VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

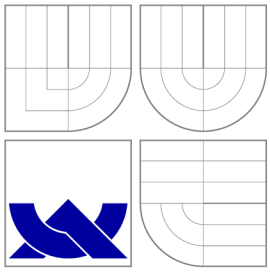
CERTIFIKACE CMMI VE VÝVOJI SOFTWARE V AGILNÍM PROSTŘEDÍ

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

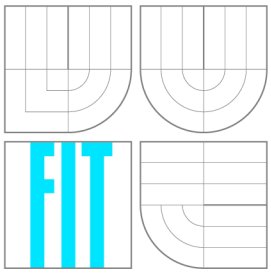
AUTOR PRÁCE
AUTHOR

Bc. RADEK GAJDUŠEK

BRNO 2013



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

CERTIFIKACE CMMI VE VÝVOJI SOFTWARE V AGILNÍM PROSTŘEDÍ

CMMI CERTIFICATION FOR SOFTWARE DEVELOPMENT IN AGILE ENVIRONMENT

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — Bc. RADEK GAJDUŠEK
AUTHOR

VEDOUCÍ PRÁCE — doc. RNDr. JITKA KRESLÍKOVÁ, CSc.
SUPERVISOR

BRNO 2013

Abstrakt

Cílem diplomové práce „Certifikace CMMI ve vývoji software v agilním prostředí“ je studium modelu kvality CMMI se zaměřením na softwarový vývoj v agilním prostředí ve společnosti Siemens. Práce popisuje do hloubky model kvality CMMI a představuje specifika agilního vývoje. Hlavní částí je analýza současného stavu procesů uvnitř společnosti, jejímž výstupem je seznam oblastí, které aktuálně vyžadují zlepšení tak, aby společnost dosáhla požadovaného stupně certifikace modelu CMMI. Procesní oblasti, které je třeba dále rozvíjet, jsou společně se seznamem možných zlepšení představeny konzultantovi práce a jsou diskutována vhodná řešení. V rámci implementační části je pak realizována webová aplikace, která přináší zlepšení do daných procesů. Přínos zavedených zlepšení je objektivně vyhodnocen prostřednictvím interního auditu. Součástí práce je také diskuse možného dalšího vývoje aplikace a rozvoje standardu v této společnosti.

Abstract

The goal of master thesis „CMMI Certification for Software Development in Agile Environment“ is CMMI quality model research with focus on software development in agile environment in the Siemens company. In the beginning CMMI model and Scrum methodics are introduced. The core of this thesis is focused on the current state analysis. Output of the analysis is a list of potential areas that are currently not compatible with quality model requirements. These areas are to be improved for the company to achieve the desired CMMI certification level. Possible improvements are introduced to the consultant. During the implementation part a web application is realized helping to remove most of the identified imperfections. Application benefit is objectively evaluated by an internal audit. The work includes discussion of possible further application development and quality model standard evolution in this company.

Klíčová slova

Agilní vývoj, Certifikace, CMMI, CMMI-SW, Kvalita softwaru, Management kvality, Procesní oblasti, Procesy, Scrum, Vývoj softwaru

Keywords

Agile development, Burndown, Certification, CMMI, CMMI-SW, Process areas, Processes, Quality management, Scrum, Software development, Software quality

Citace

Radek Gajdušek: Certifikace CMMI ve vývoji software
v agilním prostředí, diplomová práce, Brno, FIT VUT v Brně, 2013

Certifikace CMMI ve vývoji software v agilním prostředí

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením paní doc. RNDr. Jitky Kreslíkové, CSc. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Radek Gajdušek
20. května 2013

Poděkování

Rád bych poděkoval své vedoucí, paní doc. RNDr. Jitce Kreslíkové, CSc., za její odborné vedení, rady a připomínky, díky kterým se podařilo diplomovou práci úspěšně dokončit. Dále bych také rád poděkoval konzultantovi práce ze společnosti Siemens, Ing. Janu Vernerovi, který byl po celou dobu řešení práce k dispozici a vždy byl připraven ochotně poskytnout potřebné informace. Nerad bych zapomněl také na mou rodinu, bez jejichž podpory bych práci nedokončil, proto i jim patří velký dík.

Obsah

Seznam obrázků

Seznam tabulek

Kapitola 1

Úvod

Předmětem této diplomové práce je analýza současného stavu kvality ve společnosti Siemens, za účelem identifikace procesních oblastí, které aktuálně vyžadují zdokonalení tak, aby společnost v připravovaném auditu dosáhla požadovaného stupně kvality podle modelu CMMI. Vybraná zdokonalení jsou pak realizována v rámci praktické části.

Úvodní kapitola práce se zabývá definicí modelu kvality CMMI, popisuje jeho vznik a detailně osvětluje dvě základní stupnice používané při klasifikaci procesních oblastí a procesů samotných. U jednotlivých procesních oblastí lze nalézt také informace, týkající se specifik agilního vývoje.

Ve druhé kapitole je rozebrána agilní vývojová metodika Scrum, používaná ve zkoumané společnosti. Nejprve jsou vysvětleny základní pojmy agilního vývoje, poté následuje výčet rolí členů Scrum týmu, nechybí objasnění časově ohraničených událostí, které tento model vývoje používá a na závěr je uveden výčet artefaktů, které jsou produkovány jako výstup v průběhu vývoje.

Následuje kapitola věnující se analýze současného stavu. V textu lze nalézt seznam procesních oblasti, které byly ze zkoumání vyřazeny společně s důvodem. V závěru kapitoly je uvedena sumarizace všech potencionálních míst pro zlepšení.

Vstupem další kapitoly je seznam oblastí, které vyžadují vylepšení z předchozí kapitoly. Možná vylepšení byla představena konzultantovi z analyzované společnosti a byly diskutovány příčiny nalezených nedokonalostí. Závěrem tohoto setkání je shrnutí hlavního problému společně s nástinem řešení.

Následující kapitola obsahuje specifikaci požadavků implementovaného řešení. Dále se také zabývá detailním návrhem aplikace a lze v ní nalézt seznam zvolených technologií společně s návrhem databáze.

Kapitola popisující samotnou implementaci představuje výslednou aplikaci a rozebírá její klíčové části. Závěrem je popsán způsob testování aplikace a je uveden kompletní soupis provedených testů.

Závěrečná kapitola obsahuje zhodnocení diplomové práce a dosažených výsledků. Tato část předkládá závěry interního auditu, který objektivně hodnotí přínos této práce. Rovněž lze zde nalézt zamyšlení nad budoucím možným vývojem aplikace a rozvojem modelu CMMI ve zkoumané společnosti.

Tato diplomová práce přímo navazuje na semestrální projekt ze zimního semestru, který se zabýval analýzou modelu CMMI, agilní metodikou Scrum a vyhodnocením stavu kvality ve společnosti Siemens. Výstupem byl seznam možných zlepšení, která byla základem implementační části. Uvedené kapitoly byly proto převzaty i do textu této práce.

Kapitola 2

Model CMMI

Následující kapitola představuje model kvality CMMI, osvětluje pojmy jako je Stupeň vyspělosti organizací a zabývá se také stupnicí pro klasifikaci zralosti procesů. Ve druhé části jsou pak detailně rozebrány jednotlivé procesní oblasti společně s požadavky definovanými tímto modelem.

2.1 Popis modelu

CMMI je model kvality organizace práce určený pro vývojové týmy. Zkratku CMMI, která vznikla z anglického Capability Maturity Model Integration, lze volně přeložit jako Stupňovitý model vyspělosti. Jde o souhrn cílů a doporučených pracovních postupů pro vývojové týmy, které vedou ke kvalitnímu plánování a řízení prací s cílem produkovat co nejkvalitnější výstupy. Autorem modelu je tým pracující při Carnegie Mellon University, konkrétně jejich Software Engineering Institute, zkráceně SEI-CMU. Model se poprvé objevil před 25 lety a jeho definice je volně dostupná na internetu. [4]

Model CMMI nelze chápat jako návod, jak úspěšně realizovat kvalitativní požadavky současného trhu. Součástí definice je seznam toho, co by se mělo provádět, ale neposkytuje přesné návody ani postupy, jak toho dosáhnout. Jedná se z velké části spíše o doporučení.

Model definuje seznam procesních oblastí a cílů, kterých musí organizace v každé oblasti dosahovat. Model má 5 úrovní vyspělosti a prostřednictvím auditu se hodnotí na jaké z úrovní kvalita práce týmu je. Procesy definované v jednotlivých procesních oblastech jsou klasifikovány podle Stupně zralosti.

Ze statistiky z roku 2010, kterou publikoval pan Yoram vyplývá, že počet společností, které nechávají prověřit úroveň a kvalitu svých procesů rok od roku stoupá. Z tabulky 2.1 je vidět, že počet posouzení se od roku 2007 do roku 2010 téměř ztrojnásobil [12]. V USA je dokonce certifikace CMMI vyžadována u většiny výběrových řízení, stejně jako u nás certifikace ISO 9001[1].

2.1.1 Stupeň zralosti

Stupeň zralosti (Capability level) se aplikuje na jednotlivé procesní oblasti a do nich spadající procesy. Společně se zdokonalováním procesních oblastí roste také úroveň podřízených procesů. Procesy lze rozdělit do čtyř stupňů, které lze klasifikovat stupnicí od 0 do 3. Daný

[1] http://www.iso.cz/?page_id=38

	2007	2008	2009	2010
Počet posouzení	1964	3113	4134	5499
Počet opakovaných posouzení	208	361	564	826
Podíl firem mimo USA	65%	69%	71%	74%
Počet zemí s auditovanými společnostmi	59	63	67	69

Tabulka 2.1: Počet prověření podle CMMI v letech 2007-2010.

stupeň je dosažen za předpokladu, že byl splněn obecný cíl daného stupně, který však vyžaduje splnění specifických cílů. Popis jednotlivých stupňů:

- **Stupeň 0 – Nekompletní**
  Jedná se o proces, který není vůbec vykonávaný nebo pouze částečně. Není uspokojen jeden nebo více specifických cílů a neexistuje žádný generický cíl.
- **Stupeň 1 – Vykonávaný**
  Jsou splněny všechny specifické cíle, které jsou požadovány k vytvoření produktu.
- **Stupeň 2 – Řízený**
  Jedná se o proces, který je vykonávaný, plánovaný a prováděný v přesně definovaném pořadí, což znamená, že jsou plněny specifické cíle ve správném sledu. Umožňuje řízení, kontrolu a umožňuje řídit a spravovat zaběhnuté postupy i v období stresu.
- **Stupeň 3 – Definovaný**
  Řízený proces, který je součástí standardních procesů společnosti. Proces je detailně popsán a je součásti globální definice firemní množiny procesů.

Hlavní rozdíl mezi stupněm 2 a 3 je v definici procesů. U stupně 2 může být definice odlišná od globální definice ve firmě, zatímco u stupně 3 by měla být v souladu s touto definicí. Procesy ve stupni 3 jsou také definovány mnohem více formálněji než u stupně 2.

