VŠE

Využití metodiky Scrum pro velké projekty – Scrum of Scrums

Bc. Martin Křena

25.10.2011

Obsah

1 Úvod

Vývoj softwaru je složitý proces, který s sebou nese riziko neúspěchu. Abychom se neúspěchu vyvarovali, je potřeba vývoj řídit. K řízení vývoje softwaru slouží metodiky, které nám udávají postup, jak by měl vývoj probíhat. Výběr vhodné metodiky patří k nejdůležitějším krokům k úspěšnému ukončení projektu. Hlavní problém, se kterým se při zkoumání metodik vývoje softwaru setkáme, spočívá v tom, že metodik je velké množství, nejsou dobře kategorizovány ani jednotným způsobem popsány, což velmi znesnadňuje vyhledání vhodné metodiky pro určitý typ projektu [Buchalcevová, 2009]. Kritérií pro kategorizaci metodik je mnoho, v praxi však rozlišujeme především dvě kategorie. Jsou to Rigorózní (těžké) metodiky a agilní (lehké) metodiky. Agilní metodiky se začali rozšiřovat až po roce 2001, kdy byl sepsán Manifest agilního vývoje, který definoval základní principy pro lepší vývoj softwaru. Mezi agilní metodiky řadíme: Dynamic Systems Development Method (DSDM), Adaptive Software Development (ASD), Feature-Driven Development (FDD), OpenUP, Extreme Programming (XP), Crystal metodiky, Agilní modelování. Mezi agilní metodiky patří také metodika *Scrum*, které se bude věnovat tato práce.

Cílem této práce bude charakterizovat využití metodiky *Scrum* pro velké, nedistribuované projekty.

Abych tohoto cíle dosáhl, tak nejdříve stručně charakterizuji samotnou metodiku *Scrum*. Následně nastuduji odbornou literaturu a poté na základě načerpaných znalostí charakterizuji použití metodiky *Scrum* pro velké projekty.

Čtenář této práce by měl mít základní znalosti o metodikách vývoje softwaru a také o představu o tom jak v dnešní době probíhá vývoj softwaru.

Jako omezení této práce vidím mojí malou zkušenost s vývojem softwaru a také malé zkušenosti s metodikou *Scrum*. Další problém při psaní této práce je nedostatek relevantní odborné literatury, na základě které, je možné splnit cíl práce.

2 Scrum

Scrum v překladu z angličtiny znamená mlýn v ruby a tento název byl vybrán proto, že metodika Scrum je podobně jako rugby, adaptivní, rychlá a postavená na samoorganizujících týmech [Buchalcevová, 2009]. Jak již bylo zmíněno v úvodu metodika Scrum patří mezi agilní (lehké) metodiky vývoje softwaru. Podle výsledků průzkumů [Ambler, 2006] patří *Scrum* mezi nejpoužívanější agilní metodiky. Vývoj podle *Scrumu* probíhá v iteracích, takže mluvíme o iterativním vývoji softwaru. Scrum je zaměřen hlavně na řízení projektu. Autoři metodiky jsou přesvědčeni, že vývoj softwaru není definovatelný proces, ale je empirický proces, který vyžaduje odlišný styl řízení [Buchalcevová, 2009].

2.1 Charakteristika metodiky

V následujících dvou podkapitolách stručně charakterizuji metodiku *Scrum*. Nejdříve si ukážeme, jaké role metodika vymezuje, a pak se podíváme, jak vypadá životní cyklus projektu řízený metodikou *Scrum*.

2.1.1 Role

Scrum rozlišuje pouze tři základní role: *Product Owner*, *Scrum Master* a *Team*. Názvy rolí není možné do češtiny rozumně přeložit a tak pouze pro roli *Team* budu používat český překlad tým.

Product Owner

Product Owner by měl být pouze jedna osoba, která je odpovědná za tvorbu a úpravy *Product Backlog*. Dále musí zajistit, aby s tímto seznamem byli všichni členové týmu seznámeni a aby položkám v seznamu rozuměli [Deemer, 2009]. *Product Owner* jako jediný komunikuje se zákazníkem a jeho požadavky předává osobě *Scrum Master* a týmu.

Scrum Master

Scrum Master podporuje osobu *Product Owner* a také tým. Je to v podstatě takový projektový manažer. Zajišťuje, aby tým zůstal motivovaný a udržoval pracovní tempo. Odpovídá za dodržování jednotlivých praktik metodiky. Role *Scrum Master* a *Product Owner* se vzájemně doplňují [Pichler, 2010].

