VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

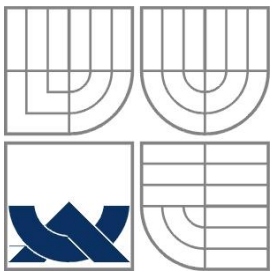
APLIKACE PRO SLEDOVÁNÍ A VYKAZOVÁNÍ VÝKONŮ

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

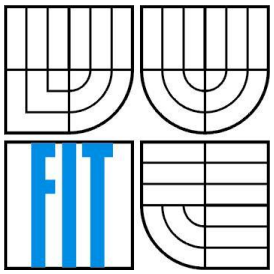
AUTOR PRÁCE
AUTHOR

JIŘÍ ŠLAPÁK

BRNO 2012



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

APLIKACE PRO SLEDOVÁNÍ A VYKAZOVÁNÍ VÝKONŮ

APPLICATION FOR WORKTIME MONITORING AND REPORTING

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

JIŘÍ ŠLAPÁK

VEDOUCÍ PRÁCE
SUPERVISOR

ING. JAKUB MALÝ

BRNO 2012

Abstrakt

Tématem této bakalářské práce je management vývoje softwaru, vykazování a sledování práce vývojářů. Základními ideologiemi pro tuto práci jsou agilní metodiky Scrum, Kanban a jejich kombinace - tzv. Scrumban. Výsledkem bakalářské práce je intuitivní webová aplikace nazvaná Scrumban Board poskytující možnost správy týmů, plánování práce pro týmy a sledování jejich výkonnosti. Druhým výsledkem pak je minimalistická aplikace s názvem Scrumban Client pro operační systém Windows pro jednoduché vykazování práce.

Abstract

This bachelor thesis studies the management of software development, reporting and tracking work of developers. The basic ideologies for this thesis are agile methodologies Scrum and Kanban. The combination of the two is called Scrumban. The final result of this thesis is an intuitive web application called Scrumban Board, which offers a way for team management, planning tasks for teams and tracking efficiency of the development. The second part is a minimalistic application called Scrumban Client for the Windows operation system. This application offers an easy way of tracking work time.

Klíčová slova

Software management, Scrum, Kanban, Scrumban, agilní metodiky vývoje, vykazování práce, měření výkonu, .NET 4.0 Framework

Keywords

Software management, Scrum, Kanban, Scrumban, agile software development, work time reporting, development efficiency tracking, .NET 4.0 Framework

Citace

Šlapák Jiří: Aplikace pro sledování a vykazování výkonů, bakalářská práce, Brno, FIT VUT v Brně, 2012

Aplikace pro sledování a vykazování výkonů

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Ing. Jakuba Malého. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Jiří Šlapák
6.4.2012

Poděkování

Za poskytnuté rady a konzultace bych rád poděkoval vedoucímu práce Ing. Jakubovi Malému. Dále bych rád poděkoval mému nadřízenému Ing. Antonínu Moravcovi, mému bývalému učiteli ze střední školy Mgr. Pavlu Dohnalovi za cenné informace týkající se aplikování nástrojů pro agilní metodiky v praxi a mému vrstevníkovi Martinu Hlaváčovi za doporučení literatury z dané oblasti.

Obsah

1 Úvod

Programování je velice mladá disciplína. Natolik mladá, že celý svět informačních technologií ani netuší, kde jsou limity, kterých dokážeme v této disciplíně dosáhnout. Doslova ještě stále pořádně nevíme, co děláme - už jen získat přesné specifikace od zákazníka je doslova nadlidský úkol, natožpak uspokojit jeho požadavky. Jak s takovými předpoklady lze vlastně řídit lidi pracující v této oblasti? Jak sledovat jejich produktivitu?

Jak si ukážeme v průběhu celé bakalářské práce, tyto dvě otázky jsou neoddělitelně propleteny. Sledování produktivity vždy vychází ze způsobu managementu projektu. Ten totiž jako jediný dokáže poskytnout prostředky, které mohou posloužit pro dostatečně přesná měření produktivity. Jelikož jsou tyto nástroje zároveň součástí managementu projektu, nevyžadují žádnou zbytečnou práci navíc.

Tato práce je zaměřena na použití spolu s metodikami Scrum a Kanban, ze kterých do značné míry i vychází. Proč zrovna tyto agilní metodiky? Jaký je vlastně rozdíl v agilních metodikách oproti klasickým procesům? Krátké iterace vývoje, častý feedback od zákazníka, odlehčení procesů, upřednostňování individuálního přístupu před složitou sadou pravidel. Tuto tématiku podrobně probereme v 2. kapitole.

Podobné zásady, jako jsme jmenovali pro agilní metodiky v minulém sloupci, platí i pro podpůrné nástroje agilního managementu. V kapitole 3 si ukážeme, jak byly tyto zásady zvažovány při návrhu každé části aplikace. 4. kapitola se zabývá technologickými prostředky použitými při vývoji práce a samotnou implementací.

2 Koncept

Jak již bylo naznačeno v úvodu, tato práce je mířená pro používání spolu s agilními metodikami. Pro pochopení celé implementace je klíčové pochopit smýšlení v agilních metodikách.

V první podkapitole shrneme v rychlosti historii managementu softwaru. Druhá část je určena výběru metodik, které se staly teoretickými podklady projektu. Část 2.3 věnována několika termínům společným pro většinu agilních metodik. Ještě než se pustíme do hloubi daných metodik, budeme se věnovat několika připomínkám k existujícím projektům, které se mi podařilo získat od potenciálních uživatelů, případně z literatury.

V podkapitolách 2.6 a 2.7 si podrobně probereme teoretické podklady použité k implementaci a rozebereme pravidla a ideologie Scrumu a Kanbanu. Závěrem si pak ukážeme modelové nástroje, které jsou nejdůležitějšími praktickými podklady pro tento projekt.

2.1 Historie metodik vývoje

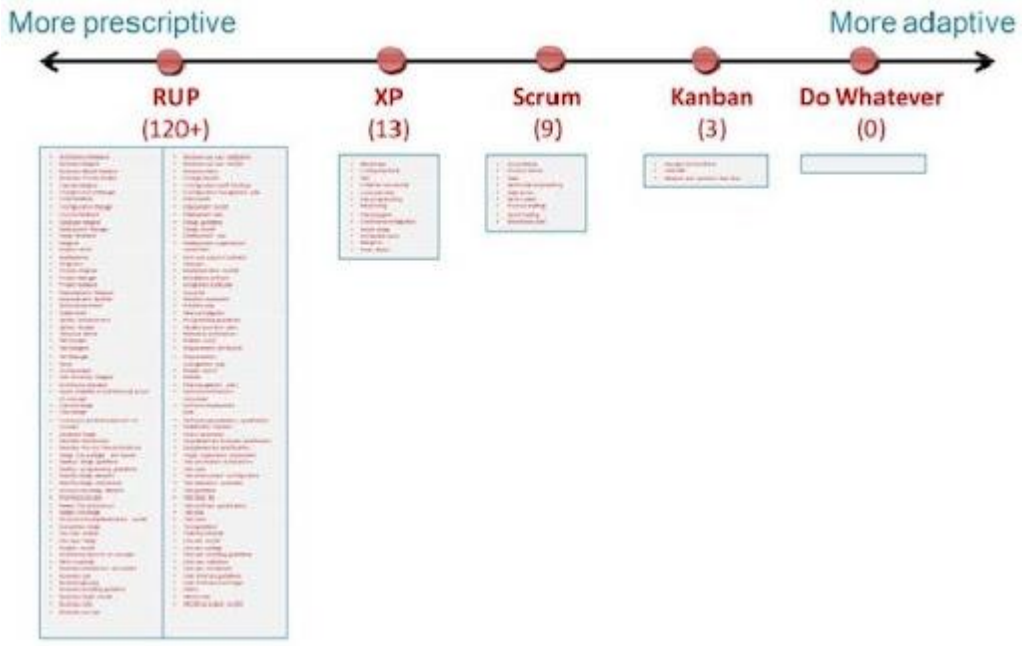
Počátky agilních metodik vývoje se datují zpět do roku 1957, kdy byla použita metodika založená na krátkých iteracích ve firmě Motorola pod vedením Bernie Dimsdala [14]. První oficiální formulace přišla až v roce 1974 [7]. Nezávisle na tom na začátku 70. let Tom Gilb rozvíjel koncept Evolutionary Project Managementu.

Oproti tomu ve světě lineárního přístupu byla v roce 1970 Winston W. Roycem poprvé formálně definována metodika vodopádu [20]. Lineárně iterativní přístup spirálového modelu byl pak popsán v roce 1986 B. Boehmem [5].

Později v 90. letech se začaly vyvíjet tzv. lightweight, neboli lehké metody, v současné době známy jako agilní. Tato vlna přišla jako reakce na těžkopádné restriktivní procesy, jako je metoda vodopádu, nebo Rational Unified Process (1989).

Na obrázku 2.1 je znázorněn počet pravidel, které obsahují jednotlivé metodiky. Čím více pravidel proces obsahuje, tím více je považován za těžkopádný a tím více vlevo na obrázku se nachází, čím méně jich obsahuje, tím je agilnější a přizpůsobitelnější a tím více vpravo je umístěn.

Mezi tyto lighweight metody patří Scrum (1995), Crystal Clear (1996), Extreme Programming (1996), Feature Driven Development (1997), Adaptive Software Development (2000), a mimo jiné nejnovější z nich: Behavior Driven Development (2009).



Obrázek 2.1: Srovnání počtu pravidel metodik vývoje. (RUP – Rational Unified Process, XP – Extreme Progaming, Do Whatever – žádné řízení vývoje) [11]

2.2 Volba cílových metodik

Z minulé kapitoly již tedy víme, že metodik vývoje je spousta. V každém nástroji pro správu vývoje je promítnuta část metodiky, pro kterou je určen. Z toho zákonitě vyplývá, že žádný nástroj nemůže být aplikovatelný na všechny z nich. V této části se pokusíme racionalizovat výběr metodik, které se staly podkladem pro tento projekt.

2.2.1 Klasické procesy

Jeden ze základních důvodů proč zavrhnout klasické procesy je ten, že se mezi nimi špatně hledá společná půda – nelze najít společnou skupinu pravidel vhodnou pro náš. Je tedy téměř nemožné takovýmto nástrojem pokrýt požadavky většího množství zákazníků.

Kdybychom se zaměřili na požadavky konkrétní firmy, bylo by možné definovat skupinu pravidel, kterou by daná společnost požadovala. V takovém případě bychom se však potýkali s nutností změny v našem nástroji při každé změně procesu dané společnosti.

Další důvod je ten, že u modelů rozdělujících vývoj na několik částí má každá fáze své konkrétní požadavky, které se liší od ostatních. Proto je těžké pokrýt celý proces vývoje od začátku až do konce jedním jediným nástrojem.

V neposlední řadě jakýkoli nástroj omezený nadmírou pravidel a regulací je vždy problematické používat. To platí jak ze strany management, tak ze strany členů týmu.

2.2.2 Agilní metodiky

Jelikož mají agilní metodiky výrazně méně pravidel, je jednodušší splnit jejich požadavky na nástroj pro správu vývoje. Podíváme-li se na nástroje v praxi používané, pak je tou nejdůležitější částí vždy grafická pomůcka umožňující:

1) Kdykoli rychle zjistit stav práce na dané části softwaru
2) Jasně znázornit prioritu úkolů
3) Konkretizovat problematiku úkolů v průběhu jejich vývoje
4) Komunikaci v týmu na téma daného úkolu
5) Změnu odhadu náročnosti úkolu ve chvíli, kdy jsou odhaleny detaily onoho úkolu
6) Kdykoli rychle zjistit vytížení týmu

Podíváme-li se pak na počet potenciálních uživatelů takovéto aplikace, zjistíme, že máme daleko větší okruh firem, které by takovýto produkt mohl zajímat. Tento počet navíc neustále roste, jelikož se agilní metodiky stávají stále populárnějšími a minimálně v příštích několika letech tento trend bude pokračovat.

