Metodika agilního vývoje softwaru na OVSS ÚVT

Vendula Švendová, ÚVT MU

1 Úvod

Tento článek popisuje agilní metodiku vývoje softwaru, kterou úspěšně používáme v našem týmu na Oddělení vývoje systémových služeb. Agilní metodika je způsob rozvržení a ověřování práce. Při důsledném dodržování jejích pravidel se lze vyhnout problémům jako jsou:

1. z pohledu zákazníka:
   - software vyvíjený na základě sepsaných požadavků zákazníka po dokončení zákazníkovi nevyhovuje (změnily se či přibyly požadavky) a produkt je třeba přetvořit;
   - produkt není dokončen včas a je tudíž v požadovaném čase nefunkční;
   - produkt není dokončen v rámci rozpočtu;
2. v rámci týmu:
   - jednotliví členové týmu nevědí, na čem pracují jejich kolegové;
   - vedoucí týmu se nepotkává dostatečně často se členy týmu a nemá každodenní čerstvé informace, jak projekt postupuje a jaké řeší členové týmu problémy.

Většina terminologie v článku je uvedena v českém překladu. Některé pojmy jsou však ponechány v původní anglické podobě, abychom zachovali srozumitelnost a spojitost s terminologií užívanou ve většinou anglických textech jiných autorů, stejně jako při běžném provozu.

2 Agilní manifest

Co slovo „agilní“ vlastně znamená? Podle slovníku cizích slov čilý, aktivní, horlivý. Základní určující podmínkou agility v kontextu vývoje programů je možnost změny zadání v průběhu celého vývoje. Zákazník tedy má možnost za chodu svoje požadavky modifikovat, aniž by to mělo za následek masivní přetváření již provedené práce, a tím zbytečné plýtvání časem i zdroji všech zúčastněných.

Agilní metodiky vznikly v polovině 90. let minulého století jako reakce na těžké tradiční metodiky, kterým byla vytýkána byrokratičnost, zkostnatělost, neschopnost flexibilně reagovat na změny.

Tradiční programovací metodiky mají na začátku vývoje fixně danou funkcionalitu, které se snaží dosáhnout; naproti tomu čas a zdroje jsou proměnné – podřizují se oné funkcionalitě. Agilní metodiky naopak mají pevně dané zdroje a časové úseky (tzv. iterace), během nichž se postupně implementují jednotlivé vlastnosti produktu dle jejich priority. Po implementaci každé vlastnosti by měl být systém provozuschopný.

Častá, zákazníky nevítaná, situace „uplynul termín pro dokončení, ale software je stále nefunkční, proto potřebujeme více času“ tak v agilním vývoji nemůže nastat. I v případě, že dojde ke zpoždění, stane se pouze to, že nebudou implementovány vlastnosti s nejnižší prioritou (tudíž nebude ohrožena funkčnost systému).

Agilních metodik existuje několik typů (Extrémní programování, Feature-Driven Development, Dynamic Systems Development Method, Agilní modelování, SCRUM aj.) a jejich společné rysy byly zformulovány v roce 2001 v tzv. *Manifestu agilního vývoje softwaru* [1] organizací Agile Alliance [2]. Sepsání manifestu bylo inspirováno dvěma hlavními myšlenkami:

- umožnit změnu je mnohem efektivnější než se jí snažit zabránit;
- je třeba být připraven na nepředvídatelné události, protože ty stejně nastanou.

Základními body manifestu jsou:

- přednost individualitám a komunikaci před procesy a nástroji;
- přednost funkčnímu SW před obsažnou dokumentací;
- přednost spolupráci se zákazníkem před jednáním o smlouvě;
- přednost reakci na změnu před plněním plánu.

Jinými slovy při dodržení těchto specifik se nemůže stát, že zákazník není spokojen, přestože je smlouva naplněna a obohacena o rozsáhlou dokumentaci.

