Agilní proces pro výuku softwarového inženýrství v předmětu ASWI (verze 1.1)

Přemek Brada
Katedra informatiky a výpočetní techniky
Fakulta aplikovaných věd, Západočeská univerzita v Plzni
<brada@kiv.zcu.cz>

leden 2011

Abstrakt

Tento dokument popisuje softwarový proces navržený pro výuku procesních praktik v předmětu Pokročilé softwarové inženýrství (ASWI) ve verzi pro akademický rok 2010/2011. Mezi jeho základní vlastnosti patří zaměření na agilní způsob vývoje kombinované s ověřenou praxí činit v projektu včas klíčová globální rozhodnutí. Kromě popisu struktury iterací, typů rolí a artefaktů definovaných ASWI procesem a potřebné softwarové podpory jsou také zmíněny důvody pro konkrétní volbu jednotlivých prvků a jejich aspektů.

1 Úvod

Agilní přístup k vývoji software [13, 2] je již přes 10 let s úspěchem používán u mnoha menších a středně velkých projektů. K jeho přednostem patří schopnost projektového týmu reagovat v průběhu řešení na měnící se potřeby zákazníka a okolí projektu, přesnější plánování, a důraz na dodání vysoké užitné hodnoty výsledného produktu.Znalost základů tohoto přístupu se tedy stává nutnou součástí výuky softwarového inženýrství.[14][6] (srovnej též UPEDU [15] či AgileUP [1] jakožto varianty Unifikovaného procesu [12, 8]).

Texto text specifikuje aktualizovanou podobu softwarového procesu navrženého v roce 2009 pro použití v rámci předmětu Pokročilé softwarové inženýrství (KIV/ASWI) na navazujícím stupni inženýrského studia FAV ZČU. Motivací pro využití agilních prvků v procesu byla zkušenost z předchozích let (před rokem 2009), kdy projekt měl pevně stanovené tři "iterace" s délkou 4 týdny — studenti používající takový proces nemohli rozpoznat výhody iterativního přístupu, nenaučili se plánovat svoje aktivity, aktivně pracovat se zákazníkem a zpracovávat změny, docházelo ke splývání konceptů fáze a iterace, nebyl dostatečně jasný význam milníků [3]. U některých týmů se proces blížil spíše vodopádovému modelu a/nebo byla tendence akcentovat (formální) dokumentaci na

úkor aktivit účelných z hlediska vedení projektu a potřeb zákazníka. Nevýhodou z pedagogického hlediska byla také relativně malá viditelnost do probíhajícího projektu.

Vytvoření nové verze procesu pro ASWI projekty bylo tedy vedeno snahou odstranit nepotřebné formality a soustředit se na podstatu iterativního softwarového procesu. Oba tyto aspekty jsou důležité jak z hlediska pedagogického (ohodnocení skutečně získaných znalostí, nikoli formálně vykazovaných výsledků) tak praktického (příprava pro budoucí zaměstnání). Obdobné cíle přitom sleduje agilní přístup k tvorbě software [13], takže bylo přirozené, že byl použit jako kostra procesního frameworku nově vytvářeného procesu — zejména metodika Scrum [16, 10] a [7].

První nasazení ASWI procesu proběhlo v akademickém roce 2009/2010, kdy v předmětu pracovalo celkem 15 týmů na projektech různého charakteru. Zkušenosti z prvního roku (viz též [5]) lze shrnout klíčovým konstatováním, že během 2-3 iterací došlo u většiny týmů k osvojení si návyků iterativního vývoje včetně jejich praktické realizace v konkrétních nástrojích. Byl tedy dosažen cíl pedagogický, který je hlavním účelem tohoto procesu. Naopak mezi problémy patřilo používání příliš striktního a neměnného rozdělení rolí uvnitř týmů, nejasné vnímání pedagogicky významné hranice fází (tj. kotevních bodů) procesu, a nízké využití základních metrik pro plánování a řízení vývoje. Tyto nedostatky se snaží řešit proces inovovaný na verzi 1.1, popsaný v tomto dokumentu.

V následujících oddílech bude nejprve stručně představen kontext pro ASWI proces a následně podrobně popsány jeho jednotlivé složky — struktura iterací, milníky, týmové role a vytvářené artefakty. Následuje stručné seznámení s podpůrnými nástroji a závěrečné shrnutí.