2.2.3 Závěr

Z této kapitoly je jasné, že nástroj pro agilní správu je náš cíl. Ze kterých metodik však vycházet? Grafická pomůcka popsaná v minulé podkapitole je nejlépe definována Kanbanem, což probereme záhy. Kanban sám o sobě však nestačí, jelikož se věnuje pouze plynulosti a průběhu práce týmu (workflow). Tato metodika se však nikterak nevěnuje plánování a managementu (viz. 2.8). Proto bývá často kombinován se Scrumem, který, jak si ukážeme v sekci 2.6, je silně zaměřen na meetingy, plánování a správu týmů.

Spojení Scrumu a Kanbanu je v poslední době velmi oblíbené a názvu Scrumban dal vzniknout Corey Ladas v roce 2008 [13]. Vychází z něj, jak je vidno, i název tohoto projektu.

2.3 Terminologie

Jelikož už víme, že je vhodné, aby aplikace byla vytvořena pro Scrum a Kanban, je potřebné zde uvést základy společné terminologie pro lepší pochopení následujících kapitol.

User Story – Krátký popis funkcionality, kterou má tým vypracovat. Tento popis zahrnuje pozici uživatele, pro kterého je daná funkcionalita vyvíjena, účel ke kterému bude funkcionalita sloužit a co má funkcionalita provádět. Většinou se zapisuje v jednotné formě: „Jakožto <ROLE UŽIVATELE> chci <ABY PROGRAM NĚCO UMĚL>, abych mohl <S NÍM NĚCO PROVÉST>“ [10], [11], [13], [15], [19]. Takovouto jednoduchou formou se dají vyjádřit všechny tři nejklíčovější informace, které

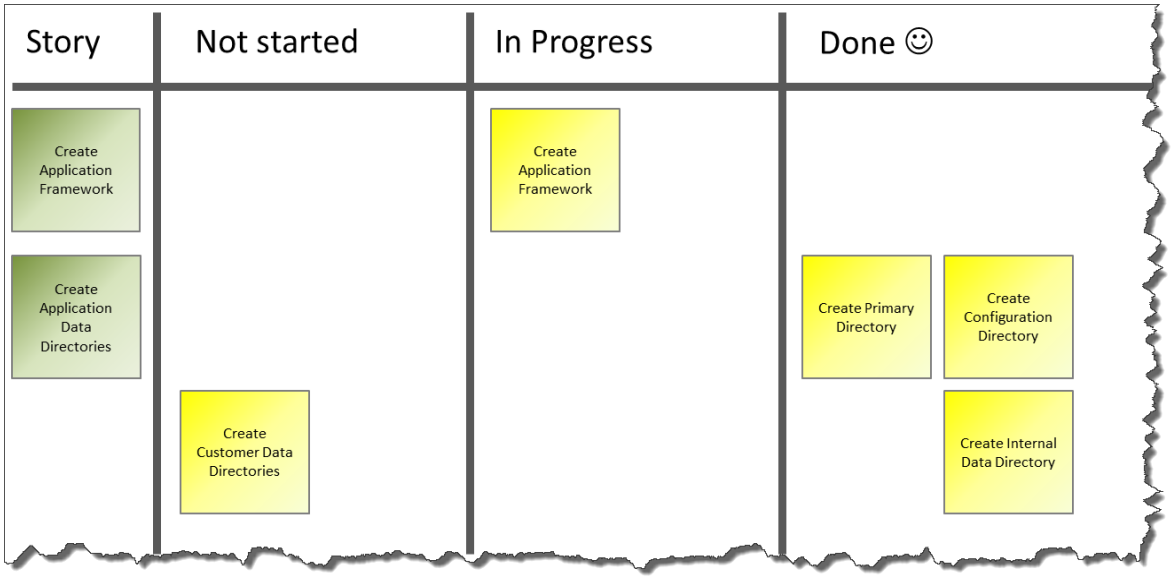
slouží jako základní kámen každé části programu. Všechny tři jsou stejně důležité, a jestliže je kterákoli část špatně formulována, hrozí riziko implementace něčeho jiného, nežli zákazník chtěl.

Story Point – Relativní jednotka pro měření náročnosti funkcionality, kterou má tým vybudovat. V praxi se jí občas říká jinak, ale na jejím názvu nezáleží, jelikož nepředstavuje žádné konkrétní množství času ani jiné reálné jednotky. Slouží čistě k relativnímu porovnávání velikosti user story mezi sebou.

V praxi se používají hodnoty vycházející z upravené Fibonacciho posloupnosti (např.: 0, 1, 2, 3, 5, 8, 13, 21, 40, 100, nekonečno). To zaručuje, že mezi každými dvěma hodnotami je dostatečně velký rozdíl, na základě kterého se dá lépe rozlišovat, nebo přirovnávat velikost.

Story Board – Nejúčinnější doposud nalezený prostředek pro znázornění průběhu práce týmu tzv. **workflow**. Ve zkratce lze říci, že jde o tabuli se sloupci, které vyjadřují stav rozpracovanosti, ve které se daná user story nachází. User Stories bývají nejčastěji na této tabuli reprezentovány jako přilepovací poznámky (tzv. sticky notes). User Story se pak zpravidla pohybuje po takovéto tabuli zleva doprava, přičemž vlevo se akumulují user stories, které tým bude vypracovávat a vpravo user stories, které tým už úspěšně dokončil. Největší výhody této tabule jsou, že je kdykoli vidět, rychle se v ní dá zorientovat, dá se lehce upravovat, a kdokoli okamžitě vidí jak je tým vytížený [10], [11], [13], [15], [19].

V různých metodikách se tomuto nástroji říká různými způsoby. Např.: scrum board, kanban board, task board apod. Všechna tato slova však představují obdobné nástroje jen s drobnými rozdíly. Příklad story boardu je ukázán na obrázku 2.2. Podrobně se budeme této problematice věnovat v kapitole 2.8.



Obrázek 2.2: Ukázka story boardu [22]

2.4 Kontakt s potenciálními zákazníky

Samozřejmě jako u každé obdobné obchodní strategie jsou zde uvedené předpoklady pro úspěch projektu jen pouhé odhady, které nelze podložit reálnými čísly. Navíc odhadovat co zákazník bude od softwaru chtít, bývá tou nejdražší chybou ve vývoji softwaru. Proto bude tato část věnována několika odborníkům z firem, které praktikují agilní vývoj a mohli by o obdobný nástroj mít zájem. Poslední z následujících citací je definice požadavků na takovýto nástroj, které lze nalézt v literatuře zabývající se touto tématikou.

Antonín Moravec (Project manager, Kentico s.r.o., 2012) mluví o informačním systému, který byl vyvinut přímo pro požadavky této konkrétní firmy: *„Náš informační systém je příliš složitý. Umožňuje spoustu nastavení oprávnění, popis detailů u každé user story, má spoustu políček k vyplnění.*

Ve výsledku jej ale používáme tak, že každý má právo na všechno, protože nikdo z managementu nemá nervy na to, aby za ním neustále někdo chodil s požadavky na úpravu. To navíc může zdržovat a prostoje stojí peníze.

Vysoká granularita popisu user stories pak působí jako šum a náš systém selhává ve vyjadřovacích schopnostech podstatných věcí, jako je například priorita, nebo stav user story. Jakékoli plánování se pak stává velmi obtížné.“

Pavel Dohnal (Development director, two bits s.r.o., 2012) mluví o jejich nástrojích: *„Pro sledování vývoje používáme Pivotal Tracker. Pro přípravu user stories před začátkem vývoje pak máme Trello. Je škoda, že musíme používat dva nástroje, jinak jsme s nimi ale vcelku spokojeni. Mají dobrý systém notifikací, nejsou přehnaně složité a oprávnění neřešíme – každý má práva na všechno. Každá změna je stejně vidět v historii.*

Hloupé je snad jen to, že Pivotal Tracker neumí spolupráci několika lidí na jediné user story. Také mi tam chybí množnost rozdělení user stories podle typů – jediné co umí je feature a bug.“

Martin Hlaváč (IT manager, Publero s.r.o., 2012) mluví o prostředcích pro správu vývoje, které používá se svým týmem: *„Pro sledování vývoje používáme GitHub. Nástroj je to jednoduchý, ale je hodně mířený na vývoj a málo na management a plánování. Synchronizaci týmu tento nástroj taky moc nenahrává. Proto si pomáháme nástroji Trello a Basecamp, které jsou pro tento účel určeny. To ale vede k duplicitě, protože všechny user stories musíme mít zapsány jednak v GitHubu a jednak v ostatních dvou systémech. Plánování tím pádem probíhá poměrně nepřehledně a při přenášení dat z jednoho nástroje na druhý vždy hrozí, že se část informací ztratí.*

I přes použití 3 nástrojů je však sledování výkonnosti slabé a nepřesné, což má následky na přesnosti našeho plánování. Navíc tyto nástroje mají špatné notifikační systémy sloužící k upozorňování na změny.“

Clinton Keith definoval požadavky pro nástroj na sledování vývoje takto:
„Pro sledování vývoje potřebujeme metodu, která dělá následující:

- ***Vyjadřuje prioritu a hodnotu dané funkcionality**: Tým se potřebuje zaměřovat na funkcionalitu, která přináší větší hodnotu před funkcionalitou s menší hodnotou.*
- ***Umožňuje a znázorňuje změny**: Změny se budou objevovat a je o nich potřeba diskutovat pravidelně a často.*
- ***Dává prostor pro budoucí komunikaci:** Dokumentace nikdy nedokáže nahradit komunikaci.*
- ***Umožňuje definovat detaily user story s tím, jak tým zjišťuje nové informace:** Neměli bychom vyžadovat všechny detaily implementace předem, nebo vynakládat velké úsilí pro rozšiřování informací o změnách.*
- ***Umožňuje neustálé zpřesňování odhadu náročnosti každé user story v průběhu zjišťování nových informací:** Nejasná zadání nemůžou mít přesné odhady náročnosti. Tato přesnost může narůstat až spolu s uhlazováním zadání.“* [10]

2.5 Scrum

Scrum je model, který byl formálně definován na konci roku 1990, přesto trvalo několik let, než se začal v praxi uplatňovat. V současné době je však bezesporu nejčastěji používanou agilní metodikou vývoje v praxi.

Jádrem Scrumu jsou 2-4 týdenní iterace vývoje. Tyto iterace se nazývají **sprinty.** Na konci každého sprintu musí být vždy produkt v potenciálně publikovatelném stavu. V praxi to znamená, že veškerá práce musí být dokončena, zkontrolována a považována za uzavřenou.

Rychlost týmu je měřena v dokončených story pointech za jeden sprint. To tedy znamená, že prvních několik sprintů tým „naslepo“ odhaduje, kolik story pointů stihne dokončit a teprve posléze získává přesnost v plánování na základě rychlostí v minulých sprintech.

O běh tohoto cyklu se společně starají 3 různé role uživatelů, které si vzápětí popíšeme. Podpůrné techniky Scrumu se rozdělují na ceremonie, neboli schůze, a artefakty, které by se daly popsat jako nástroje pro management.

2.5.1 Role

Scrum Master – Většina literatury jej popisuje jako moderátora diskuse uvnitř týmu i mimo tým. Je to člověk, který svolává porady, řídí běh celého týmu, určuje některá pravidla a stará se o svůj tým.

Nemusí být nutně technicky nejzdatnější. Jeho nejdůležitější rolí, jakožto moderátora je vědět na co se v dané situaci zeptat [10], [15].

