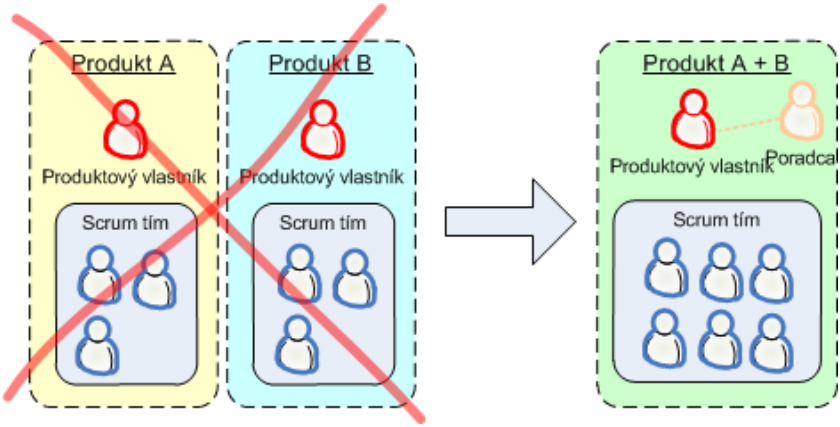
Optimálna veľkosť tímu

Väčšina kníh, ktoré som čítal tvrdia, že „optimálna“ veľkosť tímu je niekde medzi 5-9 ľuďmi.

Z toho, čo som doteraz videl, môžem iba súhlasiť. Aj keď by som povedal 3-8 ľudí. V skutočnosti verím, že stojí za nejakú tú bolesť dosiahnuť tímy tejto veľkosti.

Povedzme, že máme jeden Scrum tím 10 ľudí. Zvážte odstránenie dvoch najslabších členov tímu. Ups, práve som to povedal?

Povedzme, že máte dva rôzne produkty, s jedným tímom 3 osôb na produkt, a obidva sa hýbu veľmi pomaly. *Môže* to byť dobrý nápad skombinovať ich do jedného 6 členného tímu zodpovedného za obidva produkty. V tomto prípade nechajte jedného z dvoch produktových vlastníkov ísť (alebo mu dajte nejakú rolu poradcu alebo čokoľvek iné).

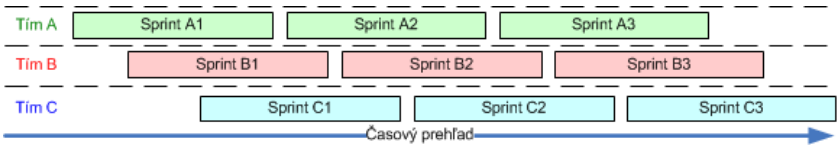


Povedzme, že máme jeden 12 členný Scrum tím, pretože kód je v takom hroznom stave, že neexistuje spôsob ako by 2 rôzne tímy mohli na ňom nezávisle pracovať. Venujte vážne úsilie pre fixovanie kódu (namiesto budovania nových funkčností) až kým sa nedostanete do bodu, keď môžete tímy rozdeliť. Táto investícia sa pravdepodobne vráti veľmi skoro.

Synchronizované sprinty – alebo nie?

Povedzme, že máme tri Scrum tímy pracujúce na rovnakom produkte, Majú byť ich sprinty synchronizované, teda odštartované a ukončené naraz? Alebo sa majú prekrývať?

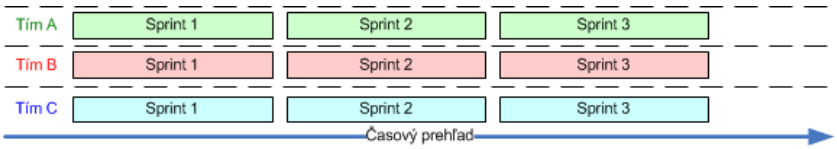
Náš prvým prístupom boli prekrývajúce sa sprinty (s ohľadom na čas).



Znelo to dobre. V akomkoľvek momente bude prebiehajúci sprint končiť a nový sprint sa začínať. Pracovné zaťaženie produktového vlastníka by sa rozložilo v čase. Release by boli distribuované v čase. Demá každý týždeň. Aleluja.

Áno, ja viem, ale skutočne to v tom čase *znelo* presvedčivo!

Práve sme s tým začali, keď som jedného dňa mal možnosť rozprávať sa s Kenom Schwaberom (v spojení s mojou Scrum certifikáciou). Podotkol, že to bol *zlý* nápad, že bude oveľa lepšie synchronizovať sprinty. Nepamätám si presný dôvod, ale po diskusii som bol presvedčený.



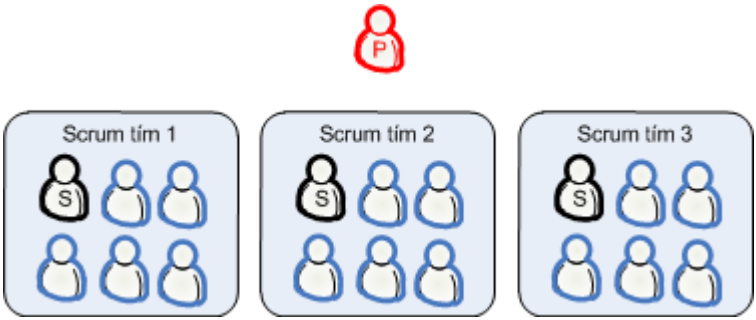
Toto je riešenie, ktoré sme odvtedy stále používali a nikdy neoľutovali. Už nikdy nebudeme vedieť, či by stratégia prekrývajúcich sa sprintov zlyhala, ale myslím si, že hej. Výhodou synchronizovaných sprintov je:

- Je tu prirodzený čas, v ktorom sa tímy dajú reanranžovať – medzi sprintmi! S prekrývajúcimi sa sprintmi neexistuje možnosť pre zmenu tímov bez prerušenia aspoň jedného tímu uprostred sprintu.

- Všetky tímy môžu pracovať na spoločnom cieli sprintu a mať plánovacie mítingy sprintov súčasne, čo vedie k lepšej spolupráci medzi tímami.
- Menej administratívneho zaťaženia, teda menej plánovacích mítingov, demo mítingov, releasov.

Prečo sme predstavili rolu "team lead"

Povedzme, že máme jeden produkt s troma tímami.



Červený chlapík označený P je Produktový vlastník. Čierni chlapíci označení S sú Scrum mastri. Zvyšok sú ~~prasce~~.... hmmm.. rešpektovaní členovia tímu.

Kto v tejto konštelácii rozhodne, kto má byť v akom tíme? Produktový vlastník? Alebo spoločne traja Scrum mastri? Alebo si každá osoba vyberie svoj tím? A čo ak všetci chcú byť v tíme 1 (pretože Scrum master 1 *tak dobre vyzerá*)?

Čo ak sa neskôr ukáže, že skutočne nie je potrebné mať viac než dva tímy pracujúce paralelne na rovnakom kóde, takže potrebujeme transformovať do dvoch tímov s 9 osobami namiesto troch 6 členných tímov. To teda znamená 2 scrum mastrov. Takže, ktorého zo súčasných 3 scrum mastrov zbavíme jeho titulu?