Obrázek 1: Bajka o praseti a kuřeti

3 SCRUM

Na našem oddělení používáme jednu z agilních metodik známou jako Scrum. Samotné slovo „scrum“ (zkrácenina slova „scrummage“, do češtiny překládáno jako „mlýn“) je převzato ze sportovní terminologie ragby a je to způsob, jak obnovit hru po ztrátě míče. Míče se zmocní ten tým, jehož členové spolu lépe komunikují a který je tedy lépe koordinovaný. Přesně to je principem Scrumu – členové týmu spolu neustále úzce komunikují, práci si společně plánují a jsou pravidelně informováni o tom, jak postupuje projekt jakožto celek, na čem pracují jejich kolegové, předávají si své know-how, a tím se práce týmu výrazně zefektivní.

3.1 Tým

Důležitou vlastností, která odlišuje Scrum od tradičních metodik, je zapojení zákazníka do týmu a rozhodování o prioritách implementovaných funkcionalit.

Tým je rozdělen na dva základní typy rolí – kuřata a prasata. Jejich pojmenování vychází z bajky o kuřeti a praseti, kteří se rozhodnou založit restauraci a přemýšlí, jak ji pojmenují. Kuře přijde s nápadem „Ham&Eggs“, což se praseti nelíbí, neboť kuřete by se proces pouze týkal, kdežto prase by bylo přímo zapojeno. Pokud by restaurace byla firma, pak kuřata budou zákazníci, kteří poskytnou požadavky a prostředky na uskutečnění projektu, prasata potom zaměstnanci firmy, kteří zapojením vlastního úsilí vytvoří produkt.

Konkrétně se role v týmu dělí do následujících:

Mezi kuřata patří

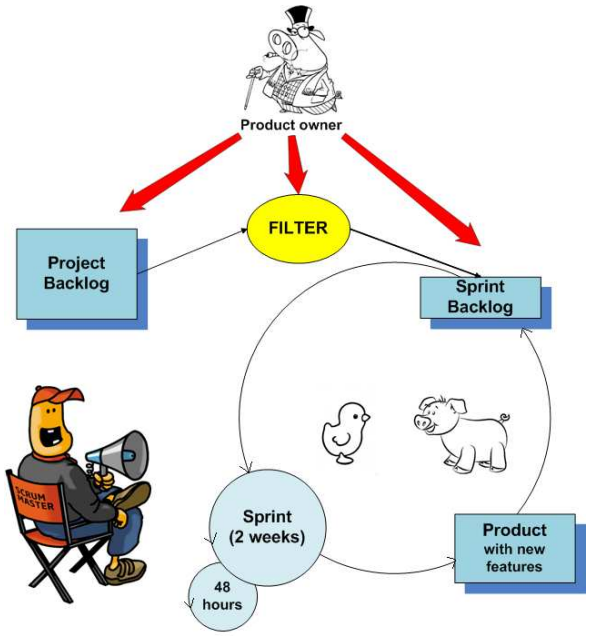
- **Zainteresované strany** (stakeholders) – koncoví uživatelé vyvíjeného systému, investoři, manažeři firmy, která software vyvíjí.

Mezi prasata patří

- **Vlastník produktu** – reprezentuje zájmy zákazníka, určuje priority jednotlivých vlastností produktu, pravidelně je upravuje, přijímá/odmítá výsledky práce;
- **Scrum vedoucí** – osoba, jejímž úkolem je zajistit hladký průběh práce – ochraňuje tým před vnějšími vlivy a udržuje ho koncentrovaný na daný úkol, vynucuje dodržování všech pravidel, organizuje pravidelné schůzky, udržuje přehled o úkolech, které je nutno splnit, odhaluje vzniklé překážky a zahrnuje je do plánu, a v neposlední řadě sleduje osobní problémy a konflikty mezi členy týmu a snaží se vytvořit příjemné pracovní prostředí;
- **Vývojáři** – osoby zodpovědné za vytvoření vlastního produktu, vzájemně úzce spolupracující.