Product Owner – Tato role navzdory jejímu názvu neznamená, že jde o manažerskou pozici. Nejlepší definici, kterou jsem kdy slyšel, byla metafora: „Product Owner je snílek. Je to člověk, který celý den polehává na lehátku a sní o tom, jak skvělý by produkt mohl být, co by mohl přinést zákazníkům a jak by je mohl udělat šťastnými.“ [12]. Zjednodušeně řečeno jde o osobu, která zprostředkovává hlas zákazníka, spravuje user stories a určuje jejich prioritu jak v dlouhodobém měřítku, tak pro další sprint.

Člen týmu – Pracovník bez ohledu na zaměření. Patří sem programátoři, testeři, technical leadeři, analytici, designeři a každý, kdo spolupracuje s týmem vstříc jejich cíli.

2.5.2 Artefakty

Product Backlog je prioritizovaný seznam všech user stories projektu, které se mají v budoucnu vypracovat. Jedná se o plánované user story, které se po dávkách vypracovávají v jednotlivých sprintech. Správa product backlogu je jedna z hlavních úloh product ownera. Před plánováním každého sprintu je vždy nezbytně nutné, aby product owner tento seznam aktualizoval a seřadil dle priority. User stories jsou přidávané product ownerem, nebo jsou vráceny ze sprint backlogu, pokud je tým nestihl vypracovat v daném sprintu.

Sprint Backlog je seznam user stories konkrétního sprintu. User stories sem vstupují na základě plánovacích porad, kdy celý tým spolu s product ownerem a scrum masterem rozhoduje, které user stories se budou vypracovávat v rámci následujícího sprintu. Vývojáři odebírají user stories ze Sprint Backlogu a postupně je vypracovávají. Po dokončení je user story přesunut mezi hotové. Pokud tým user story nestihne v době sprintu, tak je přesunut zpět do product backlogu a musí projít celým procesem plánování znovu.

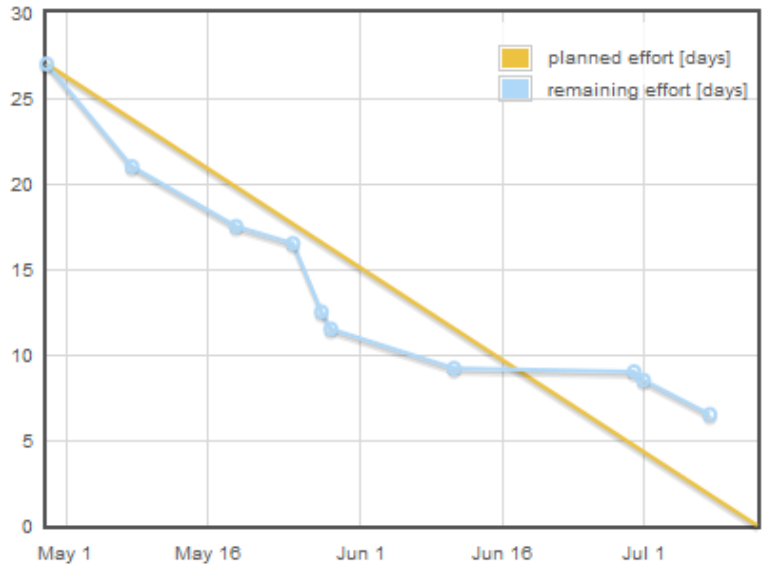
Scrum Board je jedna z nejdůležitějších pomůcek pro vnitřní běh týmu. Doslova jde o tabuli stejného typu, jako jsme zmínili již dříve (viz. obrázek 2.2). Jelikož se jedná o velice variabilní nástroj, kterou si každý tým buduje vstříc vlastním potřebám, tak nemá cenu ji zde teoreticky rozebírat blíže, než jsme to udělali v kapitole 2.4. Z praktického hlediska se na tuto tématiku pak podíváme v kapitole věnující se kanban boardu, který je pro naši aplikaci zajímavější.

Burndown Chart je graf znázorňující rychlost zpracovávání user stories. Tento graf se používá k vizualizaci toho, jestli stíháme svoji práci nad plán, podle plánu, nebo v plnění plánu zaostáváme.

Burndown Chart je pomůcka využívaná na dvou vrstvách vývoje: První vrstvou je tým, který sleduje burndown chart pro daný sprint. Tzn. slouží mu pro sebekontrolu toho, jestli stíhají udělat, co slíbili při plánování sprintu. Druhou vrstvou je pak management, který podle tohoto grafu sleduje stejné věci jako tým, jen v rozměrech celého produktu, případně současné verze produktu.

Burndown Chartů je několik druhů, které se mezi sebou liší např. tím, jak znázorňují nově přibývající user stories. Za zmínku zde stojí minimálně tzv. „Alternative Burndown Chart“, nebo „Kumulativní Burndown Chart“. Ke druhému z nich se však vrátíme v kapitole 2.7, jelikož pro Kanban je velmi zásadní.

Ukázku základního burndown chartu si můžeme prohlédnout na grafu 2.3. Osa X je osou časovou. Osa Y grafu znázorňuje počet story pointů, které má tým stihnout zpracovat. Žlutá čára je ideální rychlost vypracovávání user stories. Modrá znázorňuje reálný průběh uzavírání práce. Ve chvíli, kdy modrá čára protne nulu na ose Y, má tým hotovou veškerou svou práci. Momentu, kdy žlutá čára protne nulu na ose Y, se říká deadline – termín do kdy má být naplánovaná práce hotova.



Obrázek 2.3: Základní burndown chart.[21]

2.5.3 Ceremonie

Daily Scrum je krátká každodenní porada trvající do patnácti minut. Pokud je tato doba přesažena, porada končí bez ohledu na to, jestli se stihli vyjádřit všichni zúčastnění. Porady se účastní scrum master a členové týmu.

Cílem porady je sledovat každodenní vývoj, synchronizovat tým a adresovat aktuální problémy co nejdříve po jejich vzniku. Účelem této porady není řešit konkrétní problémy, jen na ně upozornit a zajistit, že o nich ví celý tým [10], [11].

Sprint Review je porada probíhající na konci každého sprintu. Účastní se jí product owner, scrum master a všichni členové týmu. Časově omezená tato porada nijak není – je ovlivněná velikostí týmu a délkou sprintů.

Účelem této porady je uzavření sprintu a schválení a uzavření práce týmu za poslední sprint. Na této poradě se product owner rozhoduje, jestli hotová funkcionalita odpovídá jeho představám. V případě že neodpovídá, user story je uzavřena a je vytvořen nový úkol na předělání, nebo dodělání funkcionality. V rámci porady se také řeší nedodělané user stories a rychlost týmu. Všechny nedodělané user stories jsou přesunuty ze sprint backlogu zpět do product backlogu.

Důležitou rolí této porady z pohledu managementu je možnost prohlédnout si odvedenou práci a vznést k ní připomínky.

Sprint Retrospective se koná mezi sprinty, většinou ihned po sprint review. Účastní se jí product owner, scrum master a všichni členové týmu. Jedná se o poradu, kde se tým autoevalvuje.

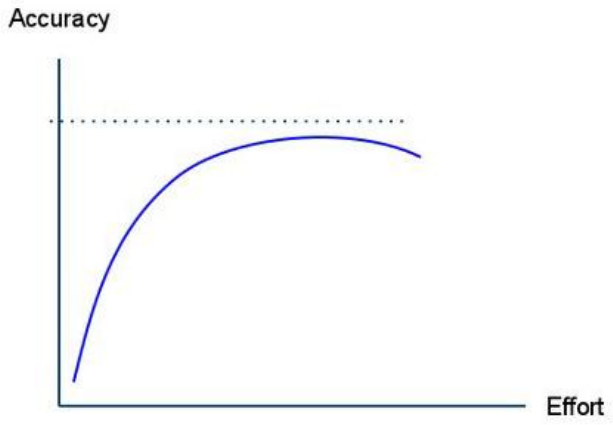
Účelem této porady je tedy zhodnocení minulého sprintu – probírání problémů a způsobů, jak jim v budoucnu předejít a naopak zhodnocení toho, co fungovalo dobře a jak toho využít do budoucna. Nejdůležitějším důsledkem této porady je neustálé zlepšování efektivity týmu.

Zpravidla bývá sprint retrospective omezena maximální délkou, která se odvíjí od délky sprintu a počtu vývojářů. Toto omezení je stanoveno zpravila proto, že po určitém čase se na ní začíná zabředávat do méně podstatných záležitostí.

Sprint Planning se koná těsně před začátkem sprintu. Účastní se jí product owner, scrum master a všichni členové týmu, nebo lidé kteří budou s týmem spolupracovat na příštím sprintu.

V rámci plánování se berou user stories z product backlogu a naplní se jimi sprint backlog. To znamená, že jsou do sprint backlogu umístěny user stories, jejichž součet story pointů odpovídá rychlosti týmu (viz. počátek kapitoly o Scrumu). Časovou náročnost úloh odhadují členové týmu. Ti se vyjadřují v relativních jednotkách (story pointech) ohledně toho, jak dlouho bude trvat zpracování user stories. Takovéto odhadování je zpravidla kolektivní práce, při které by členové týmu měli probírat, co daná funkcionalita obnáší. Nemělo by ale docházet k tomu, že se členové týmu budou navzájem ovlivňovat tím, že předčasně zveřejní svůj názor na náročnost.

Kvalita odhadu záleží na zkušenostech vývojáře a na době věnované rozebírání dané funkcionality. Pro představu vztah mezi úsilím (tzn. i časem) vynaloženým pro přesný odhad a dosaženou přesností odhadu je vyjádřen grafem 2.4.



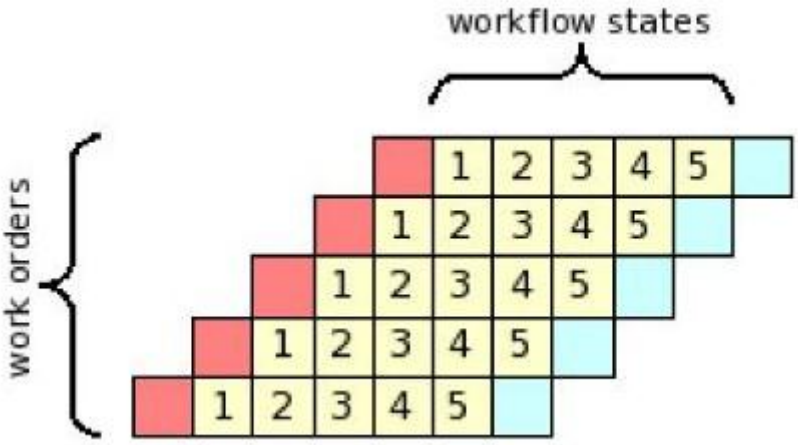
Obrázek 2.4: Přesnost odhadu náročnosti user story (osa Y) ve vztahu k úsilí, které bylo věnováno přesnému odhadu (osa X) [6].

Ačkoli se může zdát, že sprint review, sprint restrospective a sprint planning se konají najednou, je nutné jejich rozdělení. Co více, je lepší mezi nimi nechat časový interval, který dá lidem prostor pro zamyšlení a přípravu na následující kroky.

2.6 Kanban

Kanban, jakožto model, se zaměřuje čistě na optimalizaci workflow. Celý proces se dá shrnout jako přizpůsobení principu výrobní linky do prostředí softwarového inženýrství. To tedy znamená, že oproti Scrumu a většině ostatních agilních metodik nemá žádné iterace. Jde o kontinuální proces, do kterého z jedné pomyslné strany přicházejí požadavky a z druhé strany odchází vypracovaná a uzavřená funkcionalita připravená pro vydání.