V mnohých spoločnostiach to bude celkom citlivý problém.

Je zvykom nechať produktového vlastníka robiť alokáciu ľudí. Ale to v skutočnosti nie sú veľmi záležitosti produktového vlastníka, že? Produktový vlastník je

doménový expert, ktorý hovorí tímu, akým smerom by mali bežať. Skutočne by nemal byť zatiahnutý do takýchto detailov. Hlavne keď je len „kura“ (ak ste niekedy počuli o metafore kurča a prasa, inak dajte si vyhľadať v Google “chickens and pigs”).

Vyriešili sme to zavedením roly „team lead“ Korešponduje to tomu, čo by ste mohli volať „Pán Scrum Scrumov“ alebo boss alebo „hlavný Scrum master“ atď. Nemusí viesť žiaden tím, ale je zodpovedný za medzitímové problémy ako napr. kto má byť Scrum mastrom tímu, ako majú byť ľudia rozdelení do tímov, atď.

Kým sme prišli na dobré meno tejto roly, mali sme ťažké časy. „Team lead“ bol najmenej odporný, ktorý sme mohli nájsť.

Toto riešenie u nás funguje a môžem ho len odporúčať (bez ohľadu nato, ako sa rozhodnete túto rolu nazývať).

Ako alokujeme ľudí do tímov

Ak máte viacero tímov na rovnakom produkte, existujú dve stratégie pre alokáciu ľudí to tímov.

- Nechajte určenú osobu urobiť alokáciu, napr. „team lead“, ktorého som spomínal vyššie, produktového vlastníka alebo funkčného manažéra (ak je natoľko zainteresovaný, že je schopný urobiť dobré rozhodnutia).
- Nechajte to nejako urobiť samotné tímy.

Experimentovali sme so všetkými troma spôsobmi. Troma? Áno. Stratégiou 1, stratégiou 2 a kombináciou oboch.

Kombinácia sa ukázala ako najlepšia.

Pred plánovacím mítingom sprintu zvolá team lead míting alokácie tímu spolu s produktovým vlastníkom a všetkými scrum mastermi. Hovoríme o poslednom sprinte a rozhodneme sa, či sú potrebné nejaké realokácie. Možno chceme skombinovať dva tímy alebo presunúť nejakých ľudí z jedného tímu do druhého. Na niečom sa zhodneme

a spíšeme to ako *navrhovanú alokáciu tímu*, ktorú prinesieme na míting plánovania sprintu.

Prvá vec, ktorú robíme počas tohto mítingu, je prechod cez najprioritnejšie položky produktového backlogu. Potom team lead povie niečo ako:

„Ahoj. Pre ďalší sprint odporúčame nasledujúce alokácie tímov."

Priedbežná alokácia tímu		
Tím 1 - tom - jerry - donald - mickey	Tím 2 - goofy - daffy - humpty - dumpty	Tím 3 - minnie - scrooge - winnie - roo

„Ako vidíte, znamená to redukciu zo 4 na 3 tímy. Vypísali sme členov pre každý tím. Zoskupte sa prosím a vyčleňte si časť steny."

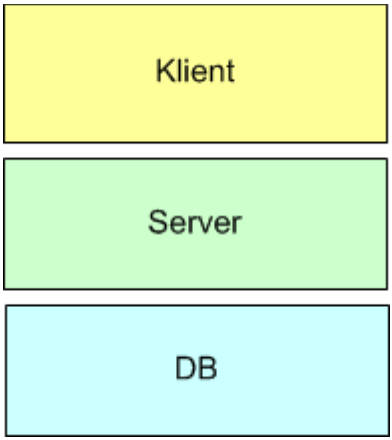
(team lead počká zatiaľ čo ľudia pochodujú po miestnosti, po chvíli sú tu 3 skupiny ľudí, každá stojaca pred prázdnou časťou steny).

„Teraz je rozdelenie tímov *dočasné*! Je to iba začiatok pre ušetrenie času. Počas vývoja progresu sprintu môžete prejsť do iného tímu, rozdeliť svoj tím na dva, spojiť sa s iným tímom alebo čokoľvek iné. Použite spoločné pocity založené na prioritách produktového vlastníka."

Toto je to čo sme našli ako najlepšie fungujúce. Určitá úroveň centralizovanej kontroly na začiatku, nasledovaná určitou úrovňou decentralizovanej optimalizácie neskôr.

Špecializované tímy – alebo nie?

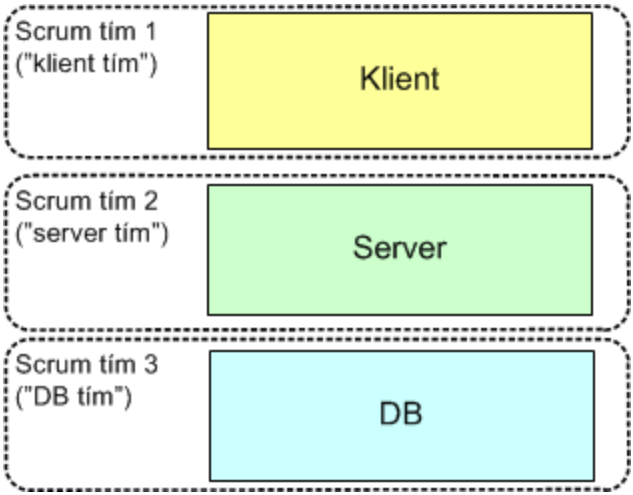
Povedzme, že vaša technológia pozostáva z troch základných komponent:



A povedzme, že máte 15 ľudí pracujúcich na tomto produkte, takže ich skutočne nechcete nechať fungovať ako jeden tím. Aké tímy vytvoríte?

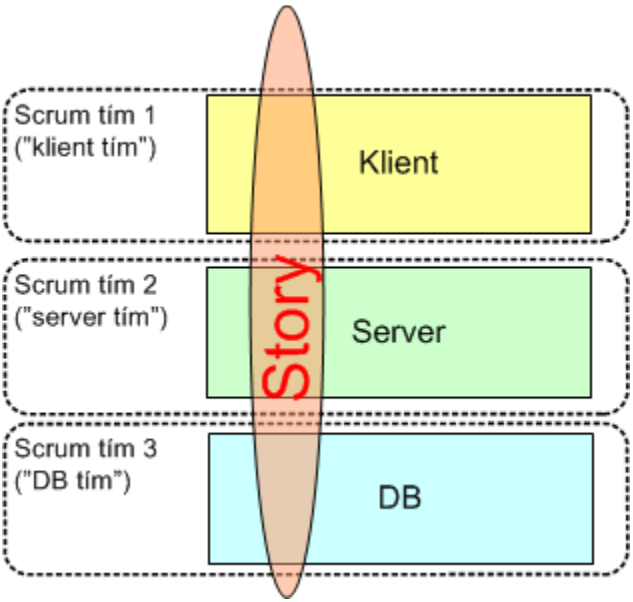
Prístup 1: Komponentovo špecializované tímy

Jedným prístupom môže byť vytvorenie komponentovo špecializovaných tímov ako napr. „klientsky tím“, „server tím“ a „DB tím“.