2 Východiska a motivace agilního ASWI procesu

Charakteristiky ASWI projektů ovlivňující podobu procesu je možno shrnout do následujících bodů:

- *malé týmy* — standardní tým tvoří 4 studenti, tato velikost je dlouhodobě ověřena jako nejvhodnější v daném kontextu (menší tým nemá dostatečnou dynamiku, u větších týmů komunikační režie převáží nad podstatou [7]);
- *zejména pedagogický cíl* — přestože studenti již prošli výukou základů softwarového inženýrství, jejich zkušenost s praktickým vedením projektu ve vlastní režii je malá. Cílem projektu je především naučit se chápat softwarový proces v jeho souvislostech a "dostat do ruky" klíčové přístupy a praktiky, nikoli dodat produkt se 100% plánované funkčnosti nebo nastudovat nové technologie;
- *malý rozsah* — projekt je časově omezen na jeden semestr (14 týdnů), v jehož rámci studenti studují i ostatní předměty. Práci na projektu proto jednotlivý student může věnovat maximálně 10 hodin týdně, což celkově omezuje rozsah zhruba na 70 člověko-dní [4];

- *omezená dostupnost mentorů* — v jednom běhu ASWI zároveň pracuje mezi 12-20 týmy, které má na starosti (mentoruje, koučuje) jen několik vyučujících, takže připadá zhruba 6-8 týmů na vyučujícího. Při této alokaci není možné, aby mentor byl pro tým dostupný kdykoli je potřeba (tak jak je tomu v případě Scrum Mastera), a osobní kontakt pro potřebné vedení a rady je tak omezen na zhruba 1h týdně.

- *reálný zákazník* — zadáním projektu je vždy skutečná potřeba skutečného zákazníka, přičemž zhruba z poloviny jde o zákazníky mimo univerzitu (neziskové organizace, oddělení rozpočtových organizací, malé společnosti, softwarové firmy). Studenti se tedy potkávají se "skutečným světem" konkrétní problémové oblasti, její terminologie a velmi často lidí bez IT zázemí.

Vzhledem k těmto charakteristikám je zřejmé, že ASWI proces musí být relativně jednoduchý ale přitom pokrývající všechny podstatné aspekty moderních přístupů k tvorbě softwarových systémů.

3 Struktura a základní charakteristiky procesu

Softwarový proces pro studentské projekty ASWI je iterativní, agilně orientovaný proces pro řízení tvorby malých až středně velkých softwarových systémů, který je založen na metodice Scrum [16] doplněné o některé pedagogicky významné momenty—zejména koncept fází či milníků procesu definovaných Boehmem [3] a použití softwarových nástrojů pro plánování a monitorování procesu. Proces je nejčastěji používán v kontextu tvorby cílového systému tzv. na zelené louce případně i pro rozvoj již existujících produktů[1] týmem, který se zaučuje do praxe iterativního způsobu vývoje.

Není definováno žádné omezení na použité metody sběru požadavků, návrhu a implementace software, či vývojové prostředky, pokud nejsou v protikladu se základními vlastnostmi procesu. Během projektu je ale z praktických a pedagogických důvodů týmům "nadirigováno" použití jednotné sady integrovaných softwarových nástrojů pro vedení agendy projektu a správu artefaktů, jakož i dodržování vlastního ASWI procesu—v tomto aspektu nejsou týmy, na rozdíl od metodiky Scrum, plně pravomocné zvolit si vlastní prostředky a postupy k řešení.

V následujících odstavcích podrobněji popíšeme, jak je proces vystavěn z pohledu základního přístupu, časového průběhu projektu, rozdělení rolí a vytvářených artefaktů.

[1] Při výběru témat projektů je dbáno ze strany vyučujících na to, aby po technologické stránce projektový tým pokud možno vystačil s již dříve získanými znalostmi, a mohl se tedy soustředit na procesní stránku věci.

3.1 Průběh projektu a iterace

Hlavní strukturuou procesu jsou iterace, jejichž průběh víceméně odpovídá pravidlům metodiky Scrum. Celkový rámec projektu je volně strukturován do fází ohraničených milníky, jejichž cílem je zdůraznit pedagogicky důležité momenty v procesu vývoje software, eliminovat nejrizikovější oblasti projektu v daném čase a umožnit zhodnocení správnosti jeho průběhu.