2.1.2 Stupeň vyspělosti

Stupeň vyspělosti (Maturity level) slouží pro klasifikaci množiny procesních oblastí, respektive úrovně kvality celého projektu. Klasifikace je aplikována na procesní oblasti a jejich zdokonalování napříč celou společností. Pro každý stupeň je předdefinována minimální úroveň jednotlivých procesních oblastí, společně s minimální sadou procesních oblastí, které musí být splněny společně s oblastmi z nižšího stupně. Stupnice má 5 stupňů – od 1 do 5:

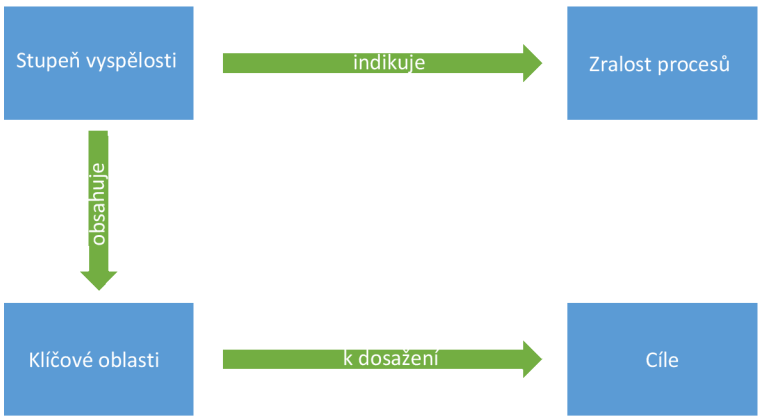
- **Stupeň 1 – Vykonávané**
  Společnosti pouze existují bez jakýchkoliv zavedených procesů či jsou procesy nedostačující a chaotické. Úspěch organizace spočívá v jedincích, nikoliv v zavedených procesech. Produkty obvykle fungují, ale často dochází k překročení nákladů z důvodu absence plánování, těžce se vypořádávají s krizemi a také jen obtížně opakují úspěch (úspěch je především dílem náhody než pravidlem).
- **Stupeň 2 – Řízené**
  Projekty jsou zajištěny procesy, které jsou plánovány a vykonávány podle pravidel. Projekty vyvíjejí kvalifikovaní lidé, kteří k tomu mají adekvátní zdroje a produkují

kontrolované výstupy. Výstupy jsou sledovány, kontrolovány a revidovány. Stav práce lze kontrolovat, jsou naplánovány milníky apod. Společnosti odolávají stresovým obdobím, protože mají plán. Rovněž se zaměřují na plánování, konfigurační řízení, správu požadavků, kvalitu a řízení požadavků.

- **Stupeň 3 – Definované**
  Procesy jsou dobře definovány a popsány ve standardech společnosti, existují nástroje a metody pro podporu procesů. Standardy jsou sdíleny napříč společností a neustále doplňovaný a zdokonalovány. Rozdíl oproti stupni 2 je především v ustálené množině standardů napříč společností a souladu v jejich používání. Díky tomu dochází k mnohem většímu pochopení souvislostí mezi procesy a lepšímu globální plánování. Důraz je kladen také na vzdělávání a školení zaměstnanců (učení se od sebe navzájem, sdílení znalostí napříč skupinami, ponaučení se z chyb). Novinkou oproti předchozí kategorii je také podpora pro rozhodování a vyhodnocování.

- **Stupeň 4 – Kvantitativně řízené**
  Zaměření na kvantitativní kritéria kvality, důraz je kladen na potřeby zákazníka, koncových uživatelů, organizace a implementátory. Je prováděna statistická analýza a sběr dat, specifická měření pro vybrané podprocesy, kvalitní řízení a řízení projektů na základě úspěšnosti předchozích projektů. Pro určení nejhodnotnějších podprocesů pro podnikání jsou používány modely a statistické techniky. Rozdíl oproti předchozí oblasti je především v predikci výkonu, neboť díky statistice a analýze jsme schopni u podprocesů předvídat jejich budoucí vývoj.

- **Stupeň 5 – Optimalizované**
  V nejvyšším stupni pokračuje zlepšování stávajících procesů a jsou používány technologické inovace a zaměření se na brzkou detekci chyb. Dále jsou používány sofistikované statistické metody a kvantitativní techniky s cílem neustále zvyšovat kvalitu procesů a také produktů. Oproti předchozímu stupni dochází k průběžné analýze a sběru dat ze všech projektů, hledání nedostatků a přijímání měřitelných zlepšení výkonu. [3]



Obrázek 2.1: Znázornění modelu CMMI.

Schématické znázornění modelu CMMI z pohledu Stupně zralosti a vyspělosti lze nalézt na obrázku 2.1. Dosahovaná úroveň se zjišťuje formou auditů, které mohou být prováděny interně v rámci společnosti nebo externím subjektem. Audit může probíhat například pro-

Stupeň	Zralost	Vyspělost
0	Nekompletní	-
1	Vykonávaný	Vykonávané
2	Řízený	Řízené
3	Definovaný	Definované
4	-	Kvantitativně řízené
5	-	Optimalizované

Tabulka 2.2: Vzájemná kompatibilita jednotlivých Stupňů zralosti a vyspělosti.

střednictvím sady otázek, u nichž jsou uvedeny možnosti. Na základě odpovědí lze vygenerovat celkovou známku.

2.1.3 Ekvivalence Stupně zralosti a vyspělosti

Oba stupně definují cestu zlepšení procesů v organizaci a měření jejich úrovně, každý však jiným způsobem. Stupně 1 až 3 jsou stejné pro oba modely, protože jsou vzájemně propojené a částečně ekvivalentní. Vyspělost je definována pro celou množinu procesních oblastí, zatímco Zralost se váže jen ke konkrétní procesní oblasti. Vzájemnou ekvivalenci některých stupňů ukazuje tabulka 2.2.

2.2 Model pro vývoj software

Model CMMI existuje v několika variantách – CMMI-SE (System Engineering) – pro týmy vyvíjející komplexní systémy, CMMI-SW (software) – pro týmy vyvíjející software. Práce se zabývá právě druhým zmiňovaným.

Model pro vývoj softwaru zahrnuje celkem 22 procesních oblastí, které budou vyjmenovány pro lepší přehlednost u jednotlivých procesních kategorií. Z celkové počtu je 16 základních, 1 sdílená a 5 specifických pouze pro vývoj softwaru. Procesní oblasti jsou organizovány do 4 kategorií - procesní řízení, projektové řízení, vývoj a podpora. Nyní budou popsány všechny procesní oblasti modelu CMMI. U vybraných procesních oblastí bude zmíněno také více informací týkajících se agilního vývoje.

Diagramy vzájemného provázání jednotlivých procesních oblastí napříč všemi kategoriemi lze nalézt v příloze A této diplomové práce.

2.2.1 Procesní řízení

Zahrnuje aktivity napříč projektem, jako je definování, plánování, vývoj, implementace, monitorování, kontrolní mechanismy, oceňování, metriky a zdokonalování procesů. Do této kategorie spadá 5 procesních oblastí, které jsou dále rozděleny do dvou kategorií, a to základní a pokročilé.

Základní oblasti umožňují sdílení nejlepších postupů, organizačních procesů a vzdělávání napříč organizací a patří mezi ně tyto procesní oblasti:

- **Firemní definice procesů** (Organizational Process Definition – OPD)
- **Firemní procesní zaměření** (Organizational Process Focus – OPF)

- **Firemní vzdělávání** (Organizational Training – OT)

Naopak pokročilé přináší zvýšení Zralosti procesů za účelem dosažení lepší úrovně kvality a výkonnosti a řadíme mezi ně tyto oblasti:

- **Řízení výkonu v organizaci** (Organizational Performance Management – OPM)
- **Řízení procesního výkonu v organizaci** (Organizational Process Performance – OPP)

Firemní definice procesů

Tato procesní oblast zahrnuje sestavení a správu používané množiny organizačních procesů, standardů pracovního prostředí, pravidel a příruček pro jednotlivé týmy z globálního pohledu společnosti. Při definici standardů prvotně dochází ke stanovení množiny firemních procesů. Musí být také definovány šablony a atributy dokumentů, které jsou podstatné pro firemní množinu procesů. Tuto množinu je zapotřebí neustále spravovat a rozvíjet. Poté následuje definice a správa modelu vývoje softwaru, kterým může být například vodopádový, spirálový či inkrementální model atd. Součástí této oblasti může být rovněž úprava zvolené vývojové metodiky, firemních procesů, či vývojového cyklu potřebám konkrétní společnosti. Tyto odchylky musí být řádně zdokumentovány a musí být vytvořeny příručky pro orientaci v těchto odchylkách.

V rámci této procesní oblasti jsou také stanoveny globální metriky, které jsou získány z analýzy metrik. Můžeme definovat například odhad trvání (člověkohodiny) nebo odhad rozsahu produktu (počet řádků kódu nebo stránek).

Při ustavení množiny firemních procesů je potřeba sestavit vývojové a tréninkové plány, při definici standardu pracovního prostředí musíme pamatovat na pravidla zabezpečené práce a například na standardy používaného hardwaru a softwaru. Rovněž je potřeba definovat příručky pro jednotlivé týmy, které musí obsahovat popis struktury týmu, komunikační kanály, eskalační hierarchii, pravidla pro psaní kódu, vymezení zodpovědnosti apod.

Firemní procesní zaměření

Oblast se zabývá plánováním, implementací a zavedení zlepšení v oblasti organizačních procesů. Základem je důkladné porozumění aktuálním silným a slabým stránkám procesů či celé množině procesů v organizaci.

Nejprve je potřeba sestavit soupis příležitostí pro zlepšení procesů, toto vyžaduje nutnost definovat cíle, kterých chceme v rámci zlepšování dosáhnout (procentuální vyjádření, zlepšení kvality produktů dodávaných zákazníkovi) a také sestavit požadavky na procesy uvnitř organizace. Dále je potřeba periodicky stanovovat potřebu analýzy stávajících procesů a jejich silných a slabých stránek. Na základě analýzy a stanovených kritérii pak rozhodujeme o tom, které procesy budeme dále rozvíjet a zdokonalovat.

V rámci plánování a zavedení procesních zlepšení sestavíme akční plán, který definuje akce spojené se zavedením zlepšení. V něm lze nalézt kromě samotné definice cílů také použitou strategii a časový harmonogram.

Poslední fází je samotné zavedení vylepšení procesů, jehož základem je vypracovaný plán, kterým se celá akce řídí. V průběhu celého procesu zavádění nových nebo zlepšování stávajících procesů je nutné sledovat celý proces a v případě potřeby reagovat na problémy spojené se zaváděním. Zkušenosti se zaváděním lze opět vyhodnotit a vzít si je jako vstup pro další analýzu a jako náměty pro zlepšení.

Firemní vzdělávání

Tato oblast má za úkol rozvíjet znalosti a dovednosti lidí tak, aby mohli plnit přiřazené role efektivně a účinně.

V počátku dochází k sestavení strategického plánu vzdělávacích potřeb v závislosti na aktuální znalosti vývojářů. Plán je zpravidla sestavován na 2 až 5 let dopředu. Vzdělávání se může týkat jednak tvrdých dovedností (jako je vývoj, testování, architektura apod.), je však také nutné myslet na měkké dovednosti jako vedení a řízení lidí nebo komunikaci.

U definovaných plánů školení je vždy nutné se zamyslet nad tím, zda jsme schopni toto školení provádět v rámci kapacit společnosti nebo využijeme externích subjektů. Rovněž je nutné sestavit taktický plán, který v podstatě definuje parametry školení (použité metody, oblasti pokrývající školení apod.).

Pro každé školení je potřeba stanovit místo a způsob konání. Může se jednat o osobní setkání či poslední dobou velmi populární online školení. Pro každé školení je nutné připravit materiály, které mohou mít tištěnou nebo elektronickou podobu.

Neméně důležitou roli hraje výběr vhodných osob, kterých se bude školení dotýkat. V ideálním případě by mělo školení zaměstnancům přinést nové poznatky a pocit, že strávený čas nebyl zbytečný. Nezbytná je rovněž evidence provedených školení společně se seznamem osob, které tato školení absolvovaly.