Team

Členové týmu jsou pracovníci, kteří tvoří produkt. Členové týmu musí mít k sobě blízký vztah a pracovat v symbióze a měli by být schopni dělat více druhů práce. Mezi dané dovednosti patří programování, tvoření návrhů, testování, obchodní analýza apod. Místnost, kde tým pracuje, by měla být vhodná pro kreativní a týmovou spolupráci.

2.1.2 Životní cyklus projektu

Na začátku životního cyklu projektu se koná *Release Planning Meeting* jehož cílem je vytvořit plán a cíle, kterým Scrum týmy a ostatní účastníci rozumí. *Product Owner* na základě *Release Planning meeting* sestaví plán toho co je potřeba udělat. Součástí toho plánu je také Product Backlog. Product Backlog je seznam všech požadavků na produkt, který se v jednotlivých fázích vývoje upravuje [Schwaber, 2009].

Práce je dělána ve *Sprintech*. Každý *Sprint* je iterace (většinou) 30 po sobě jdoucích dní. *Sprint* je zahájen poradou zvanou *Sprint plannig meeting*. Na této poradě tým společně s osobou *Product Owner* spolupracují na tom co je potřeba udělat v dalším *Sprint*u. Členové týmu si pak vybírají úkoly, na kterých chtějí pracovat. *Sprint plannig meeting* by nikdy neměl být delší než 8 hodin. Cílem je co nejdříve začít pracovat a ne o práci přemýšlet [Schwaber, 2009].

Každý den se koná 15minutová porada zvaná *Daily Scrum*, která slouží k synchronizaci práce všech členů týmu. Na této poradě každý člen týmu odpoví na tři otázky [Schwaber, 2009]:

- Co jste udělal na tomto projektu od poslední porady?
- Co máte v plánu na tomto projektu dělat do další porady?
- Jaké překážky vám stojí v cestě ke splnění vašich závazků k tomuto *Sprint*u a tomuto projektu?

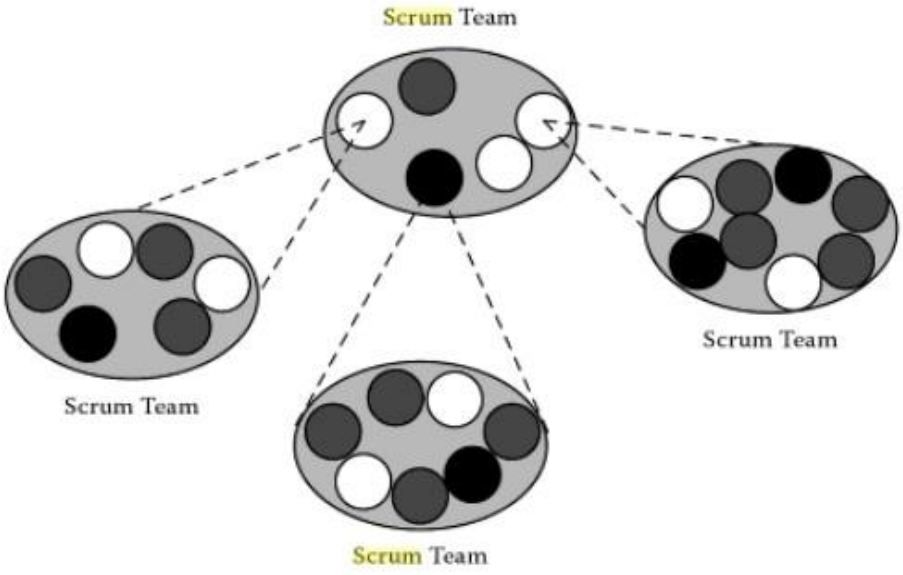
Na konci *Sprintu* se koná porada zvaná *Sprint review meeting*. Je to čtyřhodinová porada, kde tým prezentuje osobě *Product Owner* a dalším zainteresovaným stranám to, co bylo během *Sprintu* vyvinuto. Tato neformální porada pomáhá spojovat lidi v týmu a tím prospívá v jejich spolupráci. Ještě než začne další *Sprint plannig meeting* osoba v roli *Scrum Master* zorganizuje *Sprint retrospective meeting*. Během této tříhodinové porady se *Scrum Master* snaží získat od týmu zpětnou vazbu, aby byl příští *Sprint* víc efektivní [Schwaber, 2009].