3.1.1 Iterace (mikro proces)

Základní tep procesu tvoří timeboxované iterace (viz např. [13]) se standardní délkou 2 týdny; v pravomoci týmu je tuto délku výjimečně změnit při plánování následné iterace. Průběh iterace postupně prochází následujícími body, viz obr.1:

1. Naplánování iterace (cca 2h) — dohodnutí cíle iterace a ujasnění implementovaných funkčností se zákazníkem, určení technických prací a časových bodů v iteraci.

2. Práce na iteraci, během nich Weekly standup schůzky (cca 20min/týdně) — realizační práce dle cíle iterace a fáze projektu; Weekly standup je schůzka celého týmu, která se koná pravidelně jednou týdně během iterace a používá stejné schema jako denní schůzka týmu v metodice Scrum (zde je patrná modifikace pravidel na dostupný kalendářní čas).

3. Předání výsledků (cca 2h) — schůzka se zákazníkem, během které jsou vytvořené výsledky[2] předvedeny a předány zákazníkovi. Je provedeno uzavření příslušných změnových požadavků a zákazník dává týmu potvrzení o dosažení cíle daného pro proběhlou iteraci. Odpovídá technice Sprint Review ("demo") ve Scrumu.

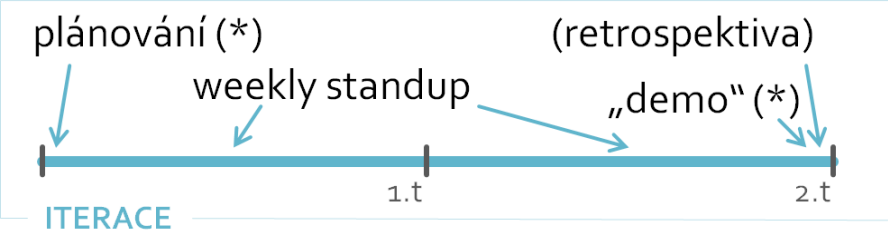
4. Retrospektiva (cca 1/2h, každou sudou iteraci + v poslední iteraci projektu za celý projekt) — interní zhodnocení průběhu ukončené/ých iterace/í v rámci týmu. Odpovídá aktivitě Sprint Retrospective z [10].

5. Po předání výsledků a případné retrospektivě tým informuje mentora o ukončení iterace a následuje zhodnocení iterace s mentorem (cca 1/2h).

Během celého semestrálního projektu tým projde nejméně čtyřmi iteracemi, jsou ovšem možné i případy 5-6 iterací na projektu tam, kde tým zvolí kratší timebox.

Iterativní přístup poskytuje v ASWI procesu standardní výhody, zejména relativně přesné plánování a možnost zákazníka korigovat výsledek projektu už v jeho průběhu podle toho, jak se s postupujícími pracemi mění a zpřesňuje pochopení zadání.

[2] Očekává se, že nejpozději od 2. iterace jde o spustitelnou aplikaci, nikoli obrázky a povídání.



Obrázek 1: Průběh ASWI iterace, symbol (*) označuje interakci se zákazníkem

3.1.2 Milníky (makro proces)

Rámec pro globální řízení projektu má podobu čtyř milníků, kterých musí tým průběžně dosáhnout. Jsou jimi tzv. kotevní body, které jsou jakožto ověřený koncept převzaty z [3] a RUP (název posledního milníku je změněn):

- "Lifecycle Objectives" (LCO) — ukončuje fázi zahájení projektu a stanovení jeho vize;
- "Lifecycle Architecture" (LCA) — ukončuje fázi určení architektury řešení;
- "Initial Operational Capability" (IOC) — ukončuje hlavní realizační práce;
- "Product Release" (REL) — završuje předání produktu do rutinního provozu a celý semestrální projekt.

Cílem každého milníku po praktické stránce je ošetřit co nejdříve rizika, která mohou projekt ohrozit, a zajistit dostatečnou stabilitu informací a rozhodnutí včas, ale nikoli příliš brzy, vzhledem k příslušné fázi vývoje projektu. Tento přístup zároveň zabraňuje divergování projektu od jeho cíle, k čemuž může docházet při čistě iterativním vývoji. Po pedagogické stránce tyto milníky vedou studenty mj. k soustředění se na aspekty, které jsou v dané chvíli klíčové pro celkový úspěch projektu, tedy k nadhledu nad projektem.