Každé provedené školení by mělo v závěru obsahovat nějakou možnost zpětné vazby, aby bylo zajištěno neustálé zlepšování v této oblasti. Typickým příkladem může být závěrečný test nebo průzkum po absolvování školení. Odpovědi účastníku je potřeba shromažďovat a připomínky zapracovávat do budoucích školení.

Řízení výkonu v organizaci

Cílem této procesní oblasti je proaktivně spravovat výkon v organizaci, aby byla schopna efektivně plnit své obchodní cíle. Zaměřuje se na zvýšení kvality produktu, produktivity samotné, efektivitu procesů nebo na optimalizaci využívání zdrojů v organizaci.

Pro řízení se používají statistické a kvantitativní techniky. Díky nim jsme schopni nalézt nedostatky v procesním výkonu a identifikovat oblasti pro zlepšení procesů. Ze všeho nejdříve je nutné správně porozumět obchodní strategii společnosti a aktuálním výkonnostním výsledkům. Analýzou dat o procesním výkonu poté určujeme, zda je společnost schopna plnit stanovené obchodní cíle. Na základně této analýzy stanovíme potenciální oblasti, které je třeba zlepšit.

Ze všeho nejdříve je nutné provést kategorizaci navrhovaných zlepšení. V podstatě je lze rozdělit na inkrementální, které upravují stávající procesy a inovativní, které zavádějí zcela nové prvky. Analýzou pak definujeme přínos opatření v závislosti na firemních kvalitativních a procesních cílech. Vybrané cíle jsou pak validovány a je ověřováno, zda jejich přínos je skutečně takový, jak se očekává (prototyp, modelování a simulace). Zvolená zlepšení jsou poté implementována, při tomto procesu je nutné brát také v potaz začlenění do příslušných procesních oblastí.

Do této oblasti spadá rovněž sestavení plánu pro zavedení vybraných zlepšení. Následně dochází k zavedení zlepšení a vyhodnocení efektu ve světle kvality a procesního výkonu. Pro vyhodnocení se opět používají statistické a kvantitativní techniky.

Řízení procesního výkonu v organizaci

Zaměřuje se na sledování a řízení kvantitativních pohledů na výkon vybraných procesů z množiny standardních firemních procesů. Poskytuje podporu pro dosažení kvalitativních a výkonnostních cílů a zároveň poskytuje data o výkonnosti procesů, základní parametry a modely pro kvantitativní řízení procesů.

Nejprve je nutné nadefinovat kvantitativní a výkonností cíle, které lze sledovat v závislosti na obchodních cílech organizace. Příkladem může být schopnost dodání produktu včas a s odhadovanými náklady. Poté vybereme standardní procesy z množiny procesů, které zahrneme do analýzy výkonnosti a sledování obchodních cílů. Pro vybrané procesy stanovíme metriky, které budou zahrnuty do výkonnostní analýzy.

Následně analyzujeme zvolené procesy za použití definovaných metrik a stanovíme základní parametry výkonnosti procesů (například komplexnost, velikost v kontextu týmu). Výsledkem je sestavení a správa výkonnostních modelů pro množinu standardních procesů v organizaci. Zde se pro simulaci používají například regresní testy nebo metoda Monte Carlo[2]. U modelů simulujeme například situaci, že se změní proces nebo cíl společnosti.

2.2.2 Projektové řízení

Mezi aktivity projektového řízení se zahrnují všechny manažerské aktivity spojené s plánováním, monitorováním a kontrolou projektu. Opět se dělí do dvou kategorií, na základní a pokročilé. Základní kategorie sdružuje aktivity spojené s řízením projektu, které se týkají sestavení a správou projektového plánu, stanovením a správou požadavků, sledování postupu projektu vzhledem k plánu a správou subdodávek. Do této kategorie řadíme oblasti:

- **Monitorování a kontrola projektu** (Project Monitoring and Control – PMC)
- **Řízení požadavků** (Requirements Management – REQM)
- **Projektové plánování** (Project Planning – PP)
- **Řízení subdodavatelů** (Supplier Agreement Management – SAM)

Za pokročilé aktivity lze označit takové, které jsou spojené s definicí procesů uzpůsobených na míru potřebám konkrétní firmy a jsou součástí globální definice společnosti. Dále oblasti zabývající se stanovením standardů vývojových prostředí, komunikací a koordinací práce mezi lidmi, problematikou sestavování týmů, kvalitativním řízením a řízením rizik. Mezi ně patří:

- **Kvantitativní projektové řízení** (Quantitative Project Management – QPM)
- **Řízení rizik** (Risk Management – RSKM)
- **Integrované projektové řízení** (Integrated Project Management – IPM)

Monitorování a kontrola projektu

Hlavním cílem je sledování stavu postupu prací na projektu a včasná detekce situace, kdy se projekt dostává do stavu, kdy neběží podle plánu.

[2] http://www.chem.unl.edu/zeng/joy/mclab/mcintro.html

Velice důležitou součástí je sledování nákladů vývoje, ať už se jedná o čas nebo cenu. Je proto velice důležité mít tyto výdaje pod kontrolou a je potřeba je sledovat periodicky. Nezbytnou součástí je monitorování rizik spojených s projektem, jejich pravidelná kontrola a zaznamenávání. Rovněž dochází ke kontrole projektových dat, zda odpovídají projektovému plánu. Projektový plán nesmí také zapomínat na zapojení zainteresovaných stran, především na řízení jejich požadavků a připomínek.

V agilním vývoji je zákazník pro úspěch produktu klíčový, protože jen on je schopen definovat své neustále měnící se požadavky. Agilní vývoj je schopen reagovat na požadavky zákazníka ve velmi krátkém čase, což je jeho nespornou výhodou.

Pro monitorování je vhodné provádět kontroly jednak postupu projektu, výkonnosti v závislosti na zbývajících úkolech. Ve větších časových úsecích se pak provádí revize milníků.

Provedení nápravných opatření nastává ve chvíli, kdy dochází k výrazné odchylce oproti projektovému plánu, vždy je nutná důkladná analýza, která je základem pro přijatá opatření. Provedená opatření se mohou projevit změnou projektového plánu, zpřesněním odhadů, přehodnocením objemu akceptované práce či dokonce úplným zastavení projektu.

Řízení požadavků

Úkolem této procesní oblasti je spravovat požadavky na produkt nebo jeho součásti a zajistit provázání mezi těmito požadavky a projektovým plánem.

V agilním vývoji jsou požadavky komunikovány a sledovány prostřednictvím mechanismů jako je Produktový katalog. Přiřazení k řešení požadavků je realizováno v pravidelných intervalech za účasti celého týmu (Denní nebo týdenní porada).

Ze všeho nejdříve je nutné porozumět požadavkům zákazníka. Správně definovaný požadavek by měl mimo jiné být kompletní, být v souladu s architektonickou a kvalitativní specifikací, být testovatelný a musí být definována jeho priorita zákazníkem. Literatura uvádí, že na počátku projektu je známo přibližně 50% požadavků.

Požadavky musí být správně zaznamenány a každý účastník projektu musí s nimi souhlasit. Jelikož se požadavky neustále mění, je nutné v průběhu trvání projektu tyto požadavky na změny zaznamenávat a řídit. Rovněž je také nutné sledovat, zda projektový plán odpovídá skutečnému stavu požadavků a zajišťovat obousměrnou synchronizaci mezi požadavky a projektovými položkami. Pro jednotlivé produktové položky je také potřeba udržovat provázání s požadavky, aby bylo možné je vzájemně dohledávat.

Dobrým základem pro efektivní správu požadavků mohou být nápady a poznatky Toma Gilba, uvedené v jeho knize „Competitive Engineering“ [6], které tvoří dobrý základ pro splnění požadavků úrovně 4 modelu CMMI.

Projektové plánování

Následující oblast zahrnuje sestavení a správu plánů pro všechny projektové aktivity. Plánování zahrnuje nejen odhady trvání, ale také rozhodování a přidělení zdrojů, identifikaci a analýzu rizik.

V agilním prostředí se veškeré činnosti spojené s plánováním provádějí mnohem častěji než v tradičních vývojových prostředích. Ve chvíli, kdy je stanoven plán projektu jako celku, týmy provádějí odhady a plánování úkolů spojených s aktuální iterací. Týmy se soustředí na vývoj požadovaný pro aktuální iteraci a neberou v potaz produkt jako takový. Kontext projektu se bere v potaz pouze u rizik nebo omezení projektu. Cílem iterace je dokončit všechny požadované Uživatelské scénáře a Úkoly s nimi spojené. Výsledkem Sprintu je doručení nové verze aplikace a následně dochází k výběru nových Uživatelských scénářů

z katalogu a celý proces se opakuje. Členové týmu zpravidla vůbec nevědí, na čem budou další den pracovat, mají pouze globální informaci o tom, co je třeba stihnout ve Sprintu. Plánování, monitorování a úprava plánů je prováděna vždy, kdy je to zapotřebí (například Denní porada).

Základem je sestavení hierarchické struktury činností (WBS)[3], kdy definujeme logické celky projektu, které mohou být odhadovány na úkoly, které lze sledovat a řídit. Rovněž rozhodujeme, zda není vhodnější je poptávat externě než je sami vyvíjet.

Další úkolem je stanovení a odhadů pro jednotlivé části produktu a úkoly k nim přiřazené, možnými atributy mohou být například počet řádku kódu, počet architektonických elementů nebo počet částí. Nezbytná je také definice fází vývojového cyklu, u kterých bude probíhat plánování. Ohodnocení pracnosti a ceny pro jednotlivé části a úkoly podléhající těmto částem. Pro toto oceňování se často používají osvědčené metody odhadů, ale také zkušenosti týmu z předchozích iterací. Výsledné ocenění je kombinací mnoha kritérií, které musí být zohledněny.

Při sestavování projektového plánu můžeme použít například Metodu kritické cesty (CPM)[4] nebo metodu PERT[5] pro stanovení posloupnosti činností a priorit vývoje. Stanovují se také jednotlivé milníky, časový plán vývoje, jednotlivých iterací a také datum předání produktu zákazníkovi.

Během plánování je potřeba brát v potaz také rizika, která mohou nepříznivě projekt ovlivnit, proto je třeba je nejen identifikovat, ale především řídit a počítat s nimi při sestavování plánu. Nedílnou součástí je plánování řídících dat (požadavky na zabezpečení, seznam sbíraných dat apod.), pro které je také potřeba sestavit plán sběru a nakládání s nimi. Při plánování je také nutné brát v potaz dostupnost jednotlivých zdrojů, ať už se jedná o technické vybavení nebo personální (klasickým příkladem jsou právě dovolené). Tyto atributy je také třeba zahrnout do plánu.

U zaměstnanců je potřeba mít na paměti jejich znalosti a dovednosti a v případě potřeby plánovat jejich zaškolování nebo získávání nových dovedností nezbytných pro úspěch projektu. Při sestavování plánu nesmíme také opomenout naplánovat zapojení zainteresovaných stran. U nich se v podstatě může jednat o stejné aktivity jako byly uvedeny výše.

Výsledkem analýzy všech výše uvedených kritérií je pak projektový plán, který má formu oficiálního dokumentu a bere v potaz všechny potřeby projektu a pamatuje také na rizika s vývojem spojená. Obsahuje milníky, úkoly, rozpočet, rizika, apod. Dokumentů může být více a mohou být rozděleny například na softwarový plán, projektový a plán vývoje.