Takto sme začali. Nefungovalo to celkom dobre, prinajmenšom ak väčšina stories zasiahla viacero komponent.

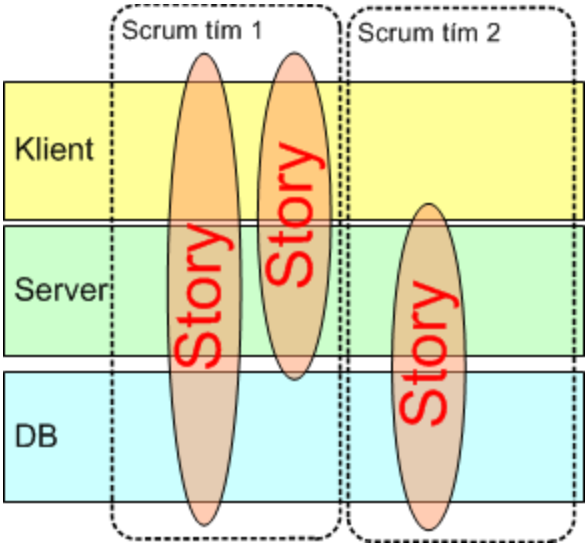
Napr. povedzme, že máme story „tabuľa, kde užívatelia môžu zasielať správy medzi sebou“. Táto vlastnosť zasiahne aktualizáciu užívateľského rozhrania v klientovi, pridanie logiky na server a pridanie nejakých tabuliek do databázy.



To znamená, že všetky tímy musia spolupracovať na tom, aby túto story ukončili. A to nie je veľmi dobré.

Prístup 2: Cez-komponentové tímy

Druhým prístupom je vytvoriť cez-komponentové tímy, teda tímy, ktoré nie sú naviazané na nijakú špecifickú komponentu.



V prípade, že nejaká z vašich stories zasahuje viacero komponent, toto rozdelenie tímov bude fungovať lepšie. Každý tím môže implementovať celú story vrátane klientskej časti, serverovej časti a databázovej časti. Tímy takto môžu pracovať nezávislejšie navzájom, čo je Dobrá Vec..

Jedna z prvých vecí, ktorú sme urobili keď sme predstavili Scrum, bola rozbitie existujúcich komponentovo orientovaných tímov (prístup 1) a namiesto toho vytvorenie cez-komponentových tímov (prístup 2). To zmenšilo počet prípadov „toto nemôžeme dokončiť lebo čakáme na chlapcov zo serverovej strany kým dokončia ich časť“.

Napriek tomu ale niekedy vytvoríme dočasné komponentovo orientované tímy keď je to silne potrebné.

Meniť tímy medzi sprintmi – alebo nie?

Sprinty sú zvyčajne dosť rôznorodé, v závislosti od typu stories, ktoré sú najprioritnejšie v danom momente. Výsledkom je, že optimálne tímy môžu byť rôzne pre rôzne sprinty.

V skutočnosti sme zistili, že skoro za každým sprintom hovoríme „*tento* sprint nebol v skutočnosti *normálnym* sprintom, pretože bla bla bla..“. Po chvíli sme jednoducho upustili od pojmu „normálny“ sprint. Neexistujú normálne sprinty. Podobne ako neexistujú „normálne“ rodiny alebo „normálni“ ľudia.

Jeden sprint je dobré mať iba klientske tímy pozostávajúce z každého kto pozná dobre bázu kódu klienta. V nasledujúcom sprinte je možno dobrým nápadom mať dva cez-komponentové tímy a rozdeliť ľudí z klientskej časti do oboch.

Jedným z kľúčových aspektov Scrumu je „tímové lepidlo“, teda ak tím pracuje spoločne počas viacerých sprintov, potom sa stane obyčajne *zomknutým*. Naučia sa ako dosiahnuť *tok skupiny* a dosiahnuť neuveriteľnú úroveň produktivity. Ale potrvá to niekoľko sprintov dostať sa na túto úroveň. Ak stále budete meniť tím, nikdy nedosiahnete túto úroveň.

Takže ak chcete preskupiť tímy, uistite sa, že ste zvážili následky. Je to krátkodobá alebo dlhodobá zmena? Ak je to krátkodobá zmena, zvážte či ju vôbec urobiť. Ak je dlhodobá, urobte to.

Jednou výnimkou je, keď prvýkrát začínate robiť Scrum s veľkým tímom. V tomto prípade pravdepodobne stojí zato experimentovať s delením tímov, až kým nenájdete to, s čím je väčšina spokojná. Uistite sa, že všetci chápu, že je v poriadku ak sa všetko na začiatku niekoľkokrát pokazí. Aspoň kým sa zlepšujete.

Part-time členovia tímu

Môžem iba potvrdiť čo sa píše v knihách o Scrume – mať členov tímu len na part-time nie je vo všeobecnosti dobrý nápad.

Povedzme, že idete vziať Joe ako part-time člena vášho Scrum tímu. Najprv si to starostlivo zvážte. Skutočne potrebujete Joa do tímu? Ste si istý, že nemôžete vziať Joe na celý čas? Aké sú jeho iné záväzky? Môže niekto iný prevziať Joeove záväzky a nechať tak Joe v pasívnejšej,

podpornej role s rešpektom k iným záväzkom? Môže sa Joe pripojiť na plný čas k tímu v *nasledujúcom* sprinte a dovtedy preniesť jeho zodpovednosti na niekoho iného?

Niekedy neexistuje cesta von. Bezpodmienečne potrebujete Joe, pretože je jediným DB v budove, ale iné tímy ho potrebujú presne rovnako, takže nikdy nebude alokovaný pre váš tím na full-time, a navyše spoločnosť nemôže najať viac DBA. Dobre. Toto je platný prípad mať ho zaradeného na part-time báze (a mimochodom je to aj to čo sa stalo nám). Ale uistite sa, že túto kontrolu urobíte zakaždým.

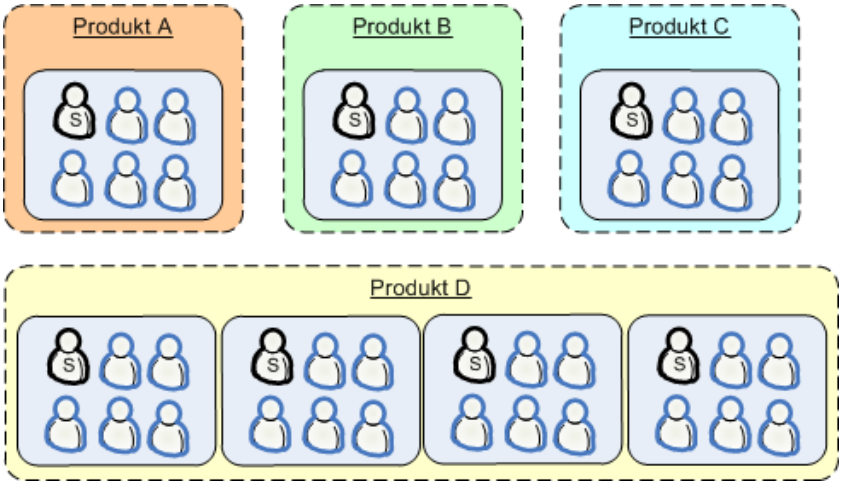
Vo všeobecnosti mám radšej tím z 3 full-timerov ako 8 part-timerov.