Obrázek 2: Makro proces ASWI projektů

Celkový účel zvoleného procesního rámce z výukového pohledu je poskytnout studentům základní rutinní zkušenost s tím, jaké aktivity je třeba provádět při vývoji softwarového produktu a jaké techniky je přitom vhodné použít. Milníky pak zároveň dávají návod, k čemu je třeba v daném stadiu vývoje dospět.

3.1.3 Technické vs. manažerské činnosti

V každém softwarovém procesu lze identifikovat činnosti dvojího druhu—technické, soustředící se na vytváření produktu, a manažerské, zabezpečující chod projektu.

Jak už bylo řečeno výše, v ASWI procesu nejsou pro technické činnosti (sběr požadavků, návrh, implementaci software, testování) pevně určeny konkrétní metody. Jejich výběr je zcela v pravomoci týmu, samozřejmě s ohledem na charakter zadání. To usnadňuje nasazení procesu pro projekty různého typu jak co do aplikační domény, tak z pohledu použitých technologií. Jsou nicméně preferovány moderní přístupy související s objektovým programováním, mj. modelování v UML, refactoring a automatizované jednotkové testy; viz též oddíl 3.3.2 níže. K získání informací o konkrétních postupech může tým využít znalostní báze, kterou představuje Rational Unified Process (RUP).

Postupy pro řízení projektu jsou naproti tomu z velké části předepsány, jak je patrné výše. Referenčním zdrojem informací pro tuto oblast jsou (kromě tohoto dokumentu a návodů na stránkách předmětu) zejména zdroje týkající se Scrumu [16, 10] doplněné o vybrané partie Rational Unified Processu.

V rámci řízení projektu musí tým zajistit soulad prováděných technických činností (během dané iterace) s celkovým rámcem projektu (daným nejbližším milníkem) a již uplynulým časem. Například lze důvodně očekávat, že v první iteraci či dvou bude probíhat zejména prvotní sběr požadavků a hledání alternativ řešení kulminující ustanovením vize projektu, nebo že poslední iterace bude soustředěna na akceptační testování a uvedení produktu do rutinního provozu.

3.2 Týmové role

ASWI proces používá oproti metodice Scrum relativně bohatý repertoár rolí a artefaktů, daný zejména pedagogickými potřebami. V rámci týmu jsou definovány následující role (řazené abecedně):

- analytik — odpovídá za sběr požadavků od zákazníka a jejich dokumentaci;
- architekt — odpovídá za návrh architektury a dohlíží na to, aby se během projektu tým od navržené architektury neodchýlil;release inženýr — má zodpovědnost za dodání hotového produktu zákazníkovi;
- systémový inženýr — role zodpovědná za nástroje a vývojové prostředí používané týmem;
- tester — zodpovídá za to, že všechny položky v plánu iterace prošly před jejich uzavřením nějakou formou testování;

- vedoucí (team leader) — hlavní odpovědností této role je starat se o fungování týmu, sledovat vývojový proces a jeho artefakty; má některé charakteristiky role Scrum Master;
- vývojář — spoluzodpovědný za vývoj produktu, psaní a kvalitu zdrojového kódu.

V procesu jsou dále definovány dvě role externí:

- mentor — vyučující, pomáhá s nejasnostmi ohledně vývojového procesu a práce s podpůrnými nástroji, provádí pedagogický dohled nad týmem;
- zákazník — podílí se na tvorbě vize produktu, definuje jeho požadavky a hodnotí jejich splnění; částečně odpovídá roli Product Owner.

Rozdělení rolí mezi jednotlivé osoby ve 4-členném týmu je v pravomoci týmu, přičemž tým má fungovat jako celek se sdílenou zodpovědností za veškeré činnosti, nikoli jako soubor jednotlivců řízených vedoucím (viz koncepty "empowered team" a "whole-team responsibility" ve Scrumu [7] [7]). Je zjevné, že jednotlivý člen týmu zastává více rolí. Z pedagogického hlediska není podstatné, kdo jakou roli zastává, protože hodnocení je přidělováno týmu jako celku (vyjma případů, kdy dojde k významným problémům uvnitř týmu nebo k jeho rozpadu).