Plány je potřeba v pravidelně stanovených intervalech kontrolovat a ujišťovat se, že dochází k jeho dodržování všemi zúčastněnými stranami. V případě výskytu relevantních událostí (rizika, nedostupnost zdroje, . . .) je rovněž nutné vhodně reagovat a případně upravovat plány. Stejně tak je potřeba spravovat závazky zainteresovaných stran, aby bylo zajištěno fungování podle plánu.

Řízení subdodavatelů

Hlavním úkolem této procesní oblasti je řídit akvizice a služby poskytované dodavateli. Mezi služby poskytované dodavateli lze zařadit části vyvíjeného systému, ale také například dokumentaci, technické vybavení nebo softwarové vybavení.

[3] http://www.projektmanazer.cz/faq/co-je-wbs
[4] http://books.fs.vsb.cz/SystAnal/texty/25.htm
[5] http://en.wikipedia.org/wiki/Program_Evaluation_and_Review_Technique

Ze všeho nejdříve je potřeba stanovit pro každý produkt nebo jeho produktové součásti ty položky, které budou získávány formou subdodávek od dodavatelů. Pro jednotlivé položky je pak nutné najít a vybrat vhodného dodavatele. Při výběru je potřeba mít na paměti jejich schopnost uspokojit naše požadavky a stanovená kritéria. Při výběru se bere v potaz například jejich geografická poloha, zkušenosti s podobnými dodávkami, citlivost na výkyvy trhu apod. S vybraným dodavatelem je nutné uzavřít smlouvy o dodávkách, které kromě definice rozsahu požadovaných dodávek, ceny a přejímacích kritérií, musí obsahovat také sankce v případě nedodržení termínů.

Ve stanovené dohodě jsou definovány aktivity, které musí dodavatel v průběhu dodávek provádět. Mohou mezi nimi být například hlášení o stavu vývoje, dokumentace apod. Typickými oblastmi sledování je dodržování časového harmonogramu, plánu a ceny. Lze také sledovat kompatibilitu s klíčovými prvky systému. Před samotným převzetím subdodávky se musíme ujistit, že předávaná část splňuje akceptační kritéria, stanovená opět ve smlouvě o dodávkách. Pokud produkt splňuje všechna kritéria dochází k jeho začlenění do našeho produktu. V tuto chvíli je nutné realizovat plány začleňování, případná školení zaměstnanců a podporu ze strany subdodavatele v případě potřeby. Rozsah podpory a školení je obvykle zakotven ve smlouvě.

Kvantitativní projektové řízení

Cílem této procesní oblasti je kvantitativně řídit projekt tak, abychom byli schopni dosahovat stabilních kvalitativních cílů, včetně stabilní výkonnosti procesních cílů.

Počátkem celého procesu je stanovení kvalitativních a výkonnostních cílů procesů a projektu. Příkladem může být například ukazatel Výkonnosti týmu v agilním prostředí. Za použití statistických a kvantitativních technik seskupujeme definované procesy, které mají vliv na dosahování sledovaných cílů.

U zvolených procesů vybíráme jejich podprocesy a atributy, které jsou kritické pro výkonnost procesů a zároveň jsou klíčové pro dosažení požadovaných cílů. Pro tyto podprocesy a atributy vybereme vhodné metriky a analytické techniky. Příkladem může být požadavek na 75-100 vyprodukovaných řádků kódu za hodinu.

Za použití kvantitativních a statistických technik sledujeme, zda jsou plněny definované projektové cíle v oblasti kvality a výkonnosti procesů nebo nikoliv. Doporučuje se také provádět Root Cause analýzu[6].

Řízení rizik

Cílem je identifikovat možná rizika před jejich výskytem. Pokud jsme schopni rizika v tuto chvíli identifikovat, není pak problém je plánovat a v případě výskytu daného rizika aktivovat plánovaná opatření s cílem jej zcela minimalizovat nebo naopak alespoň dostatečně zmírnit.

V agilním prostředí dochází k mnohem systematičtějšímu přístupu k řízení rizik. Rizika jsou přímo zahrnuta jako součást plánování a rytmu týmu. Při plánování iterací, odhadů úkolů a přijímání dokončených úkolů se s nimi běžně počítá.

Pro řízení rizik je nezbytné nejprve stanovit potenciální zdroje možných rizik a ty zařadit do kategorií. Jako příklad zdroje rizika lze uvést nedostupnost některého ze zdrojů, změnu legislativy či nemožnost uspokojení požadavků zákazníka. Rizika mohou být technická nebo například rizika spojená s projektovým řízením. Rovněž je nutné definovat parametry po-

[6] http://www.mindtools.com/pages/article/newTMC_80.htm

užívané ke kategorizaci rizik a sledování úsilí nutného pro jeho odstranění. Výsledkem je sestavení strategie řízení rizik.

Na počátku celého procesu je nejprve nutné rizika identifikovat a vhodně zdokumentovat. Při identifikace lze využít poznatků z podobných projektů nebo lze také využít experta na tuto problematiku. Rizika se mohou dotýkat projektu přímo (nedostatek zdrojů) nebo také nepřímo (stávky, úmrtí). Ve chvíli kdy máme vytvořen seznam potenciálních rizik je nutné tato rizika ohodnotit, zařadit je do příslušné kategorie a přiřadit jim relativní prioritu.

Pro každé potencionální riziko je potřeba sestavit plán na zmírnění rizika v závislosti na strategii řízení rizika ve společnosti. Rovněž je potřeba periodicky monitorovat stav každého rizika a v případě jeho výskytu zajistit provedení předem definovaného plánu na zmírnění rizika.

Integrované projektové řízení

Procesní oblast zahrnuje sestavení a vedení projektu a všech zainteresovaných stran pomocí integrovaných a definovaných procesů, které odpovídají procesům definovaných pro danou organizaci. Součástí této procesní oblasti jsou rovněž aktivity spojené se samotným vývojem, ale také se zabývá kontaktem se zákazníkem a zákaznickou podporou, sledováním akvizicí a podpůrnými aktivitami jako je dokumentace, školení, marketing či konfigurační řízení.

Jako první je potřeba sestavit definici projektových procesů, kdy definujeme rozsah a složitost projektu, vývojový cyklus, rozsah podpory pro zákazníka, dochází také k definici šablon dokumentů, komunikačních kanálů, modelů používaných pro odhady.

Poté následuje určení plánovacích kritérií pro odhadování projektových aktivit. Tento proces je nutné neustále rozvíjet na základě zkušeností z minulých iterací, používají se historická data či data získaná z podobných projektů, zároveň se také bere v potaz aplikační doména, zkušenosti jednotlivých lidí nebo také omezení prostředím. Mezi sledovaná kritéria lze zařadit cenu, dobu vývoje, počet chyb nebo přiřazení jednotlivým osobám.

V rámci této oblasti dochází ke zvolení používaného vývojového prostředí, nástrojů pro testování, nástrojů pro podporu rozhodování, zkrátka veškerých nástrojů potřebných pro vykonávání práce.

Všechny plány, které je třeba vypracovat a používat pro řízení, jako jsou strategie pro řízení rizik, dokumentační plán, vzdělávací plány zaměstnanců, plány kvality je potřeba zaintegrovat do plánů organizace.

Používání sestavených plánů vyžaduje monitorování stavu a průběhu projektu na základě definovaných plánů a vhodných reakcí na nastalá zjištění, jako je například změna plánů, přijetí opatření pro zmírnění nastalého rizika či dokonce ukončení projektu v krajním případě.

Samotné sestavení týmu je aktivita, která definuje složení týmu, které musí být zaznamenáno ve firemních dokumentech, každý zaměstnanec musí mít roli v týmu a být schopen pravidelně podávat zprávy o své práci.

Musíme také pamatovat na nutnost neustálé revize a kontroly definovaných procesů prostřednictvím sledovaných metrik, odhadů a plnění plánu – používají se metody seznamů, tréninkových modulů atd.

V rámci této kategorie je zdůrazňována potřeba řízení zainteresovaných stran zapojených do projektu. U těchto skupin je nutné spravovat závislosti, především zajistit jejich identifikaci, dohodu a sledovat kritické závislosti. Rovněž je potřeba řešit s těmito stranami problémy, který mohou nastat, jako je zpoždění subdodávek nebo nedostupnost kritického zdroje pro projekt.

2.2.3 Vývoj

Do oblasti vývoje lze zařadit všechny aktivity vývoje a údržby, které jsou sdíleny napříč všemi vývojářskými disciplínami. Tato kategorie rovněž zahrnuje také validaci a verifikaci, což jsou disciplíny, které dokáží uspořit nemalé množství finančních prostředků, jsou-li vykonávány pečlivě. Do této kategorie patří následující oblasti:

- **Integrace produktu** (Product Integration – PI)
- **Požadavky na vývoj** (Requirements Development – RD)
- **Technické řešení** (Technical Solution – TS)
- **Validace** (Validation – VAL)
- **Verifikace** (Verification – VER)

Integrace produktu

Procesní oblast se zaměřuje na výsledné sestavení celého produktu z jednotlivých komponent do jednoho celku, má za úkol kontrolovat, zda se výsledný produkt po integraci součástí chová korektně (splňuje kvalitativní požadavky a disponuje požadovanou funkcionalitou). Rovněž se zabývá doručením produktu zákazníkovi.

V agilním prostředí je integrace velmi častým jevem, zpravidla denní záležitostí. Často v tomto kontextu hovoří o kontinuální integraci, kdy dochází k neustálému vývoji a aplikace je jako celek v každém okamžiku aktualizace funkční a ve stavu, kdy je možné ji doručit zákazníkovi. Rozhraní a integrační strategie je definována již v rané fázi vývoje produktu, a je schopna reagovat velice dobře na příchozí požadavky týkající se změny datové vrstvy či rozhraní. Proto je velice dobře realizovatelná.

Nejprve je potřeba stanovit integrační strategii, kdy jsou definovány intervaly sestavování produktu (mohou být například jednou za týden, či okamžité). Rovněž je vhodné naplánovat spouštění testů, architekturu či koncept zachytávání výjimek apod. Tato oblast zahrnuje popis kritérií a integrační strategie.

Rovněž je nutné provést revizi definovaných rozhraní jednotlivých komponent tak, aby byla zajištěna jejich vzájemná kompatibilita. V průběhu vývoje produktu je potřeba neustále tuto skutečnost hlídat a v případě změn reagovat vhodnými úpravami.

V samotném závěru integračního procesu dochází ke kontrole připravenosti jednotlivých komponent pro integraci. Po ověření lze přejít k sestavení výsledného produktu podle integrační strategie za použití dříve definovaných procedur. Následuje validace a verifikace jednotlivých komponent jako celku a pokud je vše v pořádku, přecházíme k distribuci k zákazníkovi. Při distribuci produktu řešíme jednak způsob distribuce, kterým může být internet nebo pevné médium, tak licenční podmínky a ochranné známky.

Požadavky na vývoj

Cílem je zjištění, analýza a stanovení požadavků ze strany zákazníka, dále pak požadavky produktu jako takového a jeho produktových součástí. Požadavky ze všech tří kategorií jsou základem designu produktu. Z velké části se jedná především o zjištění a správné porozumění zákazníkovým požadavkům, určení priorit a definici kvalitativních požadavků. V rámci této procesní oblasti dochází také k definici základních atributů projektu, jako

je cena, harmonogram vývoje, technická nebo právní omezení a v neposlední řadě také k ohodnocení rizik spojených s projektem.