Ak máte niekoho, ktorý bude deliť svoj čas medzi viacero tímov, ako vyššie uvedený DBA, je dobrým nápadom ho mať primárne priradeného k jednému tímu. Zistite, ktorý tím ho najviac potrebuje a urobte tento tím jeho „domovským tímom“. Pokiaľ ho nikto iný nevytiahne, bude prítomný na tímovom dennom scrume, plánovacích mítingoch sprintu, retrospektívach atď.

Ako robíme Scrum scrumov

Scrum scrumov je jednoducho obyčajný míting, kde všetci Scrum mastri sa zídu porozprávať.

V jednom okamihu sme mali štyri produkty, kde tri produkty mali po jednom scrum tíme, zatiaľ čo posledný produkt mal 25 ľudí rozdelených na niekoľko Scrum tímov. Asi takto:



To znamená, že sme mali dve úrovne scrum scrumov. Mali sme jednu „produktovú úroveň“ scrumu scrumov pozostávajúcu zo všetkých tímov ProduktuD a jednu „korporátnu úroveň“ pozostávajúcu zo všetkých produktov.

Produktová úroveň Scrumu scrumov

Tento míting je veľmi dôležitý. Robíme ho raz za týždeň, niekedy aj častejšie. Diskutujeme o integračných problémoch, balansujeme tímy, pripravujeme sa na nasledujúce mítingy plánovania sprintov atď. Máme alokovaných 30 minút, ale častejšie to preženieme. Alternatívou by mohlo mať Scrum Scrumov každý deň, ale toto sme nikdy neskúšali.

Naša agenda pre Scrum Scrumov je:

1) okolo stola, každý popíše čo jeho tím dosiahol za posledný týždeň, čo chcú urobiť tento týždeň a aké problémy majú.

2) Akékoľvek medzi tímové problémy, ktoré musia byť vyriešené, napr. integračné problémy.

Pre mňa osobne agenda scrumu scrumov nie je veľmi dôležitá, dôležitou je *mať* mítingy scrum scrumov pravidelne.

Korporátna úroveň scrumu scrumov

Túto úroveň voláme „Pulz“. Robili sme tento míting rôznymi formami s rôznymi účastníkmi. Neskôr sme opustili tento koncept a nahradili ho týždennými stretnutiami všetkých zainteresovaných vo vývoji. 15 minút.

Čo? 15 minút? Všetci? Všetci členovia všetkých tímov všetkých produktov? A funguje to?

Áno, funguje to ak Vy (alebo ktokoľvek kto riadi míting) je striktným v udržaní ho krátkym.

Formát mítingu je:

1) Novinky a aktualizácie od šéfov vývoja. Informácie o nadchádzajúcich udalostiach.

2) Round-robin. Jedna osoba z každého produktu reportuje čo urobili za posledný týždeň, čo plánujú dokončiť tento týždeň a akékoľvek problémy. Aj niektoré iné osoby reportujú (CM lead, QA lead,atď.).

3) Ktokoľvek iný môže slobodne dodať informácie alebo sa spýtať otázky.

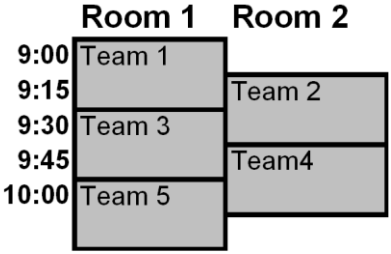
Je to fórum pre stručné informácie, nie diskusiu alebo reflexiu. Ponechaním to v tejto rovine 15 minút zvyčajne stačí. Niekedy je dlhší, ale len málokedy presiahneme celkovo 30 minút. Ak sa objaví zaujímavá diskusia, preruším ju a zainteresovaných pozvem, aby ostali po mítingu a pokračovali v diskusii.

Prečo robíme míting so všetkými? Pretože sme si všimli, že korporátna úroveň bola zväčša o reportovaní. Iba zriedka sme mali diskusie v takejto skupine. Navyše, mnoho ľudí bolo hladných po tomto type informácií. Jednoducho tímy chcú vedieť čo robia iné tímy. Takže, keď sme zistili, že sa stretneme a strávime nejaký čas informovaním sa navzájom čo ktorý tím robí, prečo nezavolať aj iných?

Prekrývanie sa denných scrumov

Ak máte veľa Scrum tímov v jednom produkte a všetci robia denný scrum v rovnakom čase, máte problém. Produktový vlastník (a zvedaví ľudia ako ja) môžu prísť iba na jeden denný scrum.

Takže sme poprosili tímy, aby sa vyhli denným scrumom v rovnakom čase.



Príklad hore je z periódy keď sme mali denné scrumy v separátnych miestnostiach a nie v jednej tímovej miestnosti. Mítingy normálne trvali 15 minút, ale každý tím dostal 30 minútové okno pre prípad, že nedodržia čas.

Je to *extrémne použiteľné* z dvoch dôvodov:

1. Ľudia ako produktový vlastník a ja môžu navštíviť *všetky* denné scrumy v jedno ráno. Neexistuje lepší spôsob ako získať presný obraz o sprintoch a o kľúčových problémoch.
2. Tímy môžu navštíviť aj iné denné scrumy. Nestáva sa to často, ale ak dva tímy robia v rovnakej oblasti, zväčša niekoľko členov tímu ostáva na

iných denných scrumoch, aby ostali synchronizovaní.

Nevýhodou je menej slobody pre tím – nemôžu si vybrať čas, kedy by chceli robiť denný scrum. Pre nás to nikdy nebol problém.

Tímy hasičov

Mali sme situácie, kedy veľký produktový tím nebol schopný adoptovať Scrum, pretože strávil veľa času hasením, teda panickými opravami chýb na ich predčasne uvoľnenom systéme. Boli to skutočne brutálne cykly, boli tak zaujatí hasením problémov, že nemali čas na robenie proaktívnej práce na *prevencii* požiarov (zlepšenie dizajnu, automatizácia testov, vytvorenie monitorovacích nástrojov, atď.).

Tieto problémy sme riešili vytvorením špecializovaného tímu hasičov a vyhradeného Scrum tímu.

Prácou Scrum tímu bolo (s požehnaním produktového vlastníka) skúsiť stabilizovať systém a efektívne predísť požiarom.

Hasiči (volali sme ich „podpora“) mali dve úlohy:

1) Hasiť požiare.
2) Chrániť scrum tím pred všetkými prerušeniami, vrátane vecí ako ad-hoc vlastností.

Hasiči boli umiestnení blízko dverám; scrum tím bol umiestnený vzadu v miestnosti. Takže hasiči tak *fyzicky chránili* scrum tím pred prerušeniami ako napr. hladní obchodníci a nahnevaní zákazníci.

Vývojári seniori boli v oboch tímoch, takže ani jeden tím nebol veľmi závislý v kľúčových kompetenciách navzájom.

Bol to pokus pre riešenie Scrum bootstrapping problém. Ako môžeme začať robiť Scrum, keď tím nemá šancu plánovať ich prácu viac než na jeden deň? Našou stratégiou bolo, ako som už povedal, rozdeliť skupinu.