Co se externích rolí týče, mentor má některé funkce Scrum Mastera ovšem s přihlédnutím k tomu, že není týmu časově plně dedikován a že je zároveň hodnotitelem práce týmu. Zákazník není "on-site" jako v metodice Extrémní programování, nicméně se předpokládá, že bude týmu k dispozici alespoň pro schůzky předání výsledků a plánování.

Z výukového pohledu jsou role takto definované zejména proto, aby pomáhaly studentům uvědomit si, jaké aktivity je třeba v projektu provádět. Mají tedy pomocnou úlohu a formální přidělení rolí jednotlivým členům není pro hodnocení práce týmu klíčové.

3.3 Artefakty

Protože metodika Scrum používá (záměrně) velmi malé množství technických a organizačních meziproduktů, byla v ASWI procesu hlavně z pedagogických důvodů nadefinována konkrétní širší sada artefaktů, které tým musí respektive může vytvořit. Cílem je dát studentům příležitost "osahat si" v dostatečné míře různé artefakty tak, aby měli dostatečné znalosti v případě, kdy si budou moci v dalších projektech takovou sadu určovat sami.

3.3.1 Přehled artefaktů

Klíčovými technickými artefakty jsou:

- vize projektu — definuje účel a důvod pro vyvíjený software z pohledu zainteresovaných osob, popisuje klíčové požadavky na něj kladené; tvoří základ pro product backlog a předání release produktu;

- popis architektury — zachycuje klíčové aspekty zvoleného řešení, nutné pro jeho pochopení a dokumentaci; může mít různou formu (dokument, webové stránky, UML modely, apod.);
- product backlog — seznam všech požadavků na produkt, povinně vedený v systému pro správu změn (a to i pokud je vytvořen formalizovaný Dokument specifikace požadavků);
- seznam chyb — vedený v systému pro správu změn, typicky spolu s product backlogem;
- testy a testovací protokoly — předpisy pro automatizovaně nebo ručně prováděné testování implementace (jednotkové a akceptační testy), a reporty dokládající výsledky testování (forma např. log xUnit testů nebo komentáře u uzavřených testovacích úkolů v bugtrackeru). V pozdějších iteracích lze důvodně očekávat, že specifikace testů a protokoly budou obsahovat regresní testy;
- zdrojový kód — vlastní realizace produktu, povinně vedená v systému pro správu verzí;
- release produktu — obsahuje instalaci finálního produktu, potřebné dokumentace a datové či konfigurační soubory;
- dokument specifikace požadavků (nepovinný) — možno využít šablon dle RUP nebo Wiegers [18], šablona dle standardu IEEE 830 je považována za koncepčně mírně zastaralou;
- prototyp uživatelského rozhraní či ovládání (nepovinný) — slouží k vyjasnění uživatelských požadavků;
- technologický prototyp (nepovinný) — slouží k analýze technických řešení v rámci definice architektury;
- plán konfiguračního řízení (nepovinný) — popisuje konvence a pravidla pro práci týmu se systémy správy změn a verzí;
- uživatelská a instalační příručka (nepovinná) — dle potřeb zákazníka.

Manažerskými artefakty jsou

- webová stránka projektu — základní informace o projektu a odkazy na jeho součásti dostupné on-line; titulní stránka musí obsahovat název projektu, cíl (odstavec z vize), jména členů týmu + kontakty, odkaz na úložiště a bugtracker, povinné artefakty ke stažení;
- plány iterace — mají formu jednoduchého elektronického dokumentu (např. webové stránky) shrnujícího cíle a časový rámec iterace, každý plán je povinně svázán s konkrétními pracovními položkami přiřazenými pro iteraci v systému pro správu změn;

- záznamy z weekly standup — zachycují obsah těchto schůzek ve formě elektronického dokumentu či webové stránky;
- potvrzení o dosažení cíle iterace — vystavuje zákazník na základě závěru z bodu iterace Předání výsledků, preferovaná je co nejjednodušší forma (např. email či "odfajfkování" a podpis na dokumentu plánu iterace);
- záznamy z retrospektiv — zapsané výsledky reflexe ukončené iterace, s formou podobou weekly standup protokolům;
- předávací protokol — jednoduchý formulář stvrzující podpisem převzetí funkčního produktu zákazníkem na konci projektu; v tomto případě je nutná papírová forma dle šablony dostupné ze stránek předmětu.