V agilním prostředí je třeba mít na paměti, že se požadavky zákazníka neustále vyvíjí, a proto je nutné je neustále zaznamenávat, analyzovat a formovat do podoby použitelné pro vývoj, tj. Uživatelské scenáře, Případy užití, položky Produktového katalogu, které jsou pak vstupem plánování v jednotlivých iteracích, kde dochází k přiřazení priorit a sledování rizik.

Na počátku dochází k získání informací o představách, očekáváních a přáních zákazníka. Získané informace jsou následně zaznamenány mezi požadavky pro vývoj. V této fázi se používají metody brainstormingů, prototypování grafického uživatelského rozhraní, průzkumy, beta testování apod.

Sběr požadavků na součásti produktu zahrnuje definici rozhraní jednotlivých části a definici požadavků na systém jako je například odezva systému nebo jeho dostupnost.

Analýzu a validaci požadavků provádíme za použití modelů, prototypů, demonstrací, čímž se ujišťujeme, že byly požadavky zákazníka správně pochopeny a jsme schopni je uspokojit v závislosti na jednotlivých součástech systému.

Technické řešení

Pro požadavky definované zákazníkem se snažíme navrhnout a implementovat vhodná řešení, která splňují požadavky zákazníka.

V agilním prostředí je kladen důraz na brzký průzkum možných řešení v závislosti na požadavcích. Správná rozhodnutí na základě těchto průzkumu napomáhají ke zlepšení kvality výsledného produktu. Řešení mohou být definována například soupisem funkcí nebo možností uživatele. V budoucnu správná analýza umožňuje rozšíření systému s relativně malým rizikem v podobě zásahu do struktury produktu.

Pro produktové části navrhneme několik řešení problému a na základě zvolených kritérií provedeme analýzu alternativních řešení, vybereme z nich to nejvhodnější. V potaz bereme parametry jako je riziko, cena či technologická omezení.

Zvolené řešení je pak dále rozpracováno, je stanovena architektura, vztahy mezi jednotlivými částmi, zásady komunikace součástí, zpracovávání chybových stavů apod. Pro lepší vizualizaci vztahů mohou být použity techniky Objektově orientovaného přístupu. Rovněž se často používají návrhové vzory. Součástí této fáze by měla být také analýza, zda je pro společnosti lepší danou součást systému vyvinout v rámci vlastních kapacit nebo ji nechat vyrobit u jiného subjektu.

Samotná implementace navrženého řešení by se měla řídit definicí uvedenou ve specifikaci. Nezbytnou součástí implementace musí být dokumentace jednotlivých částí, kódu a systému jako celku. Rovněž musí proběhnout testování částí, které je zárukou správné funkcionality jednotlivých komponent.

Validace

Smyslem validace produktu nebo jeho součástí je ověřit, zda se produkt a všechny jeho součástí chovají tak, jak očekáváme a zda je implementace v souladu se zpracovaným návrhem řešení.

Příprava na validaci spočívá ve výběru součástí k validaci a zároveň definuje validační metody, který budou použity. Měli bychom mít na paměti, že kromě validace kódu či funkcionality by mělo docházet také k validaci dokumentace, uživatelských manuálů a dalších artefaktů, které jsou výstupem technického řešení. Z metod pro ověření můžeme použít

například testování, demonstraci prototypu atd. Nesmíme také zapomenout stanovit kritéria hodnocení a validační procedury. Nezbytnou součástí je příprava testovacího prostředí. Testování může probíhat automaticky nebo také manuálně za účasti lidí s odpovídajícími znalostmi.

Samotné provedení validace probíhá metodami popsanými výše. Výstupem musí být validační zprávy pro jednotlivé součásti. Výsledek validace produktu jako celku je poté analýzou jednotlivých validací, spojených do finální zprávy shrnující celý proces.

Verifikace

Verifikace má za úkol zkontrolovat, zda vybrané části produktu odpovídají specifikaci požadavků. Často jsou v této fázi vývoje používány tzv. Vzájemné ohodnocení, kdy dochází ke kontrole odvedené práce jinou osobou než autorem, která je odborníkem v dané oblasti. Hojně používanou metodou je například párové programování.

Díky velmi časté integraci součástí v závěru každé iterace je verifikace stejně jako validace nezbytnou částí agilního vývoje. Systematické provádění napomáhá odhalení chyb v poměrně brzké době a na odstranění není proto potřeba vynakládat tak velké množství prostředků jako by tomu bylo v pozdějších fázích. Navíc může také docházet k tomu, že se prototyp z předchozí iterace v důsledku integrace nových součástí stane nevalidní.

Během přípravy na verifikaci dochází k výběru vhodných součástí k verifikaci, je definováno verifikační prostředí a stanoveny postupy a kritéria hodnocení. Jsou definovány typy prováděných testů (integrační, akceptační, ...) a testovací parametry.

Při samotném Vzájemném ohodnocení dochází ke kontrole vybraných součástí na základě sestaveného plánu (formou bodového seznamu). Sledují se definovaná kritéria, kterými může být udržovatelnost kódu, dodržování pravidel pro psaní kódu, dodržovaní standardů. Výstupem kontroly by měl být dokument, který shrnuje závěr provedené kontroly a obsahuje například množství chyb.

Verifikace vybraných součástí spočívá v kontrole, zda implementované součástí (kód, dokumentace, příručky) odpovídají požadavkům zákazníka, a zda splňují všechna kritéria. Výstupem je opět dokument o provedené kontrole.

2.2.4 Podpora

Do kategorie označené jako podpora spadají aktivity pro podporu vývoje a údržby softwaru. Tato oblast se dělí opět do dvou podkategorií, a to základní a pokročilé. Součástí základní podkategorie jsou procesy pro podporu vývoje, tzv. konfigurační řízení, dále analýza a měření a v neposlední řadě také objektivní ukazatele pro zhodnocení prováděné práce a služeb, které jsou tyto:

- **Konfigurační řízení** (Configuration Management – CM)
- **Analýza a metriky** (Measurement and Analysis – MA)
- **Zajišťování kvality produktu a procesů** (Project and Product Quality Assurance – PPQA)

Pokročilé oblasti jsou všechny ty, které mají za důsledek zvýšení Zralosti procesů, obě oblasti jsou velice závislé na vstupech, které jsou výstupy jiných procesních oblastí, řadíme mezi ně tyto oblasti:

- **Rozhodovací analýza a řešení** (Decision Analysis and Resolution – DAR)
- **Analýza příčin a řešení** (Causal Analysis and Resolution – CAR)

Konfigurační řízení

Oblast podpory vývoje se zabývá integrací produktových součástí za použití automatizovaných nástrojů a dále také poskytuje aktuální konfigurační data pro vývojáře, uživatele a zákazníka. Do této oblasti lze zařadit například hardware a zařízení, produktovou specifikaci, překladače, nástroje pro správu kódu a vývoj, automatický překlad kódu, správu testovacího prostředí, plánování a iterační katalog, ale také správu architektonických dokumentů a technických publikací.

Pro agilní vývoj je tato oblast velmi důležitá, z důvodu neustálé potřeby reagovat na požadavky vývoje, jako jsou frekventované či automatické sestavení, definice separátních pracovních prostorů pro vývoj (nová funkcionalita produktu, párové programování). Na počátku každé iterace nastává potřeba rekonfigurace, proto je nutné s touto oblastí při plánování počítat a zahrnout ji do plánování.

Stanovení základních parametrů spočívá v identifikaci konfiguračních položek a částí produktu, které budou pod správou konfiguračního řízení, výběr a spravování vhodného systému pro konfigurační a změnové řízení. Dále lze zde zahrnout definici plánu milníků nebo například plán nasazení částí produktu u zákazníka.

Do této oblasti také spadá sledování změn konfiguračních položek a jejich řízení, které by mělo být v ideálním případě podporováno systémem. Měla by být zajištěna dohledatelnost provedených změn pro všechny členy týmu.

Zajištění integrity spočívá v zajištění přístupu ke konfiguračním položkám pouze v závislosti na přidělených oprávněních. Spolehlivost těchto omezení by měla být kontrolována konfiguračním auditem, který má za cíl nalézt a odstranit nedostatky v konfiguraci.

Analýza a metriky

Definuje vhodné objekty pro měření a analýzu, pro které následně definuje použité techniky a mechanismy pro sběr metrik, jejich vizualizaci a zpětnou vazbu. Implementací definovaných mechanismů a technik dochází k nashromáždění potřebných dat, jejichž analýzou dochází k přijetí potřebných opatření.

Nejprve dochází k výběru vhodného objektu pro analýzu, například u projektového plánu můžeme sledovat hodnotu odvedené práce vůči zbývající nebo množství neplánované práce, která se objevila v průběhu iterace. Pro zvolené objekty jsou následně stanoveny metriky, které budeme sledovat a vyhodnocovat. Vybrané metriky se často zaznamenávají do tabulky.

Nesmí chybět také definice získávání metrik a způsob jejich správy a uložení. Zdrojem dat může být kromě projektového plánu také zákazník, který například formou dotazníku hodnotí množství a kvalitu práce. Součástí je také definice způsobu, jakým budou data analyzována.

Nad získanými daty jsou pak prováděny analýzy, které mají za cíl poskytnout informace o možných problémech, umožňují sledování v čase a také ukazují na možná zlepšení. Výsledky mohou být vizualizovány například formou grafu nebo vstupem speciálních analytických nástrojů. Výsledky analýzy metrik jsou poté uloženy na předem definované místo pro budoucí použití. Nedílnou součástí tohoto procesu je také seznámení všech zúčastněných stran s výsledky analýzy.

Zajišťování kvality produktu a procesů

Tato oblast poskytuje zaměstnancům a manažerům objektivní vyhodnocení kvality procesů a výsledků práce.

V agilním vývoji mají týmy tendenci se zaměřovat bezprostředně na jednotlivé iterace než přemýšlet v širším kontextu organizace. Je potřeba si zodpovědět následující otázky, aby hodnocení cílů bylo užitečné a efektivní. Jak je prováděno hodnocení cílů? Které procesy a části produktu budou vyhodnocovány? Jak budou výsledky hodnocení integrovány do rytmu týmu (Denní porady, Vzájemné ohodnocení, Retrospektiva Sprintu)?

Před doručením zákazníkovi, během testování nebo integrace je potřeba provést objektivní vyhodnocení vykonávaných procesů na základě popisu procesů, standardů a definovaných postupů a také vybraných produktů práce.

Nedodržené kvalitativní postupy jsou komunikovány se zaměstnanci a manažery, nalezená řešení jsou poté zaznamenávána a spravována pro aktivity zajišťující kvalitu.

Rozhodovací analýza a řešení

Používá se ke stanovení formálních postupů při rozhodování a slouží jako podpora rozhodování. Typickým příkladem je situace, kdy máme několik možností a nevíme, kterou zvolit. Tato procesní oblast stanovuje pravidla pro provedení rozhodovací analýzy a hodnocení zvolených řešení. Na základě hodnocení dochází k výběru nejlepšího řešení, které je nejlepší na základě stanovených měřitelných ukazatelů na počátku.

Stanovením kritérii pro hodnocení dochází k ohodnocení všech možností. Kritériem může být například riziko, obchodní hodnota nebo cena. Dále identifikujeme oblasti zkoumání a zvolíme vhodné vyhodnocovací metody. Používanými metodami jsou často různá modelování a simulace, testování, výzkumy nebo hodnocení produktu zákazníkem. Za použití zvolených metod ohodnotíme zkoumané oblasti a na základě předem definovaných kritérii zvolíme nejlepší řešení.