Fungovalo to celkom pekne. Keďže scrum tím mal priestor pracovať proaktívne, nakoniec stabilizovali systém.

Medzitým hasiči sa kompletne vzdali plánovania dopredu, pracovali úplne reaktívne, iba opravou panických chýb, ktoré prichádzali.

Samozrejme, že scrum tím nebol *úplne* nerušený. Často hasiči zatiahli kľúčových ľudí zo Scrum tímu alebo prinajhoršom celý tím.

Akokoľvek, po niekoľkých mesiacoch bol systém stabilizovaný dosť nato, že sme mohli rozpustiť tím hasičov a vytvoriť ďalšie scrum tímy. Hasiči boli šťastní, že mohli odložiť ich helmy a pripojiť sa k scrum tímom.

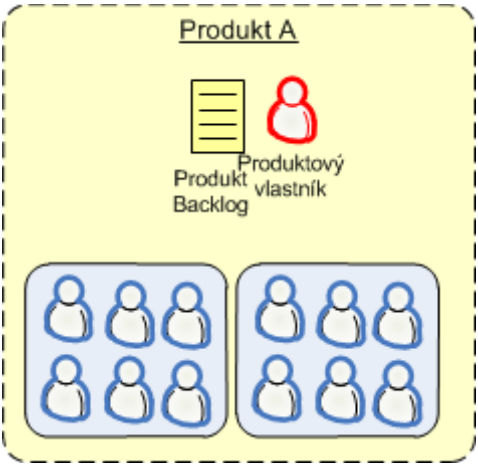
Rozdeliť produktový backlog – alebo nie?

Povedzme že máme jeden produkt a dva Scrum tímy. Koľko produktových backlogov máme mať? Koľko produktových vlastníkov? Vyskúšali sme tri modely. Výber má výrazný efekt na spôsob vedenia plánovacích mítingov.

Stratégia 1: Jeden produktový vlastník, jeden backlog

Je to model „Iba jeden“. Náš preferovaný model.

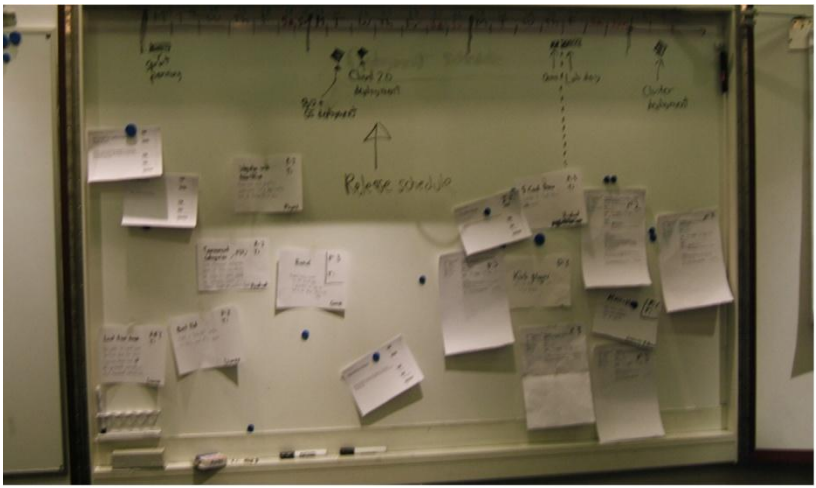
Dobré na tomto modeli je, že môžete nechať svojich členov tímu formulovať seba samých na základe top priorít produktového vlastníka. Produktový vlastník sa môže zamerať na to *čo je potrebné* a potom nechať členov tímu si rozdeliť prácu.



Aby som bol viac konkrétny, tu je popis ako plánovací míting sprintu funguje v takomto tíme:

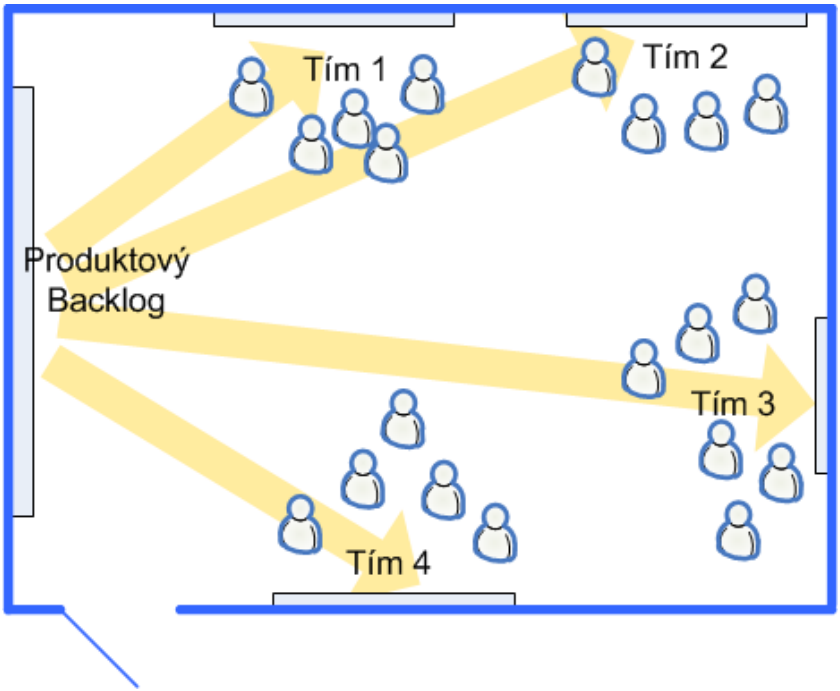
Plánovací míting sprintu je v externom konferenčnom centre.

Tesne pred mítingom produktový vlastník deklaruje jednu stenu ako „stenu produktového backlogu“ a umiestni na ňu stories (ako karty), utriedené relatívne podľa priority. Umiestňuje ich dovtedy, kým nie je stena plná, čo je zvyčajne viac než dosť položiek pre sprint.



Každý Scrum tím si vyberie prázdnu časť steny pre seba a a nalepí názov tímu na stenu. Je to ich „tímová stena“. Každý tím umiestni stories zo steny produktového backlogu, pričom začnú s najdôležitejšou story a ukladajú indexové karty na stenu svojho tímu.

Na obrázku dole je tento postup ilustrovaný, pričom žlté šípky symbolizujú tok indexových kariet zo steny produktového backlogu na steny tímov.



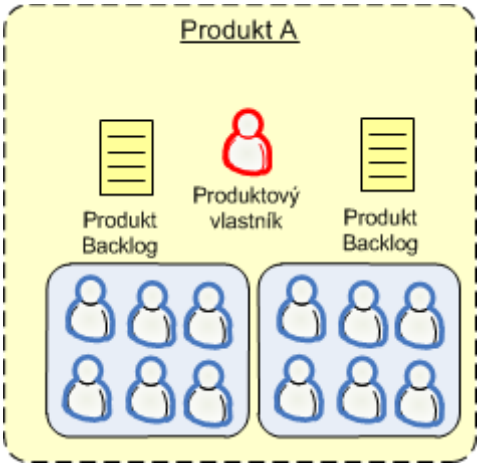
Ako míting prebieha, produktový vlastník a tímy sa dohadujú nad indexovými kartami, presúvajú ich medzi tímami, menia im priority, rozdeľujú ich na menšie časti, atď. Po nejakej tej hodine má každý tím prvú verziu backlogu sprintu na svojej stene. Potom tímy pracujú nezávisle odhadovaním času a rozdelením na úlohy.