3.2 Scrum proces

Celý Scrum proces probíhá ve striktně vymezených časových intervalech tzv. *sprintech*. Proces začíná sepsáním požadavků zákazníka (vlastnosti produktu) a ohodnocením jejich priorit vlastníkem produktu. Na začátku každého sprintu jsou vybrány požadavky dle priorit a v množství úměrném délce sprintu (obvykle 2–4 týdny). Jak už bylo řečeno v úvodu, výsledkem každého sprintu by měl být provozuschopný



Obrázek 2: Scrum proces

software. V každé další iteraci pak už jen přibývají vlastnosti produktu podle jejich priorit, přičemž funkčnost softwaru zůstává zachována. Je tedy možné zavést verzování tak, že co sprint, to nová verze.

3.3 Artefakty

Stavebními kameny Scrumu jsou tzv. *položky* (Backlog items). Jsou to ony zákazníkem jasně pojmenované a prioritizované vlastnosti, které by měl výsledný produkt mít (např. aplikace umí na dálku zastřežit všechny sledované místnosti; po zastřežení se zelená barva místnosti změní na modrou apod.). Každý projekt má svůj tzv. *Backlog*, což je seznam všech položek, které se týkají daného produktu. Na začátku každého sprintu se z něj pak vytvoří tzv. *Sprint Backlog*, obsahující pouze položky určené pro daný sprint. Backlog je dynamický v závislosti na nově vzniklých podnětech (mohou se měnit priority jednotlivých požadavků, přibývat nové,...). Zde je vidět zmiňovaný rozdíl oproti tradičním metodikám, ve kterých jsou vlastnosti výsledného produktu fixně dané, a jakékoli vyvstalé komplikace je třeba podřídit původnímu plánu, přestože třeba existuje daleko elegantnější a úspornější řešení.

3.4 Schůzky

Všechny schůzky jsou organizovány a moderovány Scrum vedoucím, který by měl zajistit jejich hladký průběh a veškeré výstupy ze schůzek zaznamenávat.

3.5 Plánování sprintu

Plánování je nejdůležitější a nejdelší schůzkou, kterou je zahájen každý sprint. Účastní se jej všechna prasata a možná, ne však nutná, je i účast kuřat. Na této schůzce vlastník produktu představí Backlog a podle priorit spolu se zbytkem týmu vytvoří program nadcházejícího sprintu – Sprint Backlog. Klíčové je vybrané položky důkladně rozplánovat, což zahrnuje:

- každé položce přidělit vlastníka z řad vývojářů, který je zodpovědný za její dokončení;
- rozepsat každou položku do jednotlivých úkolů, tzv. tasků;
- časově každou položku ohodnotit.

Časové ohodnocení je obzvlášť důležitá a obtížná část plánování. Většině z nás se nejednou stalo, že jsme si na nějakou práci vyhradili nadhodnocený čas s tím, že „se to musí zaručeně stihnout“, a po uplynutí oné přehnané doby jsme se divili jak velké množství „se nestihlo“.

Každý člen týmu by měl mít jasno, kolik hodin bude mít během Sprintu k dispozici (odečíst dovolené, svátky, pracovní schůzky,...) a z tohoto času je třeba odečíst ještě 30 % (pro výskyt nečekaných problémů).

Existuje několik metod jak správně plánovat čas určený pro řešení úkolů. Samozřejmě platí: čím méně komplexní úkol, tím snazší je odhadnout jeho časovou náročnost, proto je třeba plánovat dostatečně podrobně, aby byl odhad co nejpřesnější. Nejčastěji používaná metoda pro plánování je tzv. *plánovací poker*, jehož pravidla jsou následující:

Každý hráč (člen týmu) dostane do rukou karty s hodnotami Fibonacciho posloupnosti (1, 2, 3, 5, 8, 13, 21, 34,...), které symbolizují časové úseky (hodiny, dny,...). Stanoví se položka, jejíž časové ohodnocení je cílem kola hry. Vývojář, v dané oblasti nejzkušenější, krátce zbytku týmu představí, o co se jedná, co vše je třeba v dané věci udělat a jaká jsou pravděpodobná

úskalí problému. Tým má příležitost klást dotazy a o problému diskutovat. Následně každý z týmu vybere jednu kartu, reprezentující časový odhad položky, a položí ji lícem na desku stolu. Ve chvíli kdy mají všichni odloženou kartu, karty se obrátí a odhady porovnají. Ti, kteří mají nejextrémnější hodnoty, jsou vyzváni k odůvodnění a je vyvolána nová diskuse. Postup hry se opakuje do chvíle, než se všechny odhady shodují.