2.2 Využití metodiky ve velkých projektech

Čím je projekt složitější tím, roste objem a důležitost komunikace. Metodika Scrum je určena jako většina agilních metodik spíše pro malé týmy. Malé týmy jsou schopny realizovat v přijatelném čase pouze malé projekty. Velké projekty musí řešit velký tým nebo spíše malých týmů. Nyní se podíváme jak se metodika Scrum dá použít ve velkých, nedistribuovaných projektech.

U velkých projektů je složení týmů obdobné jako u malých projektů řízených metodikou *Scrum*. Tým se skládá většinou ze sedmi pracovníků. Každý tým musí mít svého *Scrum Mastera*. U velkých projektů týmy spadají do kaskádové hierarchie *Scrum* týmů. Tato hierarchie se nazývá *Scrum of Scrums.* Tento hierarchický koncept je podobný jako typické organizační schéma a poskytuje metody pro rozklad rozsáhlého *Product Backlog* do menších *Sub-Backlogs* (*Product Backlog* rozdělený na

menší části) a jejich přidružených *Sprint Backlogs*. U *Scrum of Scrums* se *Scrum Master*s nebo jiní vybraní členové týmu z jedné úrovně se stávají součástí týmu na další, vyšší úrovni v hierarchii (viz obrázek 1) [Pries, Quigley, 2010].



Obrázek 1 - *Scrum of Scrums* (zdroj: [Pries, Quigley, 2010])

2.2.1 *Release Planning*

Release Planning u velkých projektů vyžaduje trochu odlišný přístup, než jaký je běžný u menších projektů.*Release planning meetingu* se účastní všechny týmy spolu se svými *Scrum Mastery*. Když se několik týmů podílí na sestavování *Product Backlog,* tak je potřeba dosáhnout shody na základních východiskách pro rozsah a způsob psaní *User Stories* (*User Story* – popisuje funkcionalitu vyvíjeného softwaru) a také pro odhady náročnosti jednotlivých *User Stories*. Pokud by se tak nestalo, tak by různě napsané *User Stories* byly zmatečné a odhady náročností jednotlivých položek by byly velmi zkreslující.

Pomáhat každému týmu k úspěchu a zároveň optimalizovat průběh celého projektu vyžaduje nějakou práci navíc. První věc, kterou musíme udělat, je podívat se dopředu na dva až tři *Sprint*y, abychom pochopili, na kterých položkách z *Product Backlog* se bude pravděpodobně pracovat. To vyžaduje rozklad a vypilování *Product Backlog* dříve, než se podrobnější položky objeví v horní části *Product Backlog* a bude potřeba na nich začít pracovat. Dalším krokem je identifikovat závislosti mezi týmy, již při *Release plannig meeting*,aby bylo poté možné co nejrychleji zvládnout *Sprint planning meeting* a mohlo se začít na projektu pracovat. Závislosti mezi týmy odhalíme zodpovězením následujících tří otázek [Pichler, 2010]:

- Budou nějaké týmy pracovat na stejné funkci nebo komponentě?

- Má nějaký tým fungovat jako dodavatel jiného týmu?
- Pokud ano, je možné dodávat funkce nebo součásti a použít je ve stejném *Sprintu*?

Pro odstranění problematické závislosti budeme muset upravit prioritu položek v *Product Backlog*. Například, místo toho, aby dva týmy pracovaly na stejné části v příštím *Sprint*u, můžeme zvážit přesunutí některých požadavků až na další *Sprint*. Po vyřešení jakékoliv problematické závislosti považujeme pracovní zatížení pro každý tým za optimální. Bude nějaký tým pravděpodobně přepracovaný v příštím *Sprint*u? Nebo bude nějaký tým nevyužitý? Pokud na některou z těchto otázek je odpověď ano pak je potřeba se vrátit k *Product Backlog* a upravit priority jednotlivých položek. Jakmile je toto hotovo přidáme požadavky na příští dva až tři *Sprint*y do *Release Plan. Release Plan* je vylepšená verze *Release Burndown,* který ukazuje týmu a všem zainteresovaným osobám, jak se postupuje při vývoji a znázorňuje odhadovaný zbývající čas, který je potřeba na dokončení projektu [Pichler, 2010].