Oproti klasické výrobní lince Kanban samozřejmě disponuje řadou odlišností, které jsou vynucené specifickým prostředím informatiky. Místo vykonávání práce na konkrétním kusu výrobku tým pracuje na user stories. Aby běh těchto user stories byl co nejplynulejší, bývají user stories voleny tak, aby všechny byly v průměru stejně náročné. V takovémto případě odpadá odhadování náročnosti a potřeba story pointů a je aplikovatelný model z obrázku 2.5. Na tomto obrázku je znázorněna ideální plynulost workflow, ve kterém každá user story prochází pěti stavy. Červený čtverec pak znázorňuje příchod nové user story do systému a modrý znázorňuje výstup hotové funkcionality.



Obrázek 2.5: Ideální případ navazování práce v Kanbanu.

Není sice vždy nutné, aby každá user story měla stejnou velikost, je to však ideální případ. Pokud by se nám toho však podařilo dosáhnout, dokázali bychom přesně rozdělovat práci, přesně určit za jak dlouho bude požadovaná funkcionalita dokončena a po naplnění systému bude po každém taktu vnitřního metronomu týmu dokončena jedna user story. Jelikož jde o ideální případ, kterého v praxi téměř nelze dosáhnout, Kanban použití story pointů nezakazuje a tyto se mohou stát doplňkem, který může zjednodušit synchronizaci týmu.

2.6.1 Role

Kanban oficiálně žádné role nevynucuje, nicméně z logických důvodů musí být vždy nastoleny minimálně dvě role v systému:

1) Role starající se o zadávání user stories týmu.
2) Členové týmu pracující na user stories.

Opět zde však platí to, že Kanban je velmi volná platforma, tudíž role si může každý tým definovat dle vlastních potřeb.

2.6.2 Work in progress limit

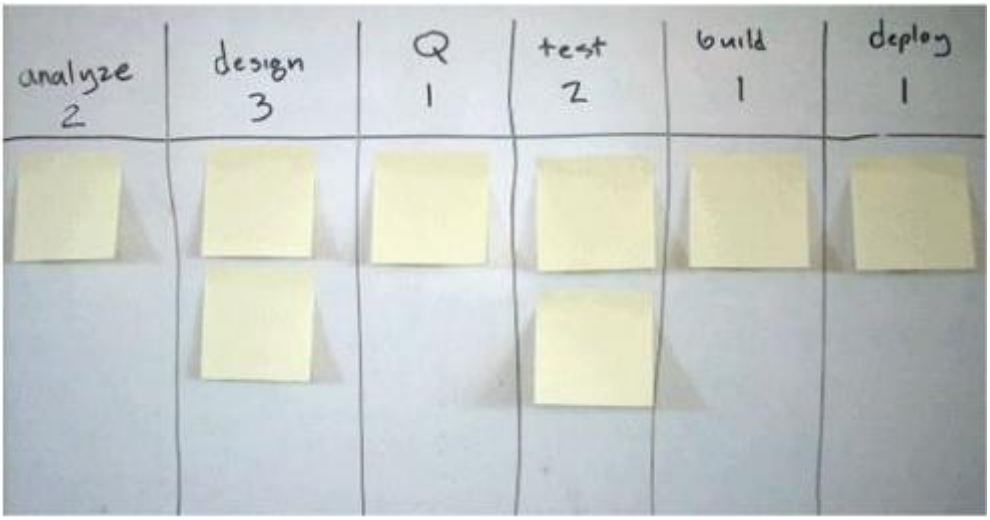
Work in progress limit, jak napovídá název, omezuje počet user stories, které jsou v danou chvíli rozpracovány.

I když to nebylo dříve zmíněno, tak Scrum má tento limit skrytě definován tím, že se každý sprint naplní dle dlouhodobě měřené rychlosti týmu. To tedy znamená, že Scrum omezuje work in progress limit pro daný sprint. Jelikož Kanban žádné cykly vývoje nemá, musí se množství momentálně rozpracovaných úkolů limitovat jinde. Tímto místem je stav, kterým úkoly prochází, neboli konkrétní sloupec v kanban boardu.

2.6.3 Kanban Board

Kanban Board je pomůcka obdobná dříve popsanému story boardu. Má však několik specifických vlastností, které jsou pro návrh naší aplikace velmi užitečné. Jelikož tento druh story boardu již může být komplexnější než dříve zmíněné, považuji za nejlepší ukázat zde vždy příklad a popsat jeho části a způsob použití.

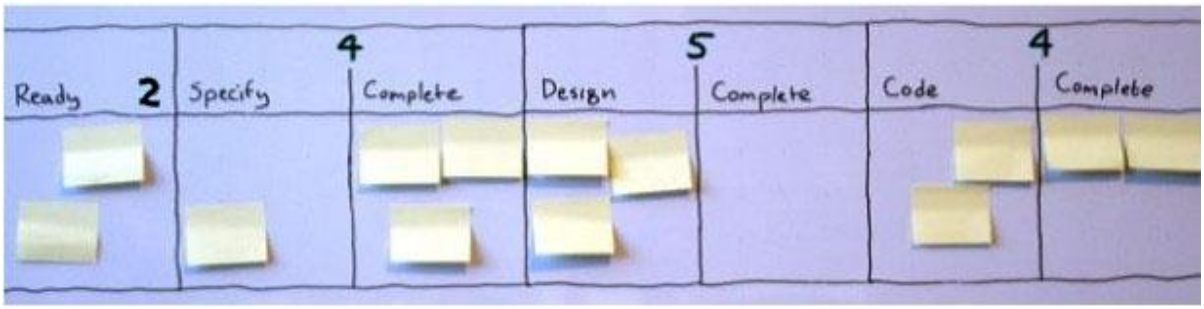
Na obrázku 2.6 je ukázka základního kanban boardu, který má pro každý stav práce určení work in progress limit (číslo pod názvem sloupce). Jak můžeme vidět, tak mimo sloupce „design“ a „analyze“ jsou všechny ostatní zaplněny.



Obrázek 2.6: Základní kanban board [13].

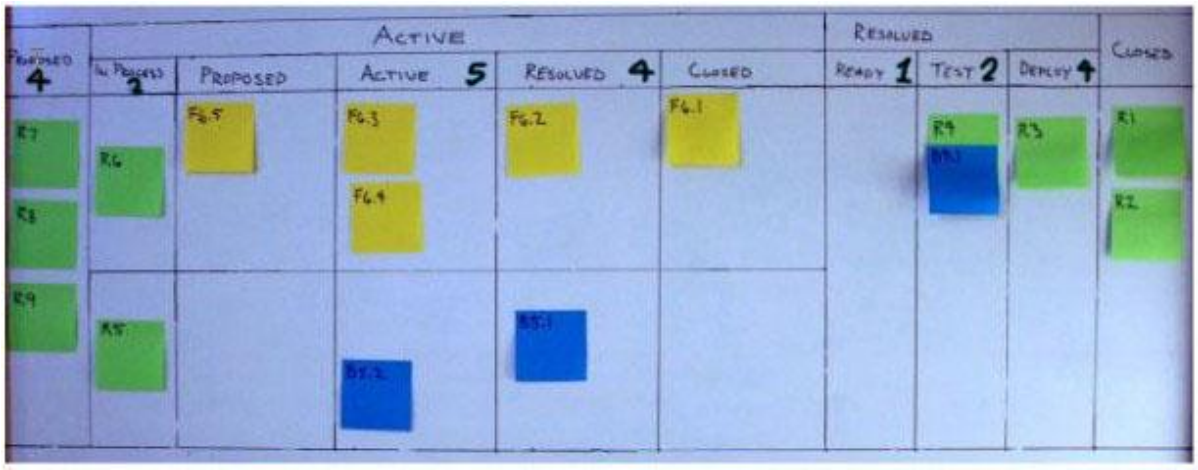
Na následujícím příkladu (obrázek 2.7) můžeme vidět, že lze definovat work in progress limit pro několik sloupců společně. Je také povšimnutí hodné, že každý sloupec má svůj zásobník pro user stories, u kterých daná část vývoje již byla dokončena.

Tato technika se používá pro lepší visuální odlišení user stories, na nichž právě probíhá práce daného druhu od těch, které jsou hotové a mohou být posunuty dále ve směru workflow. To navíc podporuje myšlenku „pull“ přístupu k přidělování práce, kdy každý pracovník starající se o následující sloupec si sám odebírá úkoly z předchozího sloupce, místo toho, aby mu je tam vnucoval někdo jiný (čemuž se pak říká „push“ přístup).



Obrázek 2.7: Kanban Board se zásobníky pro hotové úkoly a společnou definicí work in progress limitu pro několik sloupců [13].

Obrázek 2.8 poukazuje na možnost několika prioritizovaných front. V tomto příkladu je spodní část dedikovaná pouze pro úkoly s nejvyšší prioritou, jako mohou být například bugy, nebo defekty. Jestliže se objeví v prioritní frontě úkol, bývá okamžitě přerušena veškerá práce a všichni se soustředí pouze na tento prioritní úkol. Takovýto zásah sice působí dosti brutálním dojmem, nicméně v některých situacích je potřebný.



Obrázek 2.8: Kanban board s prioritní frontou (spodní část s modrými user stories) [13].

2.6.4 Kumulativní flow chart diagram

Tento diagram je podobný burndown chartu tak, jak jsme si jej popsali u Scrumu, je však obohacen o možnost sledování slabých míst týmu.

Na obrázku 2.9 lze vidět příklad takovéhoto grafu, kde osa X je osou časovou a osa Y je osou, která znázorňuje story pointy v systému. První, čeho si lze všimnout je to, že všechny sloupce kanban boardu od backlogu až po sloupec pro hotové úkoly zde mají jednu vrstvu. Graf tedy zachycuje počet úkolů/story pointů, který daným sloupcem celkově prošlo. Rozdíl oproti nižší vrstvě pak určuje kolik úkolů/story pointů je v danou chvíli v onom sloupci.

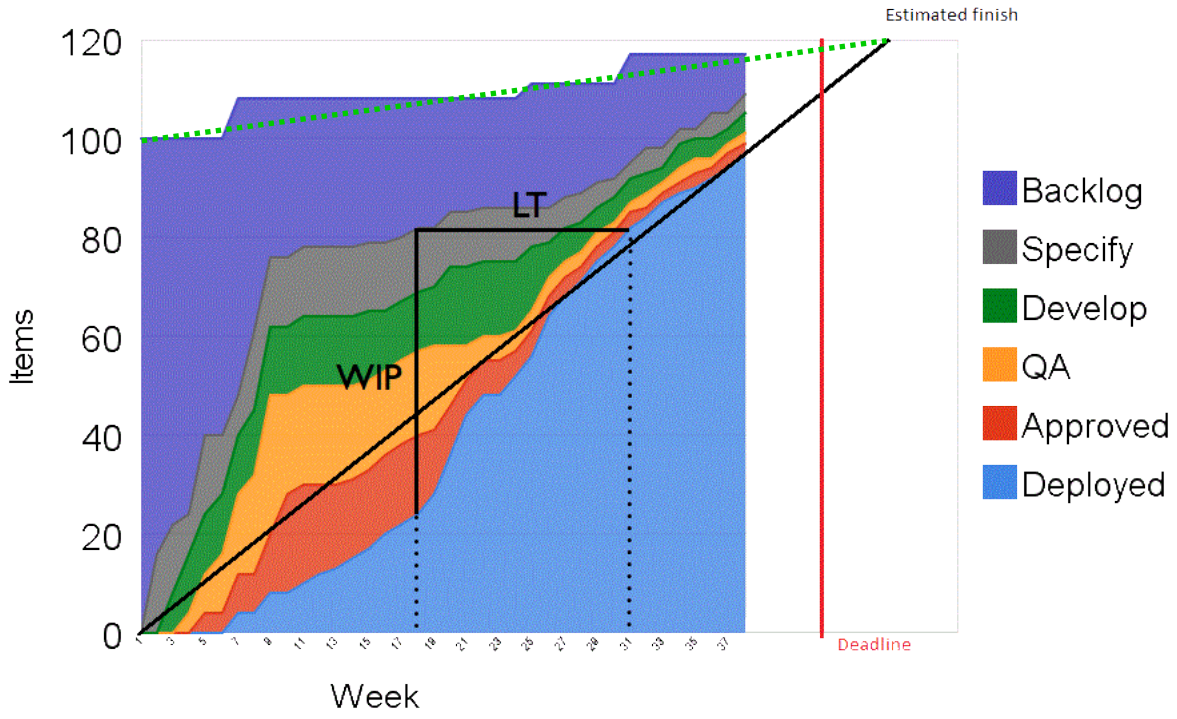
Jestliže v kterémkoli bodě osy X povedeme svislici, a zaznačíme její průsečíky s vrstvou pro backlog a pro hotové user stories. Délka takto vzniklé úsečky značí celkové množství úkolů/story pointů, na kterých bylo v danou chvíli pracováno (v grafu jako úsečka WIP).