Nakonec je definován jediný čistě administrativní artefakt

- archiv projektu — archiv s všemi artefakty vytvořenými během projektu (aby bylo možné ex-post projekt a jeho jednotlivé iterace posoudit procesně i věcně); předává tým mentorovi po dosažení milníku Product Release pro účel uzávěrky projektu .

Obsah a formální struktura artefaktů Product backlog, Záznam z weekly standup a Záznam z retrospektivy jsou odvozeny od zvyklostí metodiky Scrum; obsah a struktura artefaktů Vize projektu, Popis architektury, Plán iterace a Release produktu jsou zjednodušenými verzemi analogických artefaktů v metodice RUP. Lze říci, že proces definuje pouze dva velké artefakty dokumentové povahy (vize a architektura), které jsou relativně náročné na vytváření (což odpovídá jejich významu v projektu).

Z výukového pohledu poskytuje zvolený výběr a definice artefaktů studentům důležitou základní orientaci v otázce, k jakým výstupům a (mezi)výsledkům má softwarový projekt obecně dospět. Je také vhodné zmínit, že použití a provázanost artefaktů souvisejících se správou změn umožňuje analýzu přesnosti odhadů pracnosti (a tedy postupné zlepšování plánování), vytváření statistik aktivity a podílu členů týmu (a tedy vyhodnocení týmové dynamiky), a sledování průběhu projektu (a tedy případné korekce směru). Všechny tyto aspekty jsou užitečné jak pro tým, z praktického hlediska vlastních prací na projektu a z hlediska procesu učení, tak pro mentory, z hlediska pedagogického působení na studenty a jejich hodnocení.

3.3.2 Podrobnosti k některým artefaktům

Vize projektu a Specifikace požadavků (DSP) jsou výsledkem základní definice projektu resp. detailního sběru a analýzy požadavků, na základě komunikace mezi týmem a zákazníkem. Během těchto aktivit mohou být použity vcelku libovolné techniky (strukturovaný rozhovor, analýza stávajícího systému a/nebo dokumentace, debata nad modely, prototypování, ...). V obou dokumentech je možné použít případy užití a UML diagramy, ale pouze pokud tyto nástroje přinášejí v projektu dostatečnou hodnotu (tj. nejsou povinné) a jsou formálně správně vytvořené.

Je třeba zdůraznit, že DSP je nepovinným artefaktem—v ASWI procesu je místo něj dávána přednost zaznamenání zjištěných (a postupně přidávaných či upravovaných) požadavků do product backlogu. V něm jsou také udržovány priority požadavků, časový plán a stav jejich realizace. Z backlogu musí proto být možné získat kompletní seznam a popis nasbíraných požadavků včetně informace, zda byly do výsledného produktu implementovány.

Pokud se tým rozhodne DSP vytvořit, musí jeho obsah být v souladu s product backlogem, který je považován za referenční. Musí být také zřejmé (případně v interní technické dokumentaci dostupné mentorovi dokumentované), jakou formu popisu požadavků používá tým jako primární – use cases, user stories, RUP struktury, apod.

Popis architektury je výsledkem analýzy požadavků a ověření jedné či více variant možné technické realizace. Během těchto aktivit je v ASWI procesu doporučeno využít konceptu spustitelné architektury [12], tj. vytváření návrhu řešení nikoli pouze v podobě dokumentační (UML diagramy a podobně), ale v podobě funkční kostry aplikace ověřující klíčové funkcionality a vlastnosti.

Dokumentace architektury pak zachycuje podstatné informace nutné pro pochopení implementace vyvíjeného a dokončeného produktu—základní prvky realizace, např. klíčové třídy, celkovou strukturu produktu včetně vazeb mezi jejími součástmi, a důležitá pravidla platná pro implementaci. Je obvykle výhodné použít UML modely (tříd, sekvencí, komponent a nasazení, atd. opět za předpokladu formální správnosti diagramů) ale použití této notace není podmínkou, neboť podstatnější je celková srozumitelnost a zachycení všech důležitých informací.