Analýza příčin a řešení

Úkolem je identifikovat příčiny vybraných nedostatků a přijmout opatření pro zlepšení výkonnosti procesů. Poskytuje organizacím mechanismy pro vyhodnocení jejich procesů a předkládá možnosti zlepšení.

U vybraných nedostatků dochází k analýze příčin. Výsledky mohou pocházet jednak od zákazníka ve formě zpráv o nekvalitních částech, tak také zevnitř organizace, například ze Vzájemných ohodnocení. Často je prováděna Paretova analýza[7] a výstupem je seznam preventivních opatření, které minimalizují další výskyt. Výsledkem může být návrh školení nebo automatizace některého z procesů, změna způsobu predikce odhadu nákladů, ceny nebo času.

Zavedení opatření pro vybrané výsledky je systematický proces, kdy sledujeme efekt zavedeného opatření a zaznamenáme přínos do výsledného dokumentu.

2.3 Stupeň vyspělosti a procesní oblasti

Pro každý Stupeň vyspělosti je modelem CMMI definována sada procesních oblastí, které je nutné v rámci organizace implementovat. Každý vyšší stupeň zahrnuje procesní oblasti

[7] http://www.vlastnicesta.cz/metody-1/pareto-analyza

nižšího stupně a přidává další požadavky. Zároveň je také definován minimální požadovaný Stupeň zralosti procesů pro jednotlivé procesní oblasti v dané Stupni vyspělosti [4]. Podrobný přehled lze nalézt v tabulce 2.3. Sloupec ML označuje Stupeň vyspělosti (z anglického Maturity level), sloupec CL znamená Stupeň zralosti (Capability level).

Oblast	**Zkratka**	**ML**	**CL1**	**CL2**	**CL3**
Konfigurační řízení	CM	2			
Analýza a metriky	MA	2			
Monitorování a kontrola projektu	PMC	2			
Projektové plánování	PP	2			
Zajišťování kvality produktu a procesů	PPQA	2			
Řízení požadavků	REQM	2			
Řízení subdodavatelů	SAM	2			
Rozhodovací analýza a řešení	DAR	3			
Integrované projektové řízení	IPM	3			
Firemní definice procesů	OPD	3			
Firemní procesní zaměření	OPF	3			
Firemní vzdělávání	OT	3			
Integrace produktu	PI	3			
Požadavky na vývoj	RD	3			
Řízení rizik	RSKM	3			
Technická řešení	TS	3			
Validace	VAL	3			
Verifikace	VER	3			
Řízení procesního výkonu v organizaci	OPP	4			
Kvantitativní projektové řízení	QPM	4			
Analýza příčin a řešení	CAR	5			
Řízení výkonu v organizaci	OPM	5			

Tabulka 2.3: Minimální požadovaný Stupeň zralosti procesů pro Stupně vyspělosti.

Vysvětlující příklady

- Pro dosažení Stupně vyspělosti úrovně 2 musí všechny procesní oblasti dosahovat Stupně zralosti procesních oblastí 2 nebo 3.
- Pro dosažení Stupně vyspělosti 3 musí všechny procesní oblasti Stupně vyspělosti 2 nebo 3 dosahovat Stupně zralosti úrovně 3.
- Pro dosažení Stupně vyspělosti 4 musí všechny procesní oblasti stupně 2, 3 i 4 dosahovat Stupně zralosti úrovně 3.

Kapitola 3

Agilní vývojová metodika

Kapitola popisuje vývojovou metodiku aplikovanou ve společnosti Siemens. Jedná se o interní agilní metodiku, která vychází z vývojové metodiky Scrum. Součástí této kapitoly je popis jednotlivých rolí v týmu společně s artefakty, které při vývoji vznikají. Závěrem jsou představeny událost typické pro tuto metodiku. [9]

3.1 Historie

V únoru roku 2001 se konala konference v americkém Utahu, které se zúčastnilo 17 vývojářů. Závěrem tohoto setkání byl publikován manifest, nazvaný „Manifest agilního softwarového vývoje“, který obsahuje dvanáct principů, podle nichž by se měl agilní vývoj softwaru řídit, principy jsou následující. [1]

1. Prioritou je uspokojení zákazníka včasným dodáním kvalitního softwaru a následné kontinuální dodávky.
2. Požadavky na změny jsou vítány, a to dokonce i v pozdějších fázích vývoje. Agilní proces poskytuje zákazníkovi určitou konkurenční výhodu.
3. Časté doručování fungujícího softwaru v intervalech, od několika týdnů do několika měsíců. Ideálně v krátkých intervalech.
4. Manažeři a vývojáři musí denně spolupracovat po celou dobu vývoje projektu.
5. Projekty je dobré seskupovat okolo motivovaných jedinců, poskytovat jim potřebné prostředí, podporu a zároveň jim důvěřovat, že práci zvládnou.
6. Nejefektivnější a nejúčinnější způsob předávání informací v týmu je komunikace tváří v tvář.
7. Fungující software je primárním měřítkem postupu prací.
8. Agilní procesy podporují neustálý rozvoj. Sponzoři, vývojáři a uživatelé by měli díky tomu být schopni udržovat stále konstantní krok.
9. Průběžné sledování technické dokonalosti a dobrý design zajišťuje flexibilitu.
10. Schopnost maximalizovat množství nevykonané práce je klíčová.

11. Nejlepší architektura, požadavky a návrh vznikají v samoorganizovaných týmech.

12. Týmy v pravidelných intervalech samy vyhodnocují svou efektivitu, zamýšlí se nad možnosti zlepšení a přijímají kroky ke zvýšení efektivity.

Část z autorů manifestu následně založila neziskovou organizaci – Agilní alianci, která se zasazuje o vývoj softwaru podle zásad publikovaných v manifestu.

3.2 Tým a týmové role

Základem Scrum týmu jsou tři týmové role – Vlastník produktu (Product Owner), Mistr Scrum (Scrum Master) a Člen vývojového týmu (Development team member), které budou podrobněji popsány níže. Podstatným rysem těchto týmů je skutečnost, že jsou schopny samostatného fungování, výhodou je, že tým si sám volí nejlepší cestu k dosažení cíle, namísto toho aby byl řízen zvenčí. Toto pojetí týmové práce a zapouzdřenost týmu vůči vnějším vlivům má za cíl velkou efektivitu, kreativitu a v neposlední řadě také produktivitu. [13]

3.2.1 Vlastník produktu

Vlastník produktu (dále označovaný jako PO) je zodpovědný za produkt jako celek. Je to právě on, kdo má na starosti správu Produktového katalogu (PB) a jako jediný má právo provádět zásahy do tohoto dokumentu. Řízení a správa PB zahrnuje:

- Přesnou definici jednotlivých položek Produktového katalogu.
- Řazení položek v PB za účelem co nejlepšího dosažení cílů a úkolů.
- Zajišťuje viditelnost PB, který musí být jasný a srozumitelný pro všechny. Rovněž definuje práci týmu pro budoucí období.
- Zajišťuje pochopení všech položek PB všem členům týmu.

V některých případech může PO pověřit výše uvedenými činnostmi členy vývojového týmu, ale primární odpovědnost za správné fungování vždy nese pouze on. Velmi důležité pro práci PO je také to, že členové týmu, potažmo organizace, v něj musí mít plnou důvěru a respektovat jeho rozhodnutí. Všechna jeho rozhodnutí jsou viditelná v PB a členové vývojového týmu nejsou oprávněni pracovat na jiných úkolech než na těch, které jim přidělí právě PO. Toto přidělení práce nemůže určovat nikdo jiný.

3.2.2 Vývojový tým

Vývojový tým (dále v textu označovaný zkratkou DT) se skládá z profesionálů, kteří pracují na doručení požadované funkcionality zákazníkovi na konci daného Sprintu. Pouze tito lidé jsou schopni vyprodukovat Přírůstek, který musí splňovat definici „Hotovo“. DT jsou zmocněny organizací k plné organizaci vlastní práce. Mezi další charakteristiky těchto týmů patří:

- Týmy se řídí samy. Nikdo, dokonce ani Mistr Scrum, neříká členům týmu, jakým způsobem má dojít k přetvoření položek z Produktového katalogu ve výsledný Přírůstek či finální funkcionalitu.

- Všichni členové týmu jsou plnohodnotní a jsou stejně oprávněni pracovat na jakémkoliv úkolu. V zájmu týmu je vytvoření Přírůstku za použití všech dostupných znalostí.
- Scrum nerozlišuje v rámci DT žádné další role ani postavení, všichni vývojáři jsou označováni jako developeři, jsou na stejné úrovni, bez ohledu na množství práce, kterým přispívají.
- Jednotliví členové týmu mohou mít různé specializace a zaměření, nicméně zodpovědnost přísluší vždy týmu jako celku, nikoliv konkrétnímu člověku.
- DT neobsahuje žádné další podtýmy, které by byly vyčleněny na nějakou konkrétní oblast (testování, návrh, dokumentace).

Optimální velikost vývojového týmu není pevně stanovena, Uvádí se, že tým by měl být tak velký, aby byl schopen vytvářet významné Přírůstky na konci každého Sprintu, jako ideální se uvádí počet 3-9 lidí. Méně jak tři členové snižují množství interakce mezi členy týmu a neumožňují příliš velký narůst produktivity. Rovněž může nastat situace, kdy tým o malém počtu členů není schopen doručit uvolnitelný Přírůstek z důvodu chybějících dovedností v týmu. Proto je doporučováno, aby byl tým rovnoměrně zastoupen po znalostní stránce. Naopak více než devět členů není příliš doporučováno, neboť tento počet vyžaduje příliš velké množství koordinace a tým se tak stává méně efektivním. Mezi členy týmu se primárně nepočítají Vlastník produktu ani Mistr Scrum, pokud však pracují na položkách z PB musí být do toho počtu také zahrnuti.

3.2.3 Mistr Scrum

Mistr Scrum (dále v práci označovaný jako SM) je zodpovědný za správné porozumění filosofie Scrum vývoje u všech členů týmu. Má za úkol hlídat, aby všichni členové týmu dodržovali zásady a pravidla metodiky Scrum. Zároveň je mentorem DT. SM je pro členy týmu prostředníkem pro komunikaci s osobami mimo tým a určuje, které interakce jsou užitečné pro tým a které nikoliv. Napomáhá také změnám v těchto interakcích s cílem dosáhnout co nejlepšího výsledku. Dalo by se říci, že Mistr Scrum je podstatě středobodem mezi PO, DT a organizací. Výčet uvedených povinností vůči těmto skupinám lze nalézt níže.

Podpora SM – PO

- Poskytuje efektivní techniky řízení PB.
- Vyjasňuje vize, cíle a význam PBI pro jednotlivé členy týmu (informace získává od PO).
- Učí členy týmu jak vytvářet úplné a jasné položky Produktového katalogu.
- Správné pochopení a praktikování agilního vývoje.
- Podpora Scrum událostí v případě potřeby.

Podpora SM – DT

- Vedení týmu k samostatnému řízení.
- Vedení a učení týmu k vytváření kvalitních produktů.

- Odstraňovaní Překážek zdržujících DT.
- Vedení členů týmu v oblastech prostředí organizace v případech, kde Scrum není ještě zcela implementován.

Podpora SM – Organizace

- Řízení a vedení organizace při nasazování metodiky Scrum.
- Plánování nasazení Scrum implementace uvnitř organizace.
- Pomoc zaměstnancům a zainteresovaným stranám ve správném porozumění Scrum.
- Provádění změn s cílem zvýšit produktivitu Scrum týmu.
- Koordinace s ostatními SM s cílem zvýšení efektivity implementace Scrum uvnitř organizace.