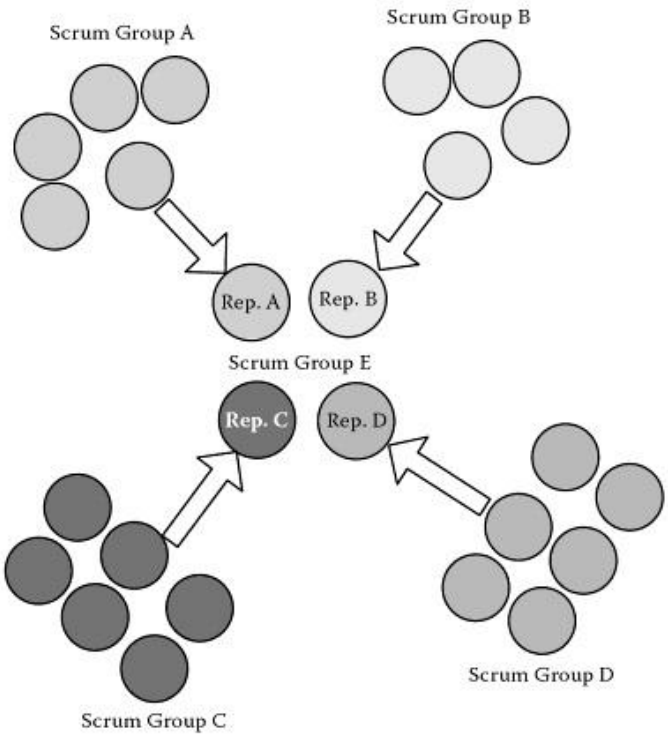
2.2.2 *Sprint Planning*

Sprint planning *meeting* probíhá na začátku iterace. Hned na začátku *Sprintu*, kromě toho prvního, je potřeba odhalit závislosti mezi týmy a upravit prioritu položek v *Product Backlog* tak jak je to popsáno v předchozí podkapitole. Následně jednotlivé týmy dostanou přiděleny jednotlivé položky z *Product Backlog*. Každý tým si pořádá vlastní *Sprint Planning meeting*, kde řeší to, na čem bude v následujícím *Sprintu* pracovat.

U velkých projektů je také důležité, aby se části týmů nebo přinejmenším zástupci jednotlivých týmů sešli na začátku *Sprint* planning *meeting* a informovali ostatní, na čem budou v daném *Sprintu* pracovat. Toto společné setkání zástupců jednotlivých týmů slouží také k porozumění cílů *Sprint*u, ke kterým by měly všechny týmy přispívat [Pichler, 2010].

2.2.3 *Scrum of Scrums*

Scrum of Scrums meeting je důležitá technika v přizpůsobení *Scrumu* na velké projekty. Tyto meetingy se konají většinou jednou týdně a umožňují týmu diskutovat o jejich práci, zvláště pak o překrývajících se částech softwaru a integraci jednotlivých částí. Představte si dokonale vyvážený projekt složený ze sedmi týmů po sedmi členech v týmu. Každý ze sedmi týmů bude provádět vlastní každodenní *Daily Scrum meeting*. Každý tým pak určí jednu osobu, která zúčastní jednou týdně pořádaného *Scrum of Scrums meetingu* (obrázek 2). Vybraná osoba by měla být spíše technicky zaměřená (např.: programátor, tester, správce databáze), radši než *Product Owner* nebo *Scrum Master* [Pries, Quigley, 2010].



Obrázek 2 - *Scrum of Scrums* (zdroj: [Pries, Quigley, 2010])

Obecně platí, že *Scrum Master* je osoba, která bude mít svou roli ve *Scrum of Scrums meeting*, alespoň v počátku projektu. Jednání se časem můžou týkat témat, kde je moudré používat ostatní členy týmu, jako jsou delegáti na další vyšší úrovni v hierarchii. Tento scénář může být přizpůsoben nějaké speciální zkušenosti určitého člena týmu [Pries, Quigley, 2010].

Mezi hlavní přínosy *Scrum of Scrums meetings* patří to, že šíří informace mezi týmy a odhalují případné problémy včas. Dalším přínosem je také to, že otevírají komunikační kanály a tak podporují neformální komunikaci mezi týmy [Šmite, 2010].

2.2.4 *Sprint Review*

Zorganizovat efektivní *Sprint review meeting* s několika týmy, zákazníky a dalšími zainteresovanými stranami je dost náročné. *Sprint review meeting* probíhá podobně jako vědecké veletrhy. Každý tým si zřídí stanoviště, kde předvede, na čem členové týmu během *Sprintu* pracovali. Zákazníci, management a další zainteresované strany vytvoří malé týmy. Každý z těchto posuzovatelských týmů pak začíná na jiném stanovišti, kde jim každý tým během 15 minut předvede vlastní výstup *Sprint*u. Takové prostředí je nabito energií, vzrušení a zábavou [Schatz, 2009]. Dostat všechny do jedné místnosti je skvělý způsob jak komunikovat a sdílet. Pokud není možné uspořádat takový *Sprint Review meeting* v budově společnosti, je potřeba zvážit alternativní umístění jako například konferenční sál [Pichler, 2010].