Pracovní položky plánu iterace jsou udržovány v *systému správy změn*, který má v procesu stěžejní roli. Položky mají těsnou vazbu na specifikace (Vize, DSP, backlog) na jedné straně, a sady změn zdrojových kódů (changesety) vytvářeného produktu spolu s jejich testy na straně druhé. Tím se v systému pro správu změn setkávají technické a manažerské aktivity a potažmo všichni členové týmu.

Je proto očekávána provázanost pracovních položek s *changesety v úložišti*, odhady pracnosti v plánech iterace, testy a testovacími protokoly, jakož i řetězec odvození "klíčový aspekt (funkčnost, vlastnost) ve vizi projektu -> položky v product backlogu -> pracovní položky iterace". Konkrétní životní cyklus a strukturu pracovních položek vč. chybových hlášení si musí tým (do)definovat sám, přičemž je použit RUP pro nastavení základních pravidel.

Manažerské artefakty mají povětšinou povahu pouhého záznamu schůzek týmu. Důvodem pro tuto jistou formální zbytnělost (která je v protikladu s agilním zaměřením procesu) je fakt, že mentor nemůže být přítomen většině schůzek týmu a přitom potřebuje získat určitý přehled o jeho fungování. Při definici těchto artefaktů byla proto snaha v co největší míře využít možnosti poskytované softwarovými nástroji, které jsou používány pro podporu vedení

projektu (viz následující oddíl). Retrospektivu iterace je možné provádět s pomocí jednoduchého dotazníku, kterým je tým směrován na zhodnocení podstatných aspektů iterativního procesu a dosažení milníků.

Na druhou stranu je vhodné zmínit explicitní absenci plánu projektu, motivovanou malým rozsahem a agilní povahou procesu. Plán pro projekty této velikosti je efektivně nahrazen popisem vize projektu a plány iterací, které s ní musí být v souladu.

4 Podpora nástroji pro vedení projektů

Jako v jiných případech je i pro ASWI proces vhodné či nutné využít podporu nástroji, přičemž nutnost vést a hodnotit velké množství souběžných projektů vede k preferování softwarových nástrojů oproti fyzickým pomůckám typu papírový backlog či nástěnka s burn-down grafem (přestože z hlediska vlastní práce na projektech by fyzické nástroje byly často vhodnější, viz např. [17]). V našem případě potřebujeme nástroji pokrýt jak technické tak manažerské aktivity, přičemž druhá oblast nemá vytvářet zvláštní nároky na nástroje či týmy ale má využívat data získávaná během standardních vývojových prací (tedy první oblasti). Motivací je přiblížit studentské projekty reálným projektům používajícím agilní metodiky, a to i co do použitého způsobu a podkladů pro konzultace a hodnocení.

Vzhledem k charakteristikám projektů uvedeným v oddíle 2 tedy ASWI proces potřebuje nástroje pro vedení správy změn (bug tracking apod.) umožňující členění úkolů do skupin (verze, komponenty), základní práci s odhady pracnosti úkolů a plánování a sledování průběhu iterace i projektu (statistiky, grafy typu burn-down chart). Dále je zapotřebí mít k dispozici verzující úložiště pro vytvářené artefakty a jednoduchý prezentační nástroj pro webové stránky týmu. Vzhledem k množství studentů a kontextu univerzitního prostředí je navíc žádoucí, aby nástroje umožňovaly integraci s některými klíčovými systémy univerzity, zejména z pohledu autentikace.

Pro výuku předmětu ASWI jsou konkrétně použity tyto nástrojové sady:

- IBM Rational TeamConcert (RTC) verze 2 [9] — sada nástrojů pro integrované vedení vývojových prací a plánování projektů, pokrývající všechny výše uvedené potřeby vyjma prezentačních stránek projektu.
- Redmine[3] a Subversion — open source alternativa k sadě RTC, umožňující správu změn a verzí (dříve používaný Flyspray[4] neposkytuje všechny potřebné funkcionality).
- Wiki (použit klon PmWiki[5]) — pro webové stránky projektů včetně zpřístupnění koncových verzí artefaktů

[3] http://www.redmine.org/
[4] http://www.flyspray.org/
[5] http://www.pmwiki.org/

Pro nástrojovou sadu RTC byla v rámci diplomové práce [4] vytvořena a v předchozím ročníku týmy rutinně použita šablona ASWI procesu podrobně definující strukturu týmu, průběh iterací a obsah položek správy změn. Na začátku semestru probíhá také úvodní zaškolení týmů do práce s oběma sadami nástrojů, a pro RTC je poskytnuta základní uživatelská podpora ze strany cvičících. Přes počáteční dojem, že RTC je příliš složitý a zbytečně komplexní nástroj, většina dosavadních týmů po zaučení ocenila a uměla efektivně využívat možnosti nástrojem poskytované.