3.3 Základní pojmy

Tato podkapitola osvětluje některé základní pojmy, které jsou pro správné pochopení filosofie Scrum zásadní.

3.3.1 Definice „Hotovo“

Ve chvíli, kdy je některá položka ze Sprint katalogu nebo Přírůstek označen atributem „Hotovo“, je velmi důležité, aby každý článek týmu věděl, co toto znamená. Definici toho, co musí Přírůstek nebo položka ze Sprint katalogu splňovat, aby mohla být takto označena, si definuje samotný tým a může se tedy tým od týmu lišit.

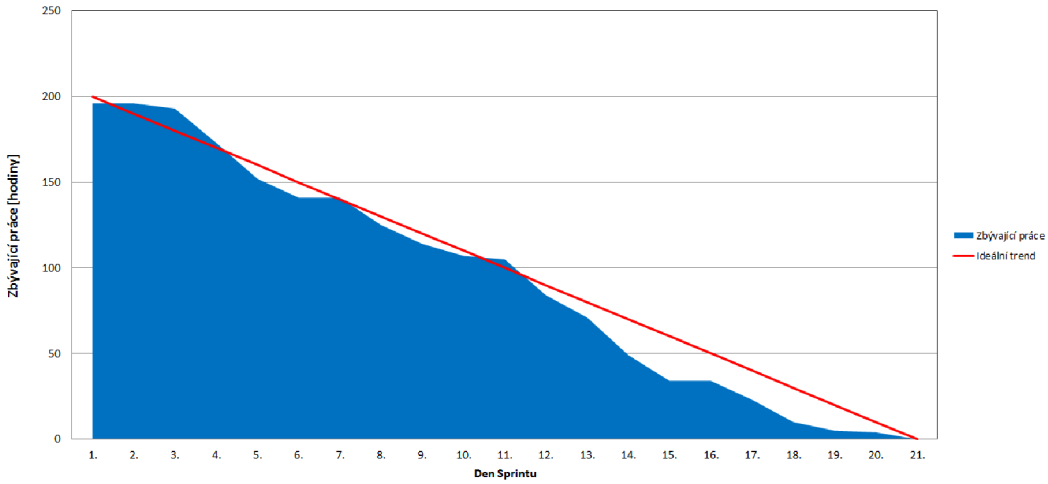
Velmi často je požadováno, aby takto označená položka splňovala požadavky definované uživatelským scénářem, splňovala akceptační kritéria, byla řádně otestována (nejen z hlediska funkcionality, ale také například z pohledu integrace), dokumentována a dokumentace uložena na místě obvyklém, kde bude k dispozici všem členům týmu.

Cílem Scrum je produkovat každý Sprint nové přírustky, proto již během plánování týmy vybírají pouze tolik položek, kolik jsou schopni za jeden Sprint stihnout a jsou si vědomi toho, že všechny vybrané položky musí na konci iterace splňovat jimi používanou definici „Hotovo“. Již při plánování berou v potaz čas nutný například na dokumentaci nebo testování, nejen samotné programování.

Postupem času, jak Vývojové týmy dospívají, se očekává, že jejich definice „Hotovo“ bude dále rozšiřována o přísnější kritéria zajišťující vyšší kvalitu uvolňovaného Přírůstku.

3.3.2 Graf zbývající práce

Jedná se o graf, který vizualizuje průběh celého Sprintu pro jednotlivé dny. Ukazuje vývoj poměru naplánované a zbývající práce v závislosti na aktuálním stavu Sprint katalogu. Graf také obsahuje trend, který ukazuje, zda tým směřuje ke splnění Cíle Sprintu nebo nikoliv. V ideálním případě by měl trend směřovat k poslednímu dni Sprintu k hodnotě 0 zbývajících hodin. Rovněž by neměl tento graf v průběhu Sprintu stoupat, ale jen a pouze klesat k nulové hodnotě.

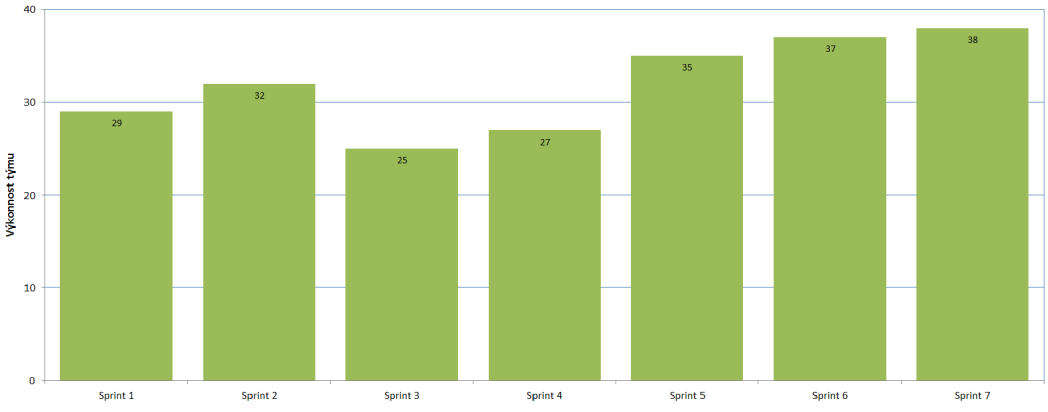


Obrázek 3.1: Vizualiace zbývající práce pro jednotlivé dny Sprintu.

Na uvedeném obrázku 3.1 lze vidět vizualizaci zbývající práce pro jednotlivé dny Sprintu. Z grafu je patrné, že na počátku byl tým lehce mimo plán, ale od 10. dne se mu dařilo dokončovat úkoly nejen podle plánu, ale také dříve, čímž získával rezervu.

3.3.3 Výkonnost týmu

Nezbytnou součástí Scrum a především plánování je ukazatel rychlosti týmu, tzv. Výkonnost týmu (Velocity). Jedná se o číslo, které udává výkonnost týmu pro jeden Sprint. Číslo je součtem všech Ohodnocení uživatelských scénářů u jednotlivých položek, které byl tým schopen doručit v minulé iteraci. Tento ukazatel napomáhá týmům určit objem práce, který jsou schopni vyprodukovat v následujícím Sprintu. Výkonnost týmu se sleduje také napříč Sprinty, aby bylo zjištěno, zda tým stagnuje nebo se zlepšuje. Na počátku by měla hodnota postupně stoupat s tím, jak tým zlepšuje své odhady. Postupem času by se však měla ustálit na konkrétní hodnotě. [14]



Obrázek 3.2: Ukázka možné vizualizace Výkonnosti týmu.

Na grafu 3.2 lze vidět průběh zpřesňování odhadování objemu práce pro požadované Uživatelské scénáře. Je vidět, že tým na počátku měl menší problémy s přesným odhadem a bylo nutné získat zkušenosti v průběhu prvních tří Sprintů. Nicméně od čtvrého Sprintu

již tým správně zpřesnil své odhady a je na první pohled dobře patrné, že jeho Výkonnost stoupá, což je ideální trend. Tým získává postupně stále více zkušeností, je schopen lépe odhadovat objem práce a také se tím sám zdokonaluje.

3.3.4 Uživatelský scénař

Nejčastější typem položek v Produktovém katalogu je „Uživatelský scénář“, která má pevně definovanou strukturu, která bývá dále doplněna dalšími požadavky uživatele. Struktura vypadá takto:

```
Jako <typ uživatele> chci <nějaká akce> tak, že <očekávaný výsledek>.
```

Identifikaci uživatele, jednotlivých akcí a požadovaného výsledku je prováděna týmem a následně dekomponována na jednotlivé Úkoly (Tasky). Pro každý Uživatelský scénář platí, že by měl splňovat požadavky definované anglickou zkratkou INVEST:

- Independent – nezávislost na ostatních položkách
- Negotiable – řešitelnost pro DT
- Valuable – schopnost přinést nějakou hodnotu pro produkt
- Estimable – odhadovatelnost v průběhu plánování
- Small – být dostatečně malý
- Testable requirement – obsahovat požadavky na testování

Jako příklad správně nadefinovaného Uživatelského scénáře lze uvést tuto definici:

```
Jako uživatel chci editovat a uložit IP adresy tak,
že nebudu moct zadat nevalidní hodnoty.
- Mohu zobrazit existující IP adresy.
- Mohu editovat existující IP adresy.
- Mohu přidávat nové a mazat existující IP adresy.
- Po přidání nebo odebrání se zobrazí vždy aktuální seznam.
```

Úkoly

Jednotlivé Uživatelské příběhy vybrané pro konkrétní Sprint se dále rozpadají na menší jednotky práce, ke kterým jsou přiřazování konkrétní členové DT, označované jako Úkoly (Tasks). Úkoly se plánují v hodinách, přičemž platí, že žádný Úkol by neměl svou délkou trvání překročit více jako 12 hodin. Ve většině týmů je navíc velmi často definováno pravidlo, které nedovoluje vytvářet Úkoly větší než 8 hodin, což zaručuje, že je tento úkol splnitelný během jednoho pracovního dne. Příklad Úkolů vytvořených pro výše uvedený Uživatelský scénář včetně odhadnutých hodin by mohl vypadat například takto:

```
Jako uživatel chci editovat a uložit IP adresy. (28 h)
- Vytvoření Uživatelské kontrolky pro zadání a editaci IP adresy (8h)
- Implementace persistenční vrstvy (8h)
- Možnost smazání jedné nebo více položek (4h)
- Zobrazení všech uložených IP adres (4h)
- Validační logika (4h)
```

Dalším významem Úkolů je také to, že z jejich původního odhadu a zbývajícího času se generuje Graf zbývající práce, který byl podrobněji popsán výše v podkapitole 3.3.2.

3.3.5 Ohodnocení scénáře

Položky z PB jsou seřazeny na základě produktových priorit, ale také obsahují tzv. „Ohodnocení scenáře“ (Story Points), což je ukazatel označující složitost dané položky z Produktového katalogu.

K ohodnocování položek z Produktového katalogu dochází během činnosti zvané Plánovací poker (z anglického slova Planning poker). Pro ohodnocení se používá abstraktní bodový systém, který umožňuje ohodnotit obtížnost konkrétní položky bez nutnosti přiřazení definice v hodinách potřebných na vývoj. Nejčastěji používaná stupnice je zaokrouhlená Fibbonaciho posloupnost[1] (1, 2, 3, 5, 8, 13, 20, 40, 100). Některé týmy používají alternativu lineární stupnice (1, 2, 3, 4, 5, 6, 7, 8), mocnin dvou (1, 2, 4, 8, 16, 32, 64, 128) nebo dokonce stupnici používanou pro velikost oblečení (XS, S, M, L, XL).

Odhadování probíhá tak, že všichni členové týmu mají v rukou kartičky a na nich uvedené jednotlivé hodnoty Ohodnocení uživatelských scenářů v závislosti na zvolené stupnici. Hlasují každý za sebe a výsledná složitost položky je dána hodnotou hlasování. Pokud nedospějí členové týmu ke shodě, diskutují svá rozhodnutí. Jedná se například o situace, kdy více členů týmu volí diametrálně rozdílná hodnocení. Po diskusi následuje opětovné hlasování. Hlasování probíhá do té doby, než naleznou shodu.

V případě, že má některý člen týmu pocit, že nemá pro odhad dostatek informací nebo není schopen objektivně rozhodnout, má k dispozici kartičku se symbolem otazníku.

Pokud má některý za členů týmu pocit, že by potřeboval krátkou přestávku, má možnost ukázat kartičku se symbolem kávy a je vyhlášena přestávka.