Zopakujeme-li stejný postup, jaký byl popsán v minulém odstavci pouze s horizontální přímkou vedenou z bodu Y, dostaneme okamžitý čas potřebný pro vyvinutí dané funkcionality (v grafu jak úsečka LT). Jestliže takto získané hodnoty zprůměrujeme, získáme průměrný čas potřebný k vypracování průměrné user story. Z této úsečky lze také vyčíst, která část vývoje trvala v daném období nejdéle. Např. z úsečky LT na grafu je zřejmé, že nejdéle trvalo vyspecifikovat danou funkcionalitu (šedá vrstva grafu). Pokud by se to opakovalo, bylo by vhodné se zaměřit na tuto část vývoje a její optimalizaci.

Průměrná rychlost práce týmu je v grafu znázorněna černou přímkou, která protíná střed souřadnic a poslední zaznamenaný bod na vrstvě pro dokončené user stories.

Počet user stories v průběhu vývoje může narůstat. To je v grafu znázorněno narůstající hodnotou vrstvy backlogu. Rychlost tohoto růstu je vyznačena přerušovanou zelenou čarou.

Odhadování času potřebného pro dokončení vývoje probíhá obdobně, jako u burndown chartu, pouze místo odčítání od naplánovaných story pointů vstříc nule, sledujeme průsečík přímky růstu celkového množství story pointů projektu/verze projektu s rychlostí práce (viz. na grafu bod „Estimated finish“). Deadline, neboli plánovaný konec vývoje je zakreslena jako červená svislá čára.



Obrázek 2.9: Kumulativní flow chart pro Kanban. S drobnými úpravami citováno z [9].

3 Architektura aplikace

Kapitola 2 nám odhalila velkou spoustu nástrojů využívaných v agilních metodikách. Samozřejmě dlouhodobým cílem naší aplikace je poskytnout svým zákazníkům všechny tyto nástroje. V rámci bakalářské práce bylo však nereálné implementovat veškerou tuto funkcionalitu.

Tato kapitola popisuje stávající funkcionalitu těch nejzásadnějších z nástrojů. Popis způsobu používání však není její prioritou. Tyto informace včetně obrázkových návodů lze nalézt v záložce „FAQ“ ve webové aplikaci, která je standartní součástí instalace.

3.1 Webová aplikace

Hlavní součástí bakalářské práce je webová aplikace, která slouží k administrativě našeho informačního systému. To tedy znamená, že po dokončení vývoje tohoto projektu bude webová část obsahovat veškeré nástroje pro správu a plánování, které byly uvedeny v předchozí kapitole.

Tato kapitola je věnována tomu, co bylo zpracováno v rámci bakalářské práce, co bude následovat v dalších iteracích vývoje tohoto projektu a konkrétním odlišnostem oproti vzorovým řešením z kapitoly 2.

Webová aplikace byla zvolena z důvodu jednoduchého ovládání z kteréhokoli zařízení bez ohledu na platformu a operační systém jen s pomocí internetového prohlížeče.

Momentálně webová část podporuje tyto primární prohlížeče v nejnovější verzi:

1) Firefox
2) Chrome
3) Opera
4) Safari

Další prohlížeče, které jsou podporovány mimo drobné vzhledové odlišnosti:

1) Internet Explorer 10
2) Internet Explorer 9
3) Internet Explorer 8

3.1.1 Prerekvizity a instalace

Prerekvizitami provozování webové aplikace jsou: .NET Framework 4.0, běžící služba SQL Express server a nainstalovaný IIS server 7.0 se zpětnou kompatibilitou pro verzi 6.0. Zpětná kompatibilita je nutná kvůli instalátoru. Instalátor pro verzi 7.0 nebyl v době vývoje ještě k dispozici a tak je prozatím tento požadavek bohužel standartní pro spoustu obdobných projektů. Poslední věcí pro správnou

funkci je nainstalovaná podpora Windows Comunication Foundation v IIS serveru. Ta je využívána např. v klientské aplikaci, nebo při editaci každého story boardu.

V rámci instalace je kontrolována verze rozhraní .NET Frameworku zvoleného fondu aplikací. Pokud je verze starší, nežli 4.0, pokusí se program nalézt vhodný, nebo vytvořit nový fond aplikací. Taktéž se pokouší přidělit uživateli, který je využíván SQL serverem a uživateli fondu aplikací IIS serveru oprávnění pro čtení a editaci adresáře do kterého je software instalován. Toto zaručuje uživatelskou přívětivost a omezuje nutnost zásahů z uživatelské strany ihned po instalaci. Nevýhodou však je nutnost přidělení administrátorských oprávnění instalačnímu programu.

Detailní popis instalace a případných problémů lze nalézt v příloze 1 „Scrumban Board Installation Guide.pdf“.

3.1.2 Správa uživatelů

Role uživatelů jsme použili z metodiky Scrumu, obohacené o administrátorskou roli. To tedy znamená, že máme následující druhy uživatelů:

1) **Člen týmu**, který se stará o plnění úkolů. Jeho nejdůležitější starostí je produkovat pohyb na story boardu svého týmu a obohacování user stories o doplňující informace spolu s tím, jak sám prohlubuje své znalosti v dané oblasti. Mimo tyto dvě zodpovědnosti má v současné verzi možnost pouze sledovat kdo z jeho týmu je momentálně aktivní.
2) **Scrum Master** je jediná role, která je primárně určená k plánování. Má tedy u story boardu každého týmu možnost kdykoli začít proces plánování. Dodržování cyklů, ceremonií atp. aplikace nesleduje a ponecháva je v rukou scrum masterů. Tato role již má navíc právo upravovat uživatelské účty, spravovat týmy a jejich story boardy.
3) **Product Owner** je role primárně určená k zadávání user stories do pracovního procesu a volbu jejich priorit. Proto má svou unikátní možnost sledovat a spravovat backlogy jednotlivých týmu i backlog celého produktu. Stejně tak jako scrum master má product owner právo pro editaci uživatelů, týmů a story boardů.
4) **Administrátor** je uživatelské oprávnění umožňující vše výše zmíněné.

Jelikož u firemního softwaru nechceme, aby se uživatelé mohli registrovat sami do systému, je tato zodpovědnost čistě na autorizovaných uživatelích, kteří mají práva pro editaci uživatelských účtů. Ze stejného důvodu nemá člen týmu možnost editovat svůj vlastní profil s výjimkou změny profilového obrázku a hesla.

Pro správu profilových obrázků uživatele byla použita aplikace tzv. „**gravatar**“ (viz.: [1]). Tato služba se stará o to, aby každý uživatel ihned po registraci získal automaticky generovaný profilový obrázek, který si může jednoduše změnit. Takovýto profilový obrázek je pevně svázán s e-mailovou adresou. To znamená, že ve všech ostatních službách, které využívají gravatar bude mít uživatel

shodný profilový obrázek a nebude jej muset tedy nastavovat vícekrát. Příklady automaticky generovaných gravatarů si lze prohlédnout na obrázcích 3.1 a 3.2.

Jelikož tato aplikace nebyla nikdy zamýšlena jako interní stránky obsahující detailní uživatelské informace, jsou tyto informace velmi omezeny a všude se předpokládá s identifikací uživatele podle gravataru a přihlašovacího jména, které se zobrazí v tooltipech gravatarů.

Při jakékoli změně v uživatelském účtu je také na e-mail daného uživatele odeslán notifikační e-mail s informacemi o změně.

3.1.3 Správa týmů

V rámci jednotnosti byl kladen důraz na obdobné zpracování editace týmu, jako je story board. Uživatelé se tedy přiřazují týmům pomocí tažení myší. V případě potřeby přiřazení jednoho uživatele do více týmu je zde možnost definovat toto chování v editaci profilu daného uživatele. V dalším vývoji projektu je v plánu umožnit přiřazení jednoho pracovníka do více týmů i tažením myší.

3.1.4 Správa týmových story boardů

V této části aplikace lze definovat pro kterýkoli tým vlastní story board včetně popisků sloupců a work in progress limitů. Po vytvoření týmu má každý tým přednastaveny 4 základní sloupce:

1) Team Backlog
2) Sprint Backlog
3) In progress
4) Done
5) Closed

Tyto jsou minimální sadou sloupců, které tým potřebuje. Team Backlog, Sprint Backlog, Done a Closed se nedají smazat a mají na tabuli pevně určené místo – nelze je přesouvat. U ostatních platí, že změna pořadí se provádí tažením sloupce myší na jinou pozici.

3.1.5 Sledování výkonnosti

V projektu byla vypracována první verze zobrazující dlouhodobé statistiky pracovníků rozřazených podle týmu. Tato funkcionalita je založená na porovnávání souhrnného času stráveného prací se součtem story pointů, na kterých člen týmu pracoval ve zvoleném období. Tímto způsobem zle snadno porovnávat pracovníky mezi sebou. Praktickou ukázku takovéto krátké statistiky znázorňuje obrázek 3.1.

Při používání nástroje pro sledování výkonnosti je vždy třeba brát v potaz to, že porovnávání mezi týmy bude zkresleno relativní velikostí story pointu, které má každý tým unikátní. Odlišné velikosti story pointů je dána tím, že každý tým náročnost odhaduje odlišně.

V praxi je však vždy přednější potřeba týmu získávat přesnější odhad náročnosti a optimalizace na úrovni jednotlivých lidí uvnitř týmu, nežli porovnávání pracovníků z odlišných týmů. Následky, které by s sebou nesl tradiční způsob, kdy náročnost odhaduje jeden jediný člověk, by vedly k daleko méně přesným odhadům náročnosti. Od toho se odvíjí nepřesné plánování, zhoršení běhu týmu a v neposlední řadě by při velmi nepřesných odhadech nebylo možné porovnávat výkonnost lidí ani uvnitř týmu, ani mezi nimi.

Dále je potřeba brát v potaz to, že user stories, které nebyly uzavřeny, se do tohoto výpočtu nepočítají. Toto má svou logiku pevně zakotvenou v metodikách, které jsme si zvolili za vzor. Tyto metodiky shodně říkají: „Co nebylo uzavřeno, jako by nebylo ani započato.“ [6], [10].

Informace o odpracovaném čase je sbírán pomocí klientské aplikace, proto se k tomuto tématu vrátíme v kapitole 3.2.

V dalších iteracích vývoje bude jedním z hlavních objektů zájmu tato oblast, jelikož pro vyšší úroveň managementu nám chybí burndown chart a kumulativní flow chart.

Obrázek 3.1: Statistika výkonnosti člena týmu s detailním rozpisem user stories.

3.1.6 Plánování

Plánování je v této aplikaci nástroj, který v analogii se Scrumem slouží pro sprint review a sprint planning. Znamená to tedy, že v této části aplikace se user stories jak uzavírají, tak plánují pro příští iteraci, případně u Kanbanu jsou odeslány rovnou do vývoje.