Výsledkem této schůzky by tedy měl být konkrétní časově ohodnocený plán, který se tým zaváže splnit. Tím, že se vývojáři podílí na plánování, je i jejich zainteresovanost na výsledku vyšší než při pouhém plnění úkolů nadřízených a jejich práce je předvídatelná a spolehlivá.

3.6 Stand Up

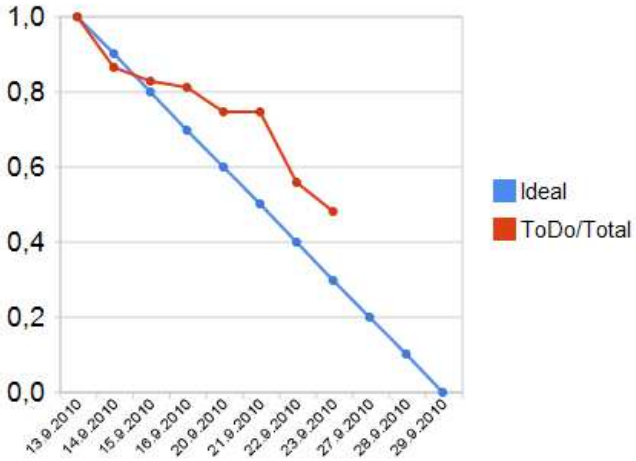
Tým je pravidelně informován o tom, jak postupuje práce všech jeho členů. Tak se děje na (ideálně) každodenní ranní Stand Up schůzce. Název naznačuje charakter takovýchto setkání – všichni přítomní stojí a schůzka netrvá déle než 10 minut. Účast prasat je povinná, účast kuřat možná. Na programu jsou odpovědi všech členů týmu na následující tři otázky:

- Co jsem udělal včera?
- Co udělám dnes?
- Co mě zdržuje v práci? (potřebuji pomoci s laděním programu, rozbil se mi počítač, nemohu sehnat osobu XY, . . .)

Stand Up schůzka není určena k řešení problémů, není to ani sběr informací o tom, kdo je pozadu. Cílem je informovanost týmu o postupu celého projektu, o komplikacích, které objevil jeden člen a mohou ovlivnit práci ostatních. Často je práce jednoho člena vázána na dokončení práce jiného a StandUp schůzky jsou chvíle, kdy se tyto informace předávají. Tým práci sám naplánoval a sám sobě se zodpovídá.

3.7 Review

V den následující konec sprintu proběhne tzv. Review schůzka. Během této schůzky prasata představí kuřatům výsledky své práce (v nějaké formě prezentace). Jak už bylo několikrát zmíněno, software je funkční po každém sprintu a každá nově přidaná vlastnost jeho funkčnost jenom rozšiřuje.



Obrázek 3: Graf Burndown

3.8 Retrospektiva

Po Review následuje poslední schůzka sprintu, tzv. Retrospektiva. Zde se sejdou všechna prasata a společně zhodnotí uplynulý sprint. Všichni členové týmu opět zodpoví několik otázek:

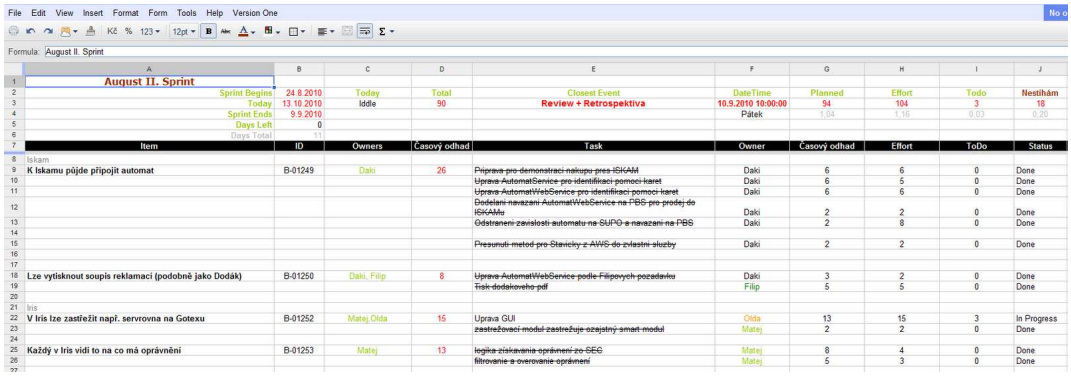
- Co se mi líbilo, co jsme dělali dobře a chci abychom v tom pokračovali?
- Co se mi nelíbilo a chtěl bych změnit?
- Co by se dalo vylepšit nebo zefektivnit?

Smyslem Retrospektivy je subjektivní zhodnocení sprintu členy týmu (nikoli pouze jejich vedoucím), čímž jsou zapojeni do rozhodování o procesu, každý vyjádří svůj názor, poslechne názory ostatních a společně se snaží vyvarovat chyb v příštím sprintu.

3.9 Sledování průběhu Sprintu

Průběh sprintu je zaznamenáván pomocí různých typů grafů. Nejdůležitějším z nich je tzv. *Burndown*, viz obrázek 3.

Na svislé ose je množství práce, kterou je třeba během sprintu udělat (formou procent či hodin), na vodorovné ose čas. Lomenou čárou je vyznačen skutečný průběh sprintu (ToDo/Total), druhá čára znázorňuje ideální průběh sprintu (Ideal), tedy rovnoměrné snižování množství práce. Na tomto grafu je vidět, že během prvních dvou dní tým pracoval rychleji, od třetího dne však nestíhá. To může být způsobeno výskytem nečekaných komplikací, případně špatným



	A	B	C	D	E	F	G	H	I	J
1	August II. Sprint									
2	Sprint Begins	24.8.2010	Today	Total	Closest Event	DateTime	Planned	Effort	Todo	Nestihám
3	Today	13.10.2010	Iddle	90	Review + Retrospektiva	10.9.2010 10:00:00	94	104	3	18
4	Sprint Ends	9.9.2010				Pátek	1.04	1.16	0.03	0.20
5	Days Left	0								
6	Days Total	11								
7	Item	ID	Owners	Časový odhad	Task	Owner	Časový odhad	Effort	ToDo	Status
8	Iskam									
9	K Iskamu půjde připojit automat	B-01249	Daki	26	~~Příprava pro demonstraci nákupu přes ISKAM~~	Daki	6	6	0	Done
10					~~Úprava AutomatService pro identifikaci pomocí karet~~	Daki	6	5	0	Done
11					~~Úprava AutomatWebService pro identifikaci pomocí karet~~	Daki	6	6	0	Done
12					~~Dodělání navázání AutomatWebService na PBS pro prodej do ISKAMu~~	Daki	2	2	0	Done
13					~~Odstranění závislosti automatu na SUPO a navázání na PBS~~	Daki	2	8	0	Done
14										
15					~~Přesunutí metod pro Stavičky z AWS do zvlástní služby~~	Daki	2	2	0	Done
16										
17										
18	Lze vytisknout soupis reklamací (podobně jako Dodák)	B-01250	Daki, Filip	8	~~Úprava AutomatWebService podle Filipových požadavku~~	Daki	3	2	0	Done
19					~~Tisk dodakového pdf~~	Filip	5	5	0	Done
20										
21	Iris									
22	V Iris lze zastřežit např. servrovna na Gotexu	B-01252	Matej,Olda	15	Uprava GUI	Olda	13	15	3	In Progress
23					~~zastřežovací modul zastřežuje ozajstný smart modul~~	Matej	2	2	0	Done
24										
25	Každý v Iris vidí to na co má oprávnění	B-01253	Matej	13	~~logika získavania oprávnení zo SEC~~	Matej	8	4	0	Done
26					~~filtrovanie a overovanie oprávnení~~	Matej	5	3	0	Done

Obrázek 4: Používaný Google Spreadsheet

plánováním (což by při dodržení výše uvedeného postupu nemělo nastat). Pomocí takového grafu lze jedním pohledem zjistit, v jakém stavu projekt je. Samozřejmě to vyžaduje důsledné každodenní zaznamenávání provedené práce do vhodného nástroje.