Je to chaotické a vyčerpávajúce, ale zároveň efektívne, zábavné a sociálne. Keď uplynie čas, všetky tímy obyčajne majú dostatok informácií pre naštartovanie ich sprintu.

Stratégia 2: Jeden produktový vlastník, viac backlogov

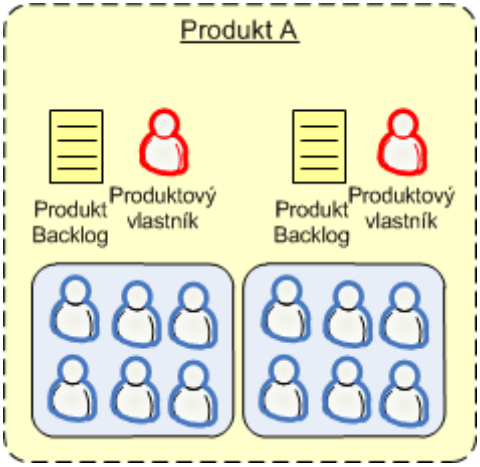
Pri tejto stratégii jeden produktový vlastník spravuje *viacero* produktových backlogov, jeden pre každý tím. Zatiaľ sme tento prístup nevyskúšali, ale boli sme už dosť blízko. Je to náš záchranný postup pre prípad zlyhania prvého prístupu.

Slabinou tejto stratégie je, že produktový vlastník alokuje stories do tímov, čo je úloha, ktorú tímy samotné zvyčajne robia lepšie.



Stratégia 3: Viacero produktových vlastníkov, jeden backlog na vlastníka

Je podobná druhej stratégii, jeden produktový backlog na tím, ale s dodatkom *jeden produktový vlastník* na tím!



Zatiaľ sme to neurobil a pravdepodobne ani nikdy neurobíme.

Ak dva produktové backlogy zasahujú rovnakú časť kódu, obyčajne to spôsobí vážne konflikty záujmov medzi dvoma produktovými vlastníkmi.

Ak dva produktové backlogy zasahujú rôzne časti kódu, je to rovnaké ako rozdelenie celého produktu do separovaných podproduktov, nad ktorými sa dá pracovať nezávisle. To znamená návrat späť k jeden-tím-na-produkt, čo je pekné a jednoduché.

Vetvenie kódu

S viacerými tímami pracujúcimi na rovnakej báze kódu sa nevyhnutne musíme zaoberať s vetvením kódu pomocou SCM (software configuration management) systému. Je množstvo kníh a článkov ako postupovať v prípade viacerých ľudí pracujúcich na rovnakej báze kódu, takže nepokračujem ďalej do detailov. Napriek tomu ale zosumarizujem niektoré najdôležitejšie lekcie, ktoré sa naše tímy doteraz naučili:

- Buďte striktní udržiavaním hlavnej línie (trunku) v konzistentnom stave. Prinajmenšom to znamená, že všetko má byť kompilovateľné a všetky unit testy musia prejsť. Musí byť možné vytvoriť funkčný release *kedykoľvek*. Preferovaným je systém kontinuálneho prekladu a automatický deploy do testovacieho prostredia každú noc.
- Označiť každý release. Kedykoľvek releasujete pre akceptačné testy alebo do produkcie, uistite sa, že verzia je označená v hlavnej línii a označuje presne to, čo bolo uvoľnené. Znamená to, že v akomkoľvek momente sa môžete vrátiť späť a vytvoriť vetvu pre údržbu z tohto bodu.
- Novú vetvu vytvorte iba ak je to nevyhnutné. Dobrým pravidlom je vytvoriť novú vetvu *iba* ak nemôžete použiť existujúcu vetvu bez porušenia pravidiel. Ak máte pochybnosti, nevetvite. Prečo? Pretože každá aktívna vetva je ocenená administráciou a komplexnosťou.
- Použite vetvy primárne pre separáciu *rôznych cyklov života*. Môžete alebo nemusíte sa rozhodnúť, aby každý scrum tím mal svoj kód vo svojej vlastnej vetve. Ale ak zmiešate krátkodobé

opravy s dlhotrvajúcimi zmenami v rovnakej vetve, zistíte, že je veľmi ťažké releasovať opravy!

- Synchronizujte často. Ak pracujete na vetve, synchronizujte do hlavnej línie kedykoľvek máte niečo, čo sa dá buildovať. Synchronizujte z hlavnej línie do vašej vetvy každý deň keď prídete do práce, tak aby vaša vetva bola aktuálna. Ak to pre vás znamená Merge-Peklo, len akceptujte fakt, že bolo by oveľa horšie čakať.

Multi-tímové retrospektívy

Ako robíme retrospektívy tímov, keď máme viacero tímov pracujúcich na rovnakom produkte?

Okamžite po deme sprintu, po potlesku, každý tím odíde to samostatnej miestnosti alebo na nejaké komfortné miesto mimo kancelárie. Urobia si ich vlastnú retrospektívu tak ako som popísal na strane 77 „Ako robíme retrospektívy sprintov“.

Počas mítingu plánovania sprintu (na ktorom sa zúčastňujú všetky tímy, keďže synchronizujeme sprinty v každom produkte), prvou vecou, ktorú robíme je nechať jedného hovorcu z každého tímu sa postaviť a zosumarizovať kľúčové body z ich retrospektívy. Trvá to 5 minút na tím. Potom máme otvorenú diskusiu počas 10-20 minút. Potom máme prestávku a neskôr začneme plánovanie sprintu.

Nevyskúšali sme iné spôsoby pre viaceré tímy, pretože to zatiaľ funguje dostatočne dobre. Najväčšou nevýhodou je, že nie je žiadna medzera po retrospektíve a pred časťou plánovania sprintu. (Pozri stranu 96, Čas medzi sprintmi).

Pre produkty s jedným tímom nerobíme žiaden sumár retrospektívy počas mítingu plánovania sprintu. Nie je to potrebné, keďže všetci boli prítomní na poslednej retrospektíve.

16

Ako zvládame geograficky distribuované tímy

Čo sa stane ak členovia tímu sú v geograficky rôznych miestach?

Väčšina z „tajomstva“ Scrum a XP je založená na úzko spolupracujúcich tímoch sediacich spolu, ktorí sa stretávajú tvárou v tvár každý deň.

Niektoré tímy máme geograficky separované a takisto máme aj členov tímov pracujúcich z času na čas z domu.