Sekce pro plánování obsahuje mimo pevně definované sloupce u každého týmu (Team Backlog, Sprint Backlog, Done a Closed) i uměle vytvořený sloupec sloužící jako souhrn veškeré rozdělané práce. Tento sloupec se jmenuje „Not finished“. Tato funkcionalita je podložena předpokladem, že pro potřeby plánování není nutná vysoká granularita stádia rozpracovanosti user stories. Podstatná je pouze informace, že se na nich pracuje a ještě nebyly dokončeny.

Ovládání a vzhled je opět obdobné, jako u story boardu – z user stories se manipuluje pomocí tažení myší. Tento nástroj také umožňuje plnou kontrolu nad user stories a možnost jejich plné editace.

Z důvodů úspory místa se zde dříve uzavřené user stories již nezobrazují, jelikož pro tento nástroj nejsou relevantní. Tyto úkoly však zůstávají v databázi a v dalších iteracích vývoje by bylo vhodné udělat nástroj pro vyhledávání v uzavřených user stories, protože tyto mohou být cenným zdrojem informací pro tým.

3.1.7 Editace user stories

Základní myšlenkou je, že každá user story je editovatelná odkudkoli a kýmkoli. Takovýmto smýšlením vycházíme z toho, že naši kolegové pracují s námi vstříc našemu cíli a nebudou nám zbytečně přidělávat problémy. Naopak budou mít snahu nám pomáhat, a pokud uvidí problém, se kterým by nám mohli pomoci, poskytnou nám své cenné informace tím, že se k dané user story budou moci vyjádřit.

Každá user story má následující části:

1) **Project name**: Tato kolonka byla vytvořena pro krátký název skupiny, do které user story patří, případně pro její název. Tento návrh bohužel nepodporuje žádná literatura, nicméně je to prostředek nezanedbatelné důležitosti pro komunikaci mezi lidmi zainteresovanými v dané user story - je jednodušší pojmenovávat user story několika málo slovy, nežli vyprávět ji celou. Tato kolonka je vždy tučně uvedena u dané user story ve story boardu.
2) **User story**: Tato textová oblast je určena k zapsání user story. Jelikož je user story velmi formální definice, která přesně popisuje požadovanou funkcionalitu, je také zobrazena ve story boardu.
3) **Estimate**: Neboli odhad je jediná položka, kterou nemohou upravovat členové týmu. Toto omezení je opodstatněné tím, že jakmile se tým shodne na náročnosti úkolu a naplánuje jej do procesu vývoje, už by s tímto odhadem nemělo být manipulováno, nebo minimálně ne bez vědomosti scrum mastera.
4) **Notes**: Zde je hlavní prostor pro komunikaci na téma user story. Každý uživatel tam jednoduše může připsat své vlastní informace, případně opravit cizí chyby. Opět se zde spoléhá na předpoklad, že jakožto kolegové si budeme pomáhat, nikoli škodit.
5) **Log**: V této záložce můžeme nalézt historii pohybu user story včetně člověka, který změnu provedl a kdy ji provedl. Dá se takto snadno sledovat průběh jedné user story systémem, což je v některých případech velmi cenná informace.
6) **Time spent**: Obsahem této záložky je seznam lidí, kteří na dané user story pracovali, nebo na ni pracují. Je zde vidět také kolik času na user story strávili.

7) **Priority**: Udává prioritu user story s možností vybrat si z pěti stupňové škály. Tyto stupně se pak na story boardu projeví odlišným formátováním a velikostí písma a v případě vysokých priorit i barevným ohraničením.

3.1.8 Implementace story boardu

Z kapitoly 2 lze snadno vyčíst, že story board pro Scrum je podmnožinou story boardu pro Kanban. Našim úkolem tedy je implementovat story board použitelný pro Kanban. Jelikož všechny požadavky již byly určeny, nezbývá než se zde vyjádřit ke způsobu implementace, který byl zvolen v rámci bakalářské práce.

Na obrázku 3.2 vidíme ukázku funkčního story boardu fiktivního týmu z pohledu jeho standartního člena. Základní rozložení neobsahuje nic, co by po definicích v kapitole 2 mělo čtenáře překvapit – sloupce jsou vertikálně rozloženy, nejdůležitější směr pohybu je zleva doprava, v nadpisu sloupců si lze přečíst jejich název a maximální počet user stories, který je ve sloupci povolen.

První co stojí za povšimnutí je možnost přidávat nové úkoly do kteréhokoli sloupce podle potřeb konkrétního týmu. Jedním z možných způsobů využití je dekompozice user stories - to, že si tým naplánoval nějaké množství user stories ještě neznamená, že zvolené user story není možné dekomponovat na podproblémy. Případně tímto způsobem lze usnadnit podpůrné procesy pro asynchronní zadávání prioritní práce.

Druhá věc, která nás zajímá je vyjádření priority. V levém sloupci vidíme nahoře user story s nejvyšší prioritou, pod ní user story s vysokou prioritou. User Story, která je ve sloupci pro analýzu má standartní prioritu a user story ve vývoji má nejnižší možnou prioritu.

Dále si lze všimnout, že na user story z projektu omega pracuje jeden člověk, na user story z projektu alfa pracují současně tři lidé a ostatní user story čekají, až se jim začne někdo věnovat.

Poslední zajímavá věc je zobrazení textu user story přímo v tabuli. Tento způsob zobrazení nás nutí definici user story psát kratší a detailům se věnovat v poznámkách k user story. Toto je opět podloženo argumenty z knížek [6] a [10], které jednoznačně tvrdí, že user story musí být hlavně krátká, aby si ji kdokoli kdykoli rád přečetl a ihned získal základní představu, o požadované funkcionalitě.

Obrázek 3.2: Ukázka implementace user story.

3.2 Windows client

Windows client je mikro aplikace běžící většinu času v oblasti skrytých ikon hlavního panelu Windows. Tato aplikace slouží pro zaznamenávání odpracovaného času na server, pravidelně připomíná uživateli kontrolu aktuálnosti svého stavu a kontinuálně komunikuje se serverem pro automatické zjišťování aktivní user story uživatele. Detailnější informace o instalaci lze opět nalézt v příloze 2 „Scrumban Client Installation Guide.pdf“ a informace o používání v sekci FAQ.

3.2.1 Instalace a prerekvizity

Jedinou prerekvizitou je jsou .NET Framework verze 4.0, nebo novější a administrátorská oprávnění nutná pro nainstalování a konfiguraci klienta. Konfigurací klienta je myšleno zadání korektní http adresy serverové aplikace. Tato konfigurace je zprostředkována druhou klientskou miniaplikací, která slouží pouze ke zkontrolování správnosti adresy a její uložení do konfiguračního souboru.

V případě potřeby změny lze tedy buďto hromadně distribuovat konfigurační soubor napříč firmou, nebo znovu zpustit konfigurační aplikaci.

3.2.2 Aktivní user story

Aktivní user story je taková user story, ke které se uživatel zaregistroval skrze webové rozhraní. Toto je nutné kvůli přehlednosti a udržení klienta co nejjednodušším z důvodů případného vytvoření klientských aplikací i pro jiné platformy, nežli je Windows.

Aktivní user story si může zaregistrovat jen člen týmu, pod který daná user story spadá. Registrace je možné dosáhnout skrz modální dialog pro editaci user story (viz obrázek 3.3). Obdobným způsobem se lze z user story odregistrovat.

Každý uživatel může mít jen jednu user story zaregistrovanou jako aktivní, tudíž při přeregistraci na jinou user story přestává být původní user story aktivní.

Obrázek 3.3: Registrace uživatele k práci na user story.

3.2.3 Vykazování práce

První věc, na kterou je třeba upozornit je, že logika vykazování práce je umístěna na straně serveru a klient pouze posílá serveru příkazy k uložení, nebo resetování počítadla. To znamená, že klientská aplikace je se serverem spojena trvalým připojením, které udržuje i uživatelské oprávnění na serverové straně s využitím databáze. Pokud je tedy připojení přerušeno z jakéhokoli důvodu jak ze strany serveru, tak ze strany uživatele a je obnoveno do hodiny, pracovník nepřichází o započtený čas, nýbrž bude v předchozí relaci pokračovat. Navíc to má nezanedbatelné bezpečnostní výhody užitečné v případech, kdy za vykázaný čas je pracovník finančně odměňován.

Ukládání a resetování nemusí být vždy z iniciativy klientské aplikace. Ve skutečnosti existují čtyři akce, které ukládají odpracovaný čas na user story:

1) Stisknutí tlačítka „Stop working on story“ v klientské aplikaci.
2) Stisknutí tlačítka „Finish working on story“ ve webovém rozhraní v modálním dialogu pro editaci user story. Toto tlačítko nahrazuje u aktivní user story původní tlačítko „Register to work on this story“ (viz obrázek 3.3).
3) Při stisknutí kteréhokoli tlačítka hodinové připomínky klientské aplikace (viz. 3.2.4).
4) Při přeregistraci uživatele na jinou user story - v této situaci automaticky dochází k uložení odpracovaného času do původně aktivní user story a pro novou user story začíná nové počítání.

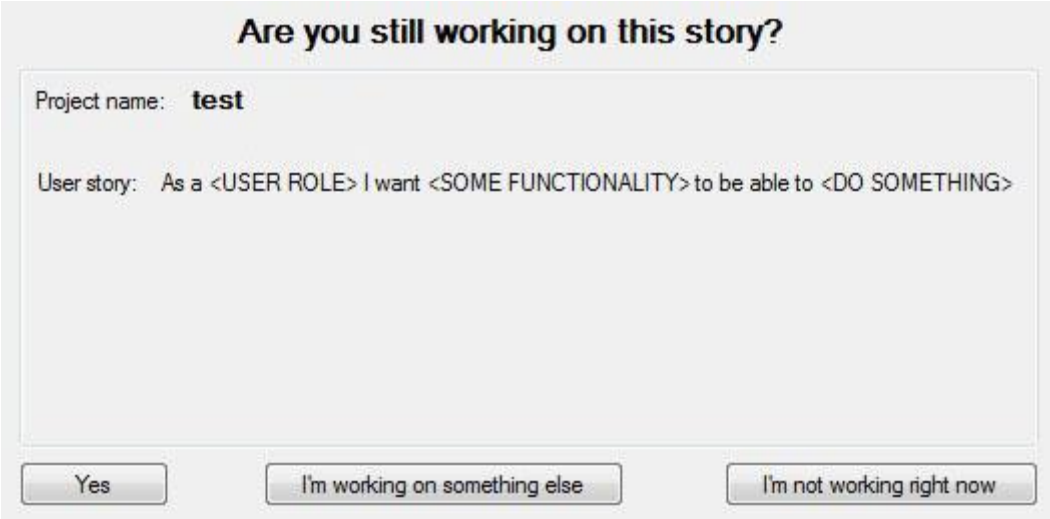
3.2.4 Systém upomínek

Klientská aplikace má dva druhy upomínek:

První z nich je v **hodinovém intervalu**. Ukázku této upomínky si lze prohlédnout na obrázku 3.4. Zobrazuje se pouze tehdy, pokud pracovník v poslední hodině nezměnil user story, na které pracuje. Po zobrazení této upomínky má pracovník 20 minut na to, aby upomínku potvrdil kterýmkoli způsobem, jinak mu tato hodina nebude započítána do odpracovaného času.

Pokud pracovník upomínku potvrdí, uplynulý čas mu bude vždy uložen na server, odlišnost mezi akcemi tlačítek je pouze v tom, jak se klientská aplikace zachová:

1) Stisknutím tlačítka „Yes“ bude aplikace pokračovat v zaznamenávání odpracovaného času.
2) Stisknutím tlačítka „I’m working on somwthing else“ nabídne uživateli změnu aktivní user story pomocí webového rozhraní a bude nadále pokračovat v zaznamenávání.
3) Stisknutím tlačítka „I’m not working right now“ přeruší zaznamenávání odpracovaného času.