2.2.5 *Sprint Retrospective*

Týmy si organizují jejich vlastní *Sprint retrospective meeting* a v realizaci opatření ke zlepšení práce na velkém projektu. Ale to nestačí. Pro optimální výsledky by týmy měli identifikovat problém a navrhnout zlepšení. Toto zlepšení pak sdílet s ostatními týmy. Není potřeba, aby se scházeli celé týmy, stačí, když se setkají zástupci týmů a ti mezi sebou sdílí jednotlivá zlepšení. Je sice možné uspořádat *Sprint retrospective meeting* se všemi týmy, ale takové setkání je nákladné a mohlo by trvat i půl dne. Na druhou stranu posiluje společný *Sprint retrospective meeting* vzájemné týmové vztahy a zvyšuje moudrost všech členů týmů. Je možné se rozhodnout i pro kombinaci společných a individuelních *Sprint retrospective meeting*. Společný *Sprint retrospective meeting* pak bude například vždy po každém třetím *Sprint*u [Pichler, 2010].

3 Závěr

V dnešní době je velmi složité vyvíjet software rychle a efektivně podle stále měnících se požadavků zákazníka. Agilní metodiky vývoje softwaru, jsou schopny rychle reagovat na tyto požadavky. Problém agilních metodik a to i metodiky *Scrum* je, že jsou určeny pro malé týmy vývojářů a to znamená, že jsou určeny pro malé projekty.

Cílem této práce bylo charakterizovat využití metodiky *Scrum* pro velké projekty. Na základě studia anglické odborné literatury jsem charakterizoval to, jak je možné metodiku *Scrum* využít ve velkých projektech. Popsal jsem životní cyklus velkého projektu, řízeného metodikou *Scrum*. Životní cyklus velkého projektu se příliš neliší od životního cyklu malého projektu. Vývoj probíhá dle notoricky známých praktik metodiky *Scrum,* ale je ovšem náročnější všechny tyto praktiky skloubit dohromady s několika týmy. Problémy se mohou objevit v nevyžádané závislosti mezi týmy. Další problémy mohou vzniknout v komunikaci mezi týmy. V této práci jsem charakterizoval jak je možné, dle odborné literatury, tyto problémy řešit.

Dle mého názoru je důležité mít v týmech v roli *Scrum* Master lidi, kteří již mají zkušenosti s metodikou *Scrum* alespoň z malých projektů.

Tématika kolem využitelnosti metodiky *Scrum* ve velkých projektech je velmi zajímavá. Tuto práci by bylo vhodné doplnit o případovou studii z reálného nasazení metodiky na určitý projekt.

4 Zdroje

BUCHALCEVOVÁ, Alena. Metodiky budování informačních systémů. Praha: Oeconomica, 2009. 205 s. ISBN 8024515407.

KOŠATA, Václav. Metodika Scrum. Praha, 2010. 51 s. Bakalářská práce. Vysoká škola ekonomická.

PICHLER, Roman. Agile Product Management with Scrum: : Creating Products that Customers Love. USA : Addison-Wesley Professional, 2010. 160 s. ISBN 9780321684134.

PRIES, Kim; QUIGLEY, Jon. Scrum Project Management. Boca Raton : CRC Press, 2010. 198 s. ISBN 9781439825150.

SCHATZ, BoB. The Sprint Review: Mastering the Art of Feedback. ScrumAlliance [online]. 2009, 5, [cit. 2011-12-01]. Dostupný z WWW: <http://www.scrumalliance.org/articles/124-the-sprint-review-mastering-the-art-of-feedback>.

SCHWABER, Ken. Scrum Guide. In Scrum Guide [online]. . : Scrum Aliance, 2009 [cit. 2011-12-01]. Dostupné z WWW: <http://www.itemis.de/binary.ashx/~download/26078/scrum-guide.pdf>.

ŠMITE, Darja; MOE, Nils Brede; ÃGERFALK, Pär J. Agility Across Time and Space: : Implementing Agile Methods in Global Software Projects. Berlin : Springer, 2010. 341 s. ISBN 3642124410.