Naša stratégia je jednoduchá. Používame akýkoľvek trik, o ktorom vieme, aby sme maximalizovali komunikačný tok medzi fyzicky separovanými členmi tímu. Nemyslím tým komunikačný tok ako Mbit/sekundu (aj keď aj toto je takisto dôležité). Myslím tým komunikačný tok v širšom zmysle:

- Schopnosť párovo programovať.
- Schopnosť stretnúť sa tvárou v tvár počas denného scrumu.
- Schopnosť mať kedykoľvek diskusiu tvárou v tvár.
- Schopnosť sa fyzicky stretnúť a socializovať sa.
- Schopnosť mať spontánne mítingy s celým tímom.
- Schopnosť vidieť rovnaký pohľad na backlog sprintu, burndown, produktový backlog a iné informačné panely, radiátory.

Niektoré z opatrení, ktoré sme implementovali (alebo implementujeme, nie všetky sme už urobili):

- Webové kamery, slúchadlá a mikrofóny pri každej pracovnej stanici.
- "Remote-enabled" konferenčné miestnosti s web kamerami, konferenčnými mikrofónmi, vždy zapnutými, vždy pripravenými počítačmi, aplikáciami pre zdieľanie pracovnej plochy, atď.
- „Vzdialené okná". Veľké obrazovky na každom mieste ukazujúce permanentný pohľad na iné miesta. Tak trochu ako virtuálne okno medzi dvoma oddeleniami. Môžete tam stáť. Môžete vidieť kto je za svojim stolom, kto sa s kým rozpráva. To vytvára pocit „hej, my sme v tom spolu".
- Výmenné programy, kde ľudia z každej lokácie cestujú a navštevujú sa navzájom na pravidelnej báze.

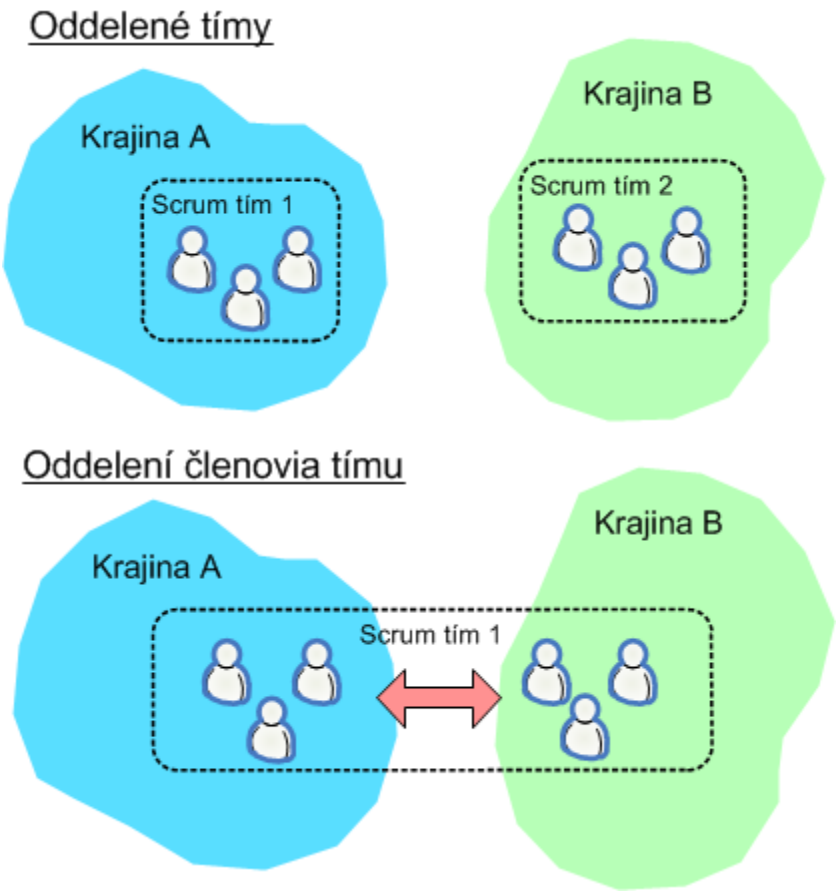
Použitím týchto techník sa pomaly ale isto dostávame k tomu, ako postupovať s mítingami plánovania sprintov, demami, retrospektívami, dennými scrum mítingami, aj keď sú členovia tímu distribuovaní geograficky.

Ako zvyčajne je to o experimentovaní. Preskúmať=>Prispôsobiť sa=> Preskúmať=> Prispôsobiť sa=> Preskúmať=> Prispôsobiť sa=>

Offshoring

Máme niekoľko offshore tímom a experimentovali sme s tým, ako ich spravovať použitím Scrumu.

Sú dve hlavné stratégie" oddelené tímy alebo oddelení členovia tímu.



Prvá stratégia, separované tímy, je presvedčivým výberom. Avšak, začali sme s druhou stratégiou, oddelenými členmi tímu. Je tu viacero dôvodov.

1. Chcem, aby sa členovia tímu navzájom spoznali.
2. Chceme excelentnú komunikačnú infraštruktúru medzi dvoma lokáciami, a chceme dať tímom silný impulz pre jej nastavenie.
3. Na začiatku je offshore tím veľmi malý pre sformovanie efektívneho scrum tímu s ponechaním zodpovednosti na nich.

4. Chceme obdobie intenzívneho zdieľania vedomostí predtým, než offshorové tímy budú uskutočniteľnou možnosťou.

V dlhodobom fungovaní možno smerujeme k stratégii „separovaných tímov“.

Členovia tímov pracujúci z domu

Práca z domu môže byť niekedy skutočne dobrá. Niekedy môžete toho viac naprogramovať doma za jeden deň ako za celý týždeň v práci. Prinajmenšom ak nemáte deti ☺.

Jedným z fundamentálnych základov v Scrume je, že celý tím by mal byť fyzicky spolu. Takže ako to máme urobiť?

V podstate nechávame rozhodnutie na tímoch kedy a ako často je OK pracovať z domu. Niektorí členovia tímu pracujú z domu pravidelne kvôli dlhému dochádzaniu. Napriek tomu povzbudzujeme tímy byť fyzicky spolu „väčšinu“ času.

Keď členovia tímu pracujú z domu, sú súčasťou denných scrum mítingov cez volania pomocou Skype (niekedy aj video). Sú online väčšinu dňa cez instantné správy. Nie je to rovnako dobré ako v rovnakej miestnosti, ale dostatočne dobré.

Raz sme vyskúšali koncept mať stredu nastavenú ako *fokus deň*. V podstate to znamená „ak chcete pracovať z domu, tak je to dobre, ale urobte tak v stredu. A dohodnite sa s tímom“. Fungovalo to v tíme, na ktorom sme to vyskúšali. Väčšina z tímu ostala doma v stredy, a napriek tomu spolupracovali dobre. Keďže to bol iba jeden deň, členovia tímu sa veľmi nerosynchronizovali s ostatnými. Hoci z nejakých dôvodov to nikdy tak dobre nefungovalo s ostatnými tímami.

Vo všeobecnosti nebol pre nás problém pracovať s ľuďmi pracujúcimi z domu.

17

Scrum master – kontrolný zoznam

V tejto záverečnej kapitole Vám ukážem náš kontrolný zoznam používaný Scrum Mastrom. Veci, ktoré sú ľahko zabudnuteľné. Vynechali sme obvyklé veci ako „odstrániť prekážky tímu“.