Oprázek 3.4: Hodinová připomínka klientské aplikace.

Druhá z nich je **pětiminutová upomínka**, která se zobrazuje právě tehdy, když je u klienta zapnuto počítání času, ale uživatel není zaregistrován jako aktivní u žádné user story. Jak napovídá název, tato upomínka se zobrazuje každých 5 minut.

Tato upomínka zobrazí hlavní okno klientské aplikace na popředí. V takovémto případě se čas započítá do user story, kterou si pracovník později zaregistruje jako aktivní. Je tomu tak pro ošetření scénáře, kdy se pracovník skrze webové rozhraní odregistroval z aktivní user story, ale zapoměl si zaregistrovat jinou a klientskou aplikaci nechal pokračovat v zaznamenávání

odpracovaného času. Z obdobného důvodu je tlačítko „Start working on story“ ponecháno aktivní i v případě, že uživatel nemá zaregistrovanou žádnou user story.

4 Použité technologie a implementační detaily

Kapitola 4. je stručným popisem technologií a zásuvných modulů využitých v projektu Scrumban Board a způsobu jejich použití.

Některé z těchto technologií již byly jmenovány v předchozí kapitole v částech věnujících se instalačním procesům. Jelikož však jako instalační projekty byly využity standartní projekty poskytované s Visual Studiem a funkcionalita těchto projektů již byla vysvětlena, nebudeme se jimi dále zabývat.

4.1 Vývojové prostředí

Pro vývoj tohoto projektu bylo použito Microsoft Visual Studio 2010 Profesional SP1 (dále jen Visual Studio) se studentskou licencí poskytovanou od VUT FIT. Jako doplněk k Visual Studiu byl použit ASP.NET MVC3 Framework (viz.: [2]).

4.2 Přehled kódu

Ze zdrojových souborů je vidět, že řešení bakalářské práce se skládá z devíti projektů, z čehož čtyři projekty přísluší serverové, čtyři klientské aplikaci pro operační systém Windows a jeden projekt pro testování akcí prováděných při instalaci serverové aplikace. Z toho je jasné, že zde nemáme kapacitu pro rozebírání jednotlivých tříd, rád bych zde však uvedl alespoň základní vysvětlení na úrovni jednotlivých projektů pro ty, kteří by měli zájem nahlédnout do zdrojových souborů.

1) `ScrumbanBoard` je projekt obsahující jádro serverové aplikace.
2) `ScrumbanBoardDB` slouží pro zpracovávání databáze do formátu distribuatelného skrze instalační soubor projektu.
3) `ScrumbanClient` obsahuje jádro klientské aplikace.
4) `ScrumbanClientConfiguration` je projekt aplikace vytvářející konfigurační soubor pro klientskou aplikaci.
5) `ScrumbanClientSetupCustomActions` slouží k definici speciálních akcí potřebných při instalaci klienta. Příkladem budiž zpuštění konfigurační aplikace ihned po dokončení instalace.

6) `SetupCustomActions` je obdoba předchozího projektu, pouze pro serverovou část. Viz. k úloze projektu v kapitole 3.1.1.
7) `SetupCustomActions.Test` je testovací projekt obsahující metody testující jinak těžko odhalitelné chyby v projektu 6. Pro zpuštění některých testů jsou potřeba administrátorská oprávnění.
8) `ScrumbanClientSetup` je projekt generující instalační soubory klientské části.
9) `SetupProject` generuje instalační soubory serverová aplikace.

4.3 Jádro serveru aplikace

Jádrem aplikace zde rozumějme technologie, na kterých byly postaveny hluboké vrstvy serverové části.

4.3.1 Microsoft .NET Framework

Jak již bylo zmíněno, projekt využívá Microsoft .NET Framework verze 4.0 (dále jako .NET). Tento framework je určen pro jednoduchý přístup k prostředkům operačního systému Microsoft Windows. Jeho velkou výhodou je podpora mnoha různých jazyků, které dokáží spolupracovat díky společnému prostředí, které .NET poskytuje. Pokud bychom se chtěli zaměřit jen na projekt Scrumban Board, dal by se jmenovat například jazyk pro tvorbu dynamických webových stránek ASP.NET a kompilovaný jazyk C# [18].

4.3.2 SQL Express

Kvůli jednoduché redistribuci byl použit databázový nástroj Microsoft SQL Express 2008 (dále jen MSSQL), který je volně dostupný. Jelikož tato platforma sdílí manipulační nástroje s plnou verzi MSSQL 2008, je v případě potřeby jednoduché tuto databázi exportovat, případně by bylo možné i vydat verzi Scrumban Boardu, která by byla určena pro plnou verzi MSSQL.

4.3.3 Jazyk C#

Jazyk C# je objektově orientovaný jazyk vysoké úrovně vycházející z C/C++ a z Javy. Mimo jiné poskytuje velmi široké množství knihoven a zásuvných modulů, které programátorovi ušetří práci při nejedné úloze.

Aspektem, který je podstatný nejen pro mne, ale jistě pro každého správného programátora, který myslí na čtenáře svého kódu je jeho výborná vyjadřovací schopnost. Díky tomu lze jednoduše aplikovat základní zásady programátorů, jako je SOLID, KISS, DRY, TDD, nebo celou řadu návrhových vzorů [16].

4.3.4 ASP.NET MVC 3

MVC [8] je softwarová architektura rozdělující zodpovědnost za obsloužení uživatele do tří vrstev:

1) **Model** – Vrstva starající se o obecnou práci s daty.
2) **View –** Prezentační vrstva zodpovědná za tzv. user experience.
3) **Controller** – Řídící vrstva, která slouží pro obsluhu uživatelských požadavků.

Pro lepší pochopení si zde uvedeme příklad tzv. „end to end“ procesu (od uživatele až zpět k uživateli) zpracovaní uživatelského požadavku [8]:

1. **krok:** Uživatel odesílá požadavek na server.
2. **krok:** Controller přijme požadavek spolu s modelovými daty požadavku (tzn. daty odeslanými od uživatele).
3. **krok:** Controller zpracuje modelová data a odešle jej modelu.
4. **krok:** Model uloží změny v modelových datech a případně je zaktualizuje.
5. **krok:** Controller získává aktuální modelová data a odesílá je k zobrazení do view.
6. **krok:** View zobrazuje modelová data do uživatelského rozhraní.
7. **krok:** Čekání na odezvu od uživatele, která zpustí nový cyklus.

Zde popsaný proces není nutností, jelikož například uživatel nemusí odesílat žádná modelová data, nebo naopak nemusí být žádná modelová data potřebná pro zobrazení korektní odpovědi, nebo kdekoli při zpracování controllerem může dojít k přesměrování, nebo výjimce.

Tuto architekturu můžeme najít ve zdrojových souborech v projektu `ScrumbanBoard`, který obsahuje složky dle názvů vrstev architektury.

4.3.5 LINQ

LINQ je integrovaný jazyk, který přišel spolu s .NET Frameworkem 3.5. Je používán pro usnadnění obecné práce s hromadnými daty a to jak v podobě databáze, tak objektů, které spadají do kategorie kolekcí [18].

V Scrumban Boardu můžeme nejčastěji použití tohoto jazyka nalézt v modelech. Za povšimnutí také stojí to, že veškerou komunikaci s databází poskytují LINQ to SQL classes, které pracují na architektuře Active Record. Tyto třídy jsou zpravidla obohaceny o vlastní implementaci logiky pro zpracování dat v partial třídách, nebo třídách, které z LINQ to SQL tříd dědí. Tyto třídy obsluhují veškerou práci s databázovými daty.

Tento způsob řešení poskytuje velice čisté řešení bez potřeby jakékoli implementace databázových dotazů a vysokou přenositelnost na jiné databáze.

4.3.6 Zásuvné moduly

Jako zásuvný modul byl v projektu Scrumban Board použit MvcMailer, který slouží k odesílání notifikačních e-mailů uživatelům.

MvcMailer poskytuje jednoduché rozhraní, které odpovídá MVC struktuře. Tzn.: každý e-mail má svůj model, controller i view a postup zpracování e-mailu je podobná, jako bylo uvedeno v kapitole věnující se ASP.NET MVC3.

4.4 Asynchronní komunikace s klientem

Jak si lze všimnout například při přesouvání user stories na tabuli, některé uživatelské akce jsou prováděny asynchronně, abychom se vyhnuli nepříjemnému a zbytečnému překreslování uživatelského rozhraní. Zde si popíšeme dva typické scénáře použité v této práci.

4.4.1 AJAX services

AJAX je zkratkou pro Asynchronní JavaScript a XML. Jelikož JavaScript je jazyk interpretovaný internetovými prohlížeči, je jasné, že tyto AJAX services budou součástí serveru a budou sloužit pro komunikaci s uživatelem skrze webové rozhraní [18].

Jednoduchý způsob jejich implementace je zajištěn díky Windows Comunication Foundation (dále jen WCF), které nám poskytuje .NET Framework. Implementace je pak velice podobná controlleru z MVC struktury. Rozdílem však je, že odpovědí od serveru nemusí být, a zpravidla ani není, kód učený pro přímé zobrazení pomocí webového prohlížeče – tzn. není potřeba view. Místo toho je většinou vracen JSON objekt (objekt v notaci jazyka JavaScript).

Příklad takovéto service můžeme nalézt v projektu `ScrumbanBoard` ve složce `Services`.

4.4.2 Proxy client service

Proxy client service se nazývá vlastní implementace proxy objektu v klientské aplikaci [18]. Tato proxy poskytuje klientské aplikaci jednoduché rozhraní pro komunikaci s WCF service, která je určena pro obsluhu Scrumban Clienta.

Implementaci si lze prohlédnout v projektu `ScrumbanClient` v souboru `ServiceReference.cs` a v projektu `ScrumbanBoard`, složka `Services`, soubor `ClientService.svc`. Podíváme-li se tedy do těchto dvou souborů, první odlišnost oproti AJAX services, které bychom si měli všimnout je, že tato service implementuje svůj vlastní interface definovaný na straně serveru, klientská část pak pouze toto rozhraní implementuje a dědí z `ClientBase<IClientService>`. Třída `ServiceReference` je pak pro jednoduché použití

zabalena do návrhového vzoru singleton za pomocí třídy `ServiceClient` v projektu `ScrumbanClient`.

Oproti zpracování AJAX services z minulé podkapitoly je největší odlišnost v tom, že se jedná o persistentní spojení s udržováním uživatelských informací na straně serveru a místo JSON objektu WCF service vrací silně typované proměnné dle anotace jazyku C# [18].

4.5 Prezentační část aplikace

Prezentační část, neboli v názvosloví MVC architektury view se skládá hlavně z ASP.NET šablony, kterým se zde však věnovat nebudeme, jelikož jsme si je představili výše v kapitole věnující se .NET Frameworku a MVC architektuře. Místo toho se zaměříme na JavaScriptovou část.

4.5.1 jQuery

JQuery je v současné době jeden z nejrozšířenějších frameworků, které pro JavaScript existují. Mimo jádro frameworku existuje celé množství zásuvných modulů. Podívame-li se pak na projekt Scrumban Board, nalezneme zde použití jQuery validátorů, jQuery UI (kterému se budeme věnovat dále), nebo modulu pro tooltipy.

4.5.2 jQuery UI

Tento zásuvný modul slouží k zlepšení dynamičnosti uživatelského rozhraní v mnoha ohledech. Při pohledu na Scrumban Board si můžeme povšimnout jeho využití při práci s každou tabulí – cokoli co je možno táhnout myší, nebo například vyskakovací menu po najetí na avatár uživatele v pravém horním rohu. Dalšími příklady jsou „datetime picker“ pro vybírání počátečního a koncového času u sledování výkonnosti, nebo v tomtéž formuláři našeptávač jmen uživatelů. Posledním použitým prvkem je tzv. „akordeon“, který zobrazuje například jednotlivé týmy v části pro sledování výkonnosti.