Pro záznam veškerých artefaktů, statistik, průběhů sprintů, grafů existuje celá škála nástrojů. My máme zkušenosti s produktem VersionOne Enterprise [3] společnosti VersionOne, Inc., který jsme používali rok. Vzhledem k jeho obsáhlosti a finanční náročnosti jsme se rozhodli licenci na další rok nekoupit a používáme nyní námi upravenou formu Google Spreadsheet, viz obr. 4.

4 SCRUM na oddělení OVSS

S implementací Scrumu jsme na našem oddělení začali zhruba před rokem. Počáteční nesmělé kroky zahrnovaly špatně naplánované a především špatně pojmenované položky a nedůsledné dodržování pravidel. Během několika měsíců jsme se však přesunuli k poměrně zaběhnutému režimu, jehož přínos je nezpochybnitelný.

Naše modifikace:

- položky označujeme písmeny M, S, W (Must, Should, Would), které ještě před konkrétním rozplánováním stanoví jejich prioritu.
- Burndown graf aktualizujeme na tabuli na chodbě, takže každý zjistí při příchodu do práce jediným pohledem, v jakém stavu je aktuální sprint.
- Pro záznam podrobností celého sprintu jsme si upravili Google spreadsheet, který je dostupný odkudkoli přes Gmail.
- Vedle Burndown grafu vytváříme i grafy pro sledování časů skutečně strávených při vývoji (ideálně by se měly shodovat s naplánovanými) a stavů položek (nová, rozpracovaná, hotová, nestíhám, nenaplánovaná).
- Vzhledem k vysokému procentu dohodářů v našem týmu pořádáme Stand Up schůzky pouze každý druhý den.
- Detailní rozplánování dílčích částí položek (tasků) je v režii vlastníka položky, který za její dokončení zodpovídá.

Přínos:

- vlastnost, kterou na začátku sprintu naplánujeme, je na konci sprintu hotová;
- pracujeme primárně na věcech, které jsou nejdůležitější;
- vedoucí týmu přesně ví, v jaké fázi produkt aktuálně je a které funkcionality bude na konci sprintu obsahovat;
- každý člen oddělení ví, na čem pracují ostatní a vzniklé problémy společně řeší.

5 Závěr

Scrum není jedinou agilní metodikou, ale vybrali jsme si ji, protože je relativně jednoduchá a její zavedení bylo v našem prostředí rychlé a efektivní. Mezi její hlavní výhody patří zejména úzké propojení uživatele aplikace s vývojovým procesem, krátké a rychlé iterace, které přinášejí zákazníkovi hodnoty průběžně a poměrně často, čímž zvyšují jeho spokojenost.

Nesporným přínosem je Scrum v univerzitním prostředí při zapojení studentů do vývoje. Díky flexibilitě plánování je zaručeno, že je možno

proces přizpůsobit jejich individuálním časovým možnostem – StandUp schůzky zvládnou hravě mezi výukou a v každém sprintu je jim naplánováno pouze tolik hodin, kolik mají skutečně k dispozici. Zejména pro studenty začínající s vývojem, jsou 14denní sprinty ideálním dávkováním práce, a pro jejich nadřízené výbornou možností kontroly.

Literatura

[1] Manifesto for Agile Software Development. http://www.agilemanifesto.org/

[2] Agile Alliance. http://www.agilealliance.org/

[3] Versionone. http://www.versionone.com/

[4] Wikipedia. Scrum (development). http://en.wikipedia.org/wiki/Scrum_(development) □