Začiatok sprintu

- Po mítingu plánovania sprintu vytvorte informačnú stránku sprintu.
    - Pridajte do dashboardu wiki linku na túto stránku.
    - Vytlačte stranu a umiestnite ju na stenu, okolo ktorej prechádzajú členovia Vášho tímu.
- Pošlite email všetkým oznamujúci začiatok nového sprintu. Zahrňte cieľ a linku na informačnú stránku.
- Zaktualizujte dokument so štatistikami. Pridajte vašu odhadovanú rýchlosť, veľkosť tímu, dĺžku sprintu atď.

Každý deň

- Uistite sa, že denný scrum míting začal a skončil načas.
- Uistite sa, že stories sú pridané/odstránené z backlogu sprintu a podľa potreby udržte sprint tak ako je nadplánové.
    - uistite sa, že produktový vlastník je upozornený o týchto zmenách.
- Uistite sa, že backlog sprintu je aktuálny podľa tímu.
- Uistite sa, že problémy sú vyriešené alebo reportované produktovému vlastníkovi a/alebo vedúcemu vývoja.

Koniec sprintu

- Urobte verejné demo sprintu.
- Každý by mal byť informovaný o deme jeden alebo dva dni predtým.
- Retrospektívu sprintu urobte s celým tímom a produktovým vlastníkom. Pozvite vedúceho vývoja takže vám pomôže rozšíriť čo ste sa naučili.
- Aktualizujte štatistiky sprintu. Pridajte aktuálnu rýchlosť a kľúčové body retrospektívy.

18

Na záver

Fuj! Nikdy som si nemyslel, že to bude také dlhé.

Dúfam, že táto publikácia Vám poskytla niektoré užitočné nápady, či už ste nový v Scrume alebo veterán.

Keďže Scrum musí byť pripravený na mieru špecificky pre každé prostredie, je ťažké konštruktívne argumentovať o najlepších praktikách na všeobecnej úrovni. Avšak zaujíma ma vaša odozva. Povedzte mi ako sa Váš prístup líši od môjho. Poskytnite mi návrhy na zlepšenie!

Kontaktujte ma emailom **henrik.kniberg@crisp.se**.

Tiež sledujem **scrumdevelopment@yahoogroups.com**.

Ak sa Vám táto kniha páči, možno by ste chceli čas od času skontrolovať môj blog. Dúfam, že budem publikovať príspevky o Jave a agilnom vývoji software.

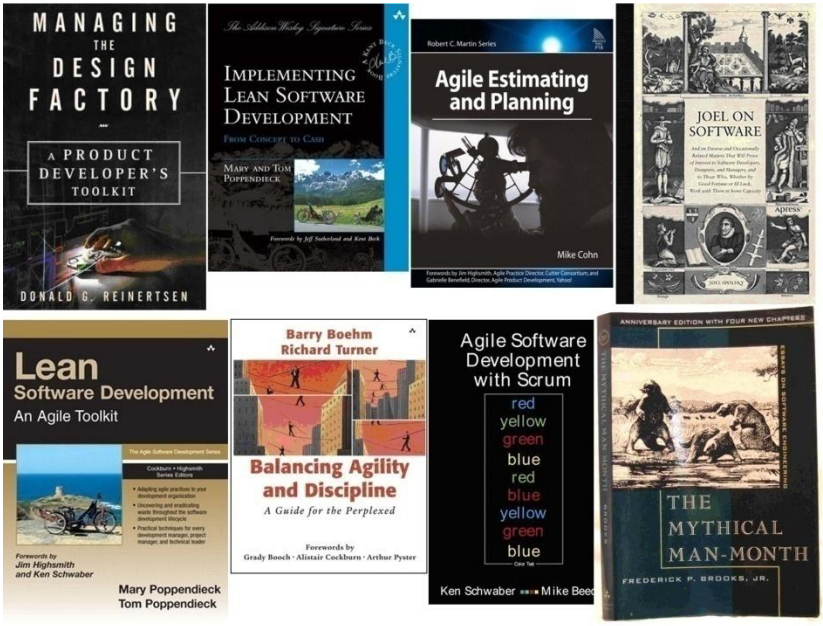
http://blog.crisp.se/henrikkniberg/

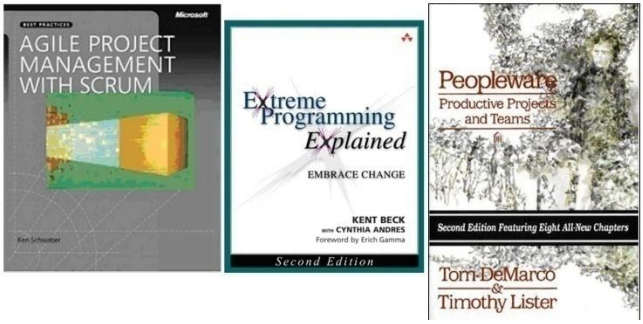
Ó a nezabudnite...

Je to len práca, že?

Odporúčané čítanie

Tu sú niektoré knihy, ktoré mi poskytli množstvo inšpirácie a nápadov. Veľmi odporúčam!





O autorovi

Henrik Kniberg (henrik.kniberg@crisp.se) je konzultantom v Crisp sídliacej v Štokholme, špecializujúci sa na Javu a agilný softvérový vývoj.

Už keď sa objavili prvé XP knihy a Agilný Manifest, Henrik prijal agilné princípy a skúsil sa naučiť ako ich efektívne aplikovať v rôznych typoch organizácií. Ako zakladateľ a CTO Goyada od 1998-2003 mal množstvo príležitostí experimentovať s vývojom riadenom testami a inými agilnými praktikami počas toho ako budoval a riadil technické platformy a 30 členný vývojový tím.

Na konci roku 2005 bol Henrik kontraktovaný ako vedúci vývoja vo švédskej spoločnosti zameranej na herný priemysel. Spoločnosť bola v kritickej situácii s urgentnými organizačnými a technickými problémami. Použitím Scrum-u a XP ako nástrojov, Henrik pomohol spoločnosti dostať sa z krízy implementáciou agilných a lean princípov na všetkých úrovniach spoločnosti.

Jeden piatok v novembri 2006 ležal Henrik doma v posteli s horúčkou a rozhodol sa zozbierať nejaké poznámky pre seba o tom, čo sa naučil za posledných pár rokov. Keď začal písať, zistil, že sa nevie zastaviť a po troch dňoch usilovného písania a kreslenia sa počiatočné poznámky rozvinuli do 80 stranového článku označeného ako “Scrum and XP from the Trenches”, ktorá sa neskôr stala touto knihou.

Henrik použil holistický prístup a vychutnal si adaptáciu rôznych rolí ako manažér, vývojár, scrum master, učiteľ a tréner. Je nadšencom pomáhania spoločnostiam vytvoriť excelentný softvér a excelentné tímy, pričom berie akúkoľvek rolu, ktorá je potrebná.

Henrik vyrástol v Tokiu a teraz žije v Štokholme s jeho manželkou Sofiou a dvoma deťmi. Vo svojom voľnom čase je aktívnym muzikantom, skladajúcim hudbu a hrajúcim na basu a klávesy s lokálnymi skupinami.

Pre viac informácii pozrite
http://www.crisp.se/henrik.kniberg