To jsou jen některé z funkcí jQuery UI. Dále lze jmenovat podporu pro roztažitelnost HTML elementu myší obdobným způsobem, jakým můžeme měnit velikost oken v dnešních operačních systémech. V sadě nástrojů je poskytována celá řada tlačítek, posuvníků, progres barů, sliderů a efektu pro zobrazení, nebo skrývání elementů.

4.6 Klientská aplikace pro Windows

Jelikož je Scrumban Client jednoduchá klientská aplikace, nejsou v něm využívány žádné další technologie, mimo .NET Framework 4.0.

Pro tento program bylo nutné vyžadovat plnou verzi .NET Frameworku, nikoli klientskou distribuci, která je podstatně menší. Plná verze je nutná, jelikož klientská aplikace pro kompilaci potřebuje knihovní soubory z projektů, pro které plnou verzi .NET Frameworku nezbytně potřebují.

5 Závěr

Cílem této práce bylo vytvoření informačního systému pro sledování výkonnosti programátorů v malé až střední firmě. Jak je vidět z bakalářské práce, tak sledování výkonnosti je až posledním krokem, z dlouhé cesty, kterou tento projekt prošel. Bez informačního systému, který sleduje vývoj funkcionality, bychom totiž neměli měřítko ke kterému přirovnávat dobu strávenou vývojem.

Nutnou součástí projektu byla klientská aplikace, bez které by sledování výkonnosti jednotlivých pracovníků nebylo možné. Tato miniaplikace je jistě velice zajímavá, obzvláště pro firmy poskytující možnost pracovat z domu. Avšak důležitější nežli měření počtu dosažených story pointů za množství času konkrétního člověka je vždy individuální přístup a týmová souhra vstříc výsledkům. Tabulkové rozdíly mezi lidmi by neměly nikdy být podstatnější, nežli kontinuální proces zvyšování efektivity práce týmu. Také považuji za nezbytně důležité si uvědomit, že programátorský tým je velice komplexní celek, který pod dobrým vedením je schopen velice efektivní samosprávy.

Z kapitoly 3 je jistě vidět, že aplikace není hotová a možných rozšíření je spousta. Nejnutnější jádro však bylo vyvinuto, tudíž dle myšlenky tzv. lean managementu je hotovo minimální možné množství funkcionality potřebné k nasazení projektu do praxe a další funkcionalitu bude na čase implementovat až na základě ohlasů, zkušeností a přesnějších informací o požadavcích našich zákazníků. Již jsem požádal autorizované osoby ve firmách Kentico s.r.o. a Publero s.r.o. o zvážení testovacího provozu této aplikace. Další vývoj je tedy do značné míry v jejich rukou a případně i v rukou vedoucího této práce, Ing. Jakuba Malého, se kterým jsem v případě zájmu ochoten spolupracovat dále na vývoji, nebo nasazení projektu.

Osobně pro mne tento projekt měl největší přínos po stránce managementu. Když si znovu přečteme předchozí kapitoly, doufám, že na vás dýchne duch agilních metodik tak, jako na mne. Základní myšlenkou agilních metodik totiž nejsou iterace, ani odlišná pravidla. Jsou to tyto základní myšlenky: zapomeňme na složité procesy, přestaňme se k programátorům chovat jako k dělníkům a dejme jim do rukou více zodpovědnosti. Poskytněme jim nástroje a nadřízené, pomocí kterých sami uvidí, co zlepšuje jejich práci a dejme jim volnost dělat rozhodnutí, které jim pomohou plnit neustále se měnící požadavky zákazníků.

Po technologické stránce pro mne však byl přínos také nemalý. Ostatně pouze kapitola věnující se použitým technologiím čítá nemalé množství frameworků, technik a jazyků, které byly použity. Některé jsem znal, jiné nikoli, nebo jsem je pouze nikdy nepoužil společně. Kdybych však měl jmenovat technologii, která mne nadchla nejvíce, pak jsou to WCF services a jejich obrovské možnosti síťové komunikace.

Věřím, že velice obsáhlý teoretický základ z kapitoly 2 poslouží jednak pro vývoj alternativních produktů, jednak by dokázal sloužit i jako základ pro každého, kdo chce ve svém týmu nasadit a používat agilní metodiky. Dovolím si zde proto citovat agilní manifest, který je považován

za oficiální shrnutí nejzásadnějších myšlenek agilních metodik a který mne provázel celého posledního tři čtvrtě roku na všech konferencích, kterých jsem se účastnil, ve všech knížkách a v každém kousku funkcionality tohoto projektu [3]:

Objevujeme lepší způsoby vývoje software tím,
že jej tvoříme a pomáháme při jeho tvorbě ostatním.
Při této práci jsme dospěli k těmto hodnotám:

Jednotlivci a interakce před procesy a nástroji
Fungující software před vyčerpávající dokumentací
Spolupráce se zákazníkem před vyjednáváním o smlouvě
Reagování na změny před dodržováním plánu

Jakkoliv jsou body napravo hodnotné,
bodů nalevo si ceníme více.

Literatura

[1] *Gravatar* [online]. 2007 [cit. 2012-05-11]. Dostupné z: http://cs.gravatar.com/

[2] ASP.NET MVC 3. *Microsoft ASP.NET* [online]. 2011 [cit. 2012-05-11]. Dostupné z: http://www.asp.net/mvc/mvc3

[3] *Manifest Agilního vývoje software* [online]. 2001 [cit. 2012-05-12]. Dostupné z: http://agilemanifesto.org/iso/cs/

[4] Agile software development. In: *DocStoc*: *Documents & Resources fot Small Bussiness and Professionals* [online]. [cit. 2012-04-28]. Dostupné z: http://www.docstoc.com/docs/6245268/Agile_software_development

[5] BOEHM, B. A spiral model of software development and enhancement. *ACM SIGSOFT Software Engineering Notes*. 1986-08-01, roč. 11, č. 4, s. 22-42. ISSN 01635948. DOI: 10.1145/12944.12948. [cit. 2012-04-28]. Dostupné z: http://portal.acm.org/citation.cfm?doid=12944.12948

[6] COHN, Mike. *Agile estimating and planning*. Upper Saddle River: Prentice Hall, 2006, 330 s. ISBN 01-314-7941-5.

[7] E.A. Edmonds. A Process for the Development of Software for Nontechnical Users as an Adaptive System. *General systems*: *yearbook of the Society for the Advancement of General Systems Theory*. New York: Gordon and Breach, 1974, roč. 19, č. 19, s. 215-18. ISSN 0072-0798.

[8] GALLOWAY, Jon S. *Professional asp.net mvc 3.0*. 1. vyd. Indianapolis, IN: Wiley Pubishing, Inc., 2011. ISBN 11-180-7658-3.

[9] HELLESØY, Aslak. *Cumulative Flow Diagrams with Google Spreadsheets*. BEKK CONSULTING AS. *BEKK Open* [online]. 2009 [cit. 2012-05-04]. Dostupné z: http://open.bekk.no/cumulative-flow-diagrams-with-google-spreadsheets/

[10] KEITH, Clinton. *Agile game development with Scrum*. Upper Saddle River, NJ: Addison-Wesley, 2010, 340 s. Addison-Wesley signature series. ISBN 978-032-1618-528.

[11] KNIBERG, Henrik a Mattias SKARIN. *Kanban and Scrum*: *making the Most of Both*. s. l: Lulu.com, 2010. ISBN 978-0-557-13832-6. [cit. 2012-04-28]. Dostupné z: http://www.infoq.com/minibooks/kanban-scrum-minibook

[12] KONIECZNY, Monika. *Agilia Conference: Gamification + Storytelling - Magic Product Owner's Tools.* Brno, 2012.

[13] LADAS, Corey. *Scrumban and other essays on Kanban System for Lean Software develoment*. Saetle, WA: Modus Cooperandi Press, 2008. ISBN 978-057-8002-149.

[14] LARMAN, C. a V.R. BASILI. Iterative and incremental developments. a brief history. *Computer*. 2003, roč. 36, č. 6, s. 47-56. ISSN 0018-9162. DOI: 10.1109/MC.2003.1204375. [cit. 2012-04-28]. Dostupné z: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1204375

[15] MARKHAM, Daniel. *ScrumMaster*: *a tiny giant book* [online]. 1. vyd. 2012 [cit. 2012- 04-28]. Dostupné z: http://tiny-giant-books.com/scrummaster.htm

[16] MILLETT, Scott E. *Professional asp.net design patterns*. 1. vyd. Indianapolis, IN: Wiley Pub., Inc., 2010. ISBN 04-702-9278-4.

[17] NAUR, Peter a Brian RANDELL. *SOFTWARE ENGINEERING*: *Report on a conference sponsored by the NATO SCIENCE COMMITTEE Garmisch, Germany, 7th to 11th October 1968* [online]. 2001. vyd. Brussels 39, Belgium: Scientific Affairs Division, NATO, 2001 [cit. 2012-04-28]. Dostupné z: http://www.google.cz/url?sa=t&rct=j&q=&esrc=s&source=web&cd=3&ved=0CD4QFjAC&url=http%3A%2F%2Fhomepages.cs.ncl.ac.uk%2Fbrian.randell%2FNATO%2Fnato1968.PDF&ei=kAKcT_HMDY-l-gbKz5XzDg&usg=AFQjCNEGETrRqg32OzwUMWRKufcMViWOig

[18] NORTHRUP, Anthony a Mike SNELL. *MCTS self-paced training kit (exam 70-515): web applications development with Microsoft .Net framework 4*. Redmond, Wash.: Microsoft Press, c2010, 967 s. MCTS self-paced training kit, exam 70-515. ISBN 07-356-2740-1.

[19] POPPENDIECK, Mary a Thomas David POPPENDIECK. *Leading lean software development*: *results are not the point*. Upper Saddle River, NJ: Addison-Wesley, 2010, 278 s. ISBN 03-216-2070-4.

[20] ROYCE, Winston W. *Proceedings*: *9th International Conference on Software Engineering, March 30-April 2, 1987, Monterey, California, USA*. Washington, D.C: IEEE Computer Society Press, 1987. ISBN 08-979-1216-0 [cit. 2012-04-28]. Dostupné z: http://leadinganswers.typepad.com/leading_answers/files/original_waterfall_paper_winston_royce.pdf

[21] VAN VERTH, Kees. BURNDOWN CHART. *Burndown Chart* [online]. 2008 [cit. 2012-05-01]. Dostupné z: http://www.burndownchart.nl/

[22] ZAPF, Andreas. Scrum board. *Andreas Zapf's Blog* [online]. 2011 [cit. 2012-04-29]. Dostupné z: http://andreaszapf.de/blog/wp-content/uploads/2011/03/002-Scrum-Board-11.png

Seznam příloh

Obsah DVD

- Průvodce instalací serverové části **Scrumban Board Installation Guide.** Tento dokument je umístěn ve složce „Scrumban Board Installation“ jak ve zdrojové podobě, tak ve formátu PDF.

- **Instalační soubory serverové části** jsou umístěny ve stejné složce, jako její installation guide („Scrumban Board Installation“).

- Průvodce instalací klientské části **Scrumban Client Installation Guide.** Tento dokument lze nalézt ve složce „Scrumban Client Installation“ jak ve zdrojové podobě, tak ve formátu PDF.

- **Instalační soubory klientské části** jsou ve složce „Scrumban Client Installation“.

- **Zdrojové texty** jsou umístěny ve složce „Zdrojové soubory programu“.

- **Bližší informace s návody pro používání obou částí lze nalézt v záložce FAQ ve webové aplikaci.** Tyto informace zde byly umístěny kvůli jednodušší možnosti úprav těchto pro potřeby konkrétní klientské firmy a pro jednodušší distribuci mezi jejími zaměstnanci.