Zároveň je z pedagogických důvodů nadále významně využíván Rational Unified Process jako znalostní báze konkrétních návodů pro jednotlivé koncepty, praktiky a artefakty. V případě potřeby vytváření UML modelů je doporučován nástroj MagicDraw, pro běžnou agendu předmětu (informace studentům, centrální sběr všech artefaktů ukončených projektů) je využíván systém Courseware ZČU.

5 Závěr

Agilní softwarový proces vytvořený pro potřeby menších výukových projektů předmětu ASWI, vedených studentskými týmy, byl dosud úspěšně použit pro vedení 16 softwarových projektů.

Na základě zhodnocení procesu s team leadery a dalšími studenty v roce 2010 je možno konstatovat, že pomáhá studentům lépe pochopit a prakticky provést iterativně řízený projekt, adaptivně plánovat jeho postup, agilně reagovat na změny požadavků či kontextu, a ocenit výhody těchto přístupů. Nasazení ASWI procesu včetně s ním spojených úprav teoretické části předmětu se odrazilo také ve velmi kladném zhodnocení výuky agilních metodik na mateřské fakultě autora procesu, které je součástí nedávno provedené Analýzy výuky agilních metod vývoje softwaru na vysokých školách v ČR [11].

Reference

[1] Scott Ambler. The agile unified process, 2005.

[2] Kent Beck and Cynthia Andres. *Extreme Programming Explained: Embrace Change*. Addison-Wesley Professional, 2nd edition edition, 2004.

[3] Barry Boehm. Anchoring the software process. *IEEE Software*, 13:73–82, 1996.

[4] Jan Boháč. Konfigurace metodik v rational team concert. Master's thesis, Katedra informatiky a výpočetní techniky, Západočeská univerzita, 2010.

[5] Premek Brada. Agilní proces pro výuku softwarového inženýrství. In *Sborník konference Objekty 2010*, Ostrava, ČR.

[6] Alena Buchalcevová. Where in the curriculum is the right place for teaching agile methods? In *Proceedings 6th ACIS International Conference on Software Engineering Research, Management & Applications (SERA 2008)*, page 205–209, September 2008. ISBN 978-0-7695-3302-5.

[7] Mike Cohn. *Succeeding with Agile: Software Development Using Scrum.* Addison-Wesley Professional, 2009. ISBN 978-0321579362.

[8] Eclipse Foundation. OpenUP - a lean Unified Process. Accessed October 2010.

[9] IBM Corporation. Rational team concert - product website. Accessed October 2010.

[10] Henrik Kniberg. *Scrum and XP from the Trenches.* C4Media, Publisher of InfoQ.com, 2007.

[11] Vojtìch Košák. Analýza výuky agilních metod vývoje softwaru na vysokých školách v Čr. Bakalářská práce, Katedra informačních technologií, Vysoká škola ekonomická v Praze, 2010.

[12] Philippe Kruchten. *The Rational Unified Process: An Introduction.* Addison-Wesley Professional, 3rd edition edition, 2003.

[13] Craig Larman. *Agile and Iterative Development: A Manager's Guide.* Addison-Wesley, 2004.

[14] Integrated Software & Systems Engineering Curriculum (iSSEc) Project. Curriculum guidelines for graduate degree programs in software engineering. Technical report, Stevens Institute of Technology, 2009.

[15] Pierre N. Robillard, Philippe Kruchten, and Patrick d'Astous. *Software Engineering Processes: With the UPEDU.* Addison-Wesley, 2003.

[16] Ken Schwaber and Mike Beedle. *Agile Software Development with Scrum.* Prentice Hall, 2001.

[17] Dave West and Jeffrey S. Hammond. The Forrester Wave: Agile Development Management Tools, Q2 2010. Technical report, Forrester Research, Inc., May 2010.

[18] Carl Wiegers. *Software Requirements.* Number ISBN 0-7356-1879-8. Microsoft Press, 2nd edition edition, 2003.