Více o způsobech odhadování se lze dočíst v knize [5].

3.3.6 Cíl Sprintu

Cíl Sprintu (Sprint Goal) je cíl definovaný v závislosti na zvolených položkách k implementaci a poskytuje členům týmu návod, jakým způsobem transformovat položky z katalogu vybrané pro aktuální Sprint v Přírůstek produktu. Každý člen týmu musí mít tento gól na paměti a postupovat při plnění úkolu v souladu s návodem, který definuje prioritu provádění jednotlivých položek Sprint katalogu v rámci daného Sprintu. Cíl Sprintu může být také v širším kontextu milníkem na časové ose vývoje produktu.

3.4 Artefakty

Scrum artefakty poskytují informaci o práci nebo hodnotě mnoha různými způsoby, jejichž cílem je maximální zajištění transparentnosti a příležitost pro kontrolu aktuálního stavu projektu nebo Sprintu ze strany členů týmu nebo Vlastníka projektu. Artefakty jsou navrženy speciálně tak, aby poskytovaly týmům klíčové informace k úspěšnému doručení Přírůstku, splňujícího definici „Hotovo“.

3.4.1 Produktový katalog

Produktový katalog (z anglického Product backlog, dále jen PB), je seřazený seznam všech požadavků na funcionalitu produktu společně s požadovanými změnami již implementovaných součástí. Za jeho správu je zodpovědný Vlastník produktu.

[1] http://cs.wikipedia.org/wiki/Fibonacciho_posloupnost

Je důležité mít na paměti, že PB není nikdy kompletní. Vyvíjí se společně se s produktem a prostředím, ve kterém se projekt bude používat. Proměna PB je dynamická a jedná se neustálou změnu ve snaze udržet v průběhu vývoje produkt aktuální, konkurenceschopný a užitečný. Tento artefakt existuje tak dlouho, jako je samotný produkt, neboť i již uvolněná verze vyžaduje podporu, která je realizována právě PB.

Jak již bylo řečeno, PB je seznam všech nových vlastností systémů, funkcí, pořadavků, vylepšení a oprav, které mají být realizovány v budoucích verzích. Položky PB, označované jako PBI (Product backlog item), mají vlastní popis, pořadí a odhad pracnosti. Seřazení PBI závisí na aktuálních potřebách vývoje a přání zákazníka. Nejčastěji bývá řazen podle priority. Nejvýše umístěné položky, tedy ty s nejvyšší prioritou, jsou narozdíl od těch, které se vyskytují níže, detailně specifikovány. Má to svůj význam, protože u těch položek s nejvyšší prioritou se předpokládá, že budou implementovány jako první.

Obecně lze také říci, že položky na vrcholu PB mají vysokou granularitu, a je tedy možné je označit definicí „Hotovo“ během jediného Sprintu, a tudíž jsou dostatečně malé a dostatečně specifikované, a mohou být plánovány pro některý z nadcházejících Sprintů.

Při správě PB je důležité mít neustále na paměti, že zákazník většinu svých požadavků bude v průběhu vývoje a používání produktu měnit, bude považovat za více prioritní jiné položky než například považoval před měsícem nebo zjistí, že původně plánované vlastnosti vůbec nepotřebuje. V případě použití Vodopádového modelu by to byl problém, ale agilní vývoj na toto pamatuje, proto je nutné PB neustále aktualizovat a doplňovat.

Na základě aktuální Výkonnosti týmu, ukazatel byl popsán v podkapitole 3.3.3, může Vlastník produktu stanovovat termíny uvolnění produktu nebo jeho součástí k zákazníkovi. Pokud zná aktuální hodnotu výkonnosti, může na základě odhadů Vývojového týmu určit, kolik Sprintů zabere implementace vybraných požadavků.

3.4.2 Sprint katalog

Sprint katalog (Sprint backlog, dále SB) je množina vybraných položek z Produktového katalogu, zpravidla těch, které se vyskytují úplně na vrcholu s nejvyšší prioritou. Součástí tohoto katalogu je také plán pro doručení Přírůstku produktu a realizaci Cíle Sprintu. V podstatě lze říci, že tento artefakt je prognózou pro Vývojový tým týkající se funkcionality, která bude součástí dalšího Přírůstku a práce potřebné k jeho doručení.

Během Plánování Sprintu jsou položky přesunuty ze Produktového katalogu do Sprint katalogu a jednotlivé Uživatelské scénáře rozplánovány do Úkolů tak, jak bylo popsáno v podkapitole 3.3.4. Sprint katalog je tedy závazný dokument, ke kterému se Vývojový tým přihlásí v průběhu plánování, a který je potom pro všechny členy týmu závazný a je nutné se podle něj v průběhu Sprintu řídit.

Stejně jako v případě Produktového katalogu, je potřeba udržovat Sprint katalog rovněž aktuální. Pokud při implementaci nastane situace, která vyžaduje práci, která nebyla původně plánována, je nutné ji okamžitě eskalovat a zanést do SB. Stejně tak, pokud se zjistí, že některé Úkoly nejsou potřeba, z SB se odstraní. V průběhu provádění prací je potřeba aktualizovat stav jednotlivých Úkolů, společně s aktualizací zbývající práce. Aktuální stav plánu Sprintu se sleduje pomocí Grafu zbývající práce (viz. kapitola 3.1) během Denních porad Vývojového týmu. Jedinými osobami, které mohou zasahovat do SB, jsou členové Vývojového týmu.

3.4.3 Přírůstek

Přírůstek (Inkrement) je souhrn všech PBI doručených v aktuálním Sprintu společně se všemi doručenými ve všech předchozích Sprintech. Na konci Sprintu musí Přírůstek jako celek splňovat definici „Hotovo“, díky čemuž je zaručeno, že celý produkt je připraven k uvolnění bez ohledu na to, zda se Vlastník produktu rozhodne aktuální Přírůstek uvolnit k zákazníkovi nebo nikoliv.

3.4.4 Překážka

V pojetí metodiky Scrum je Překážka (Impediment) něco, co brání týmu být produktivní. Překážkou v práci může být například neznalost nové technologie, chybějící detailní specifikace či dokumentace nebo také havárie jakéhokoliv typu (výpadek elektřiny, nefunkční klimatizace, výpadek internetu). Rovněž lze mezi překážky zařadit také nedostupnost Vlastníka produktu.

Pokud chce tým maximalizovat svou efektivitu, je nutné aby role SM byla plně vyčleněna pro řešení vzniklých Překážek. Primární zodpovědnost za odstraňování Překážek je v rukou SM, nikoliv Vlatníka produktu. Překážky v práci by měly byt hlášeny okamžitě, pokud se jedná o překážky které vyžadují neodkladné řešení. Pokud se jedná o překážky, které nemají tak vysokou prioritu, postačuje když jsou hlášeny v rámci Denních porad. Rozhodně by však měly být všechny vzniklé Překážky zaznamenávány a sledovány. [15]

3.5 Události

Scrum ve své definici používá časově ohraničené události, z nichž každá má stanovenu maximální délku trvání. Díky tomu je možné již při plánování relativně přesně vyčlenit určité časové kvantum pro tyto události. Mezi hlavní události patří:

- Sprint
- Plánování Sprintu (Sprint planning)
- Revize Sprintu (Sprint review)
- Retrospektiva Sprintu (Sprint retrospective)
- Revize kódu (Code review)
- Aktualizace Produktového katalogu (Backlog grooming)

Všechny události jsou navrženy s ohledem na jejich jednoduchost a velmi dobrou možnost kontroly. Obálkou všech událostí je pak Sprint, který bude definován níže. [7]

3.5.1 Sprint

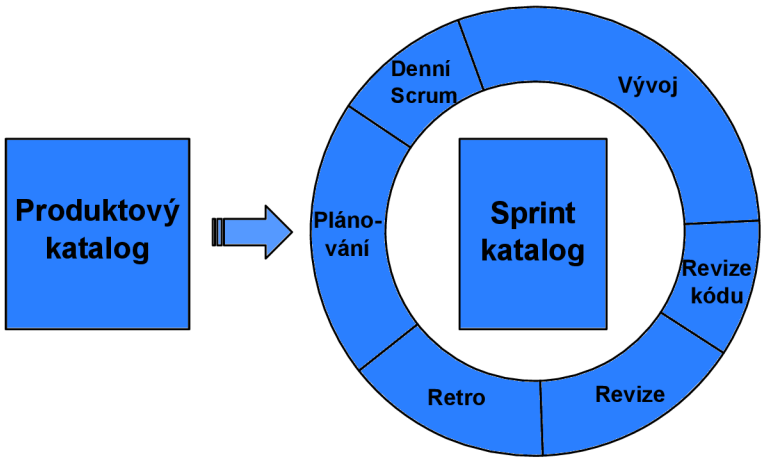
Základem agilního vývoje je Sprint neboli jedna iterace, což je časově ohraničená událost, která má pěvně stanovený začátek a konec. Výsledkem Sprintu je Přírůstek, který je uvolnitelný k zákazníkovi a splňuje definici „Hotovo“.

Každý Sprint může být považován za malý projekt, který obsahuje všechny části vývoje softwaru od plánování, přes samotný vývoj až po doručení výsledku. Nezbytnou součástí je také testování a dokumentace.

Doba trvání jednoho Sprintu je silně individuální v závislosti na týmu, literatura však uvádí, že minimálně by měl trvat 2 týdny, naopak by Sprint neměl překročit dobu jednoho měsíce. U velmi krátkých Sprintů totiž nastává jev, kdy režie spojená s plánováním přesahuje 30% času, který je k dispozici na vývoj, a tudíž administrativa zabírá velké množství času, který by bylo možné investovat do vývoje. U velmi dlouhých iterací zase naopak hrozí riziko situace, kdy dojde ke změně priorit na produktu, změnám či zvýšení rizika.

V průběhu každého Sprintu se objevují v uvedeném pořadí následující události, které budou popsány dále podrobněji:

- Plánování Sprintu
- Denní Scrum
- Revize Sprintu
- Retrospektiva Sprintu



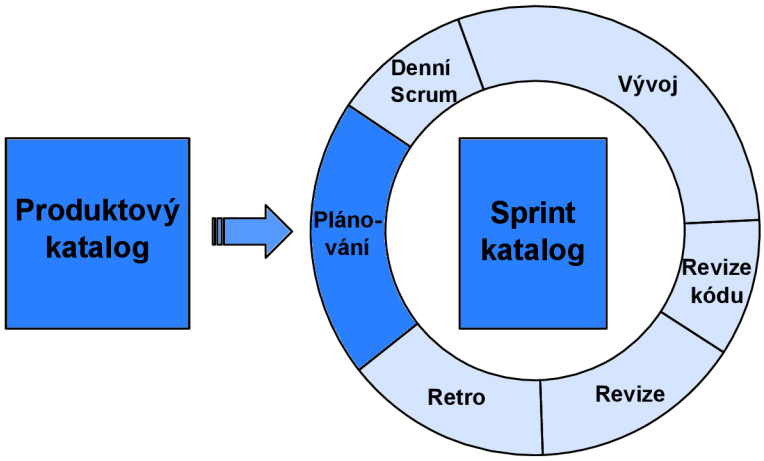
Obrázek 3.3: Vizualizace událostí Sprintu.

3.5.2 Plánování Sprintu

Objem práce, která má být odvedena v daném Sprintu je plánována za účasti a spolupráce celého týmu a odehrává se na Plánovacím setkání. Délka trvání tohoto plánování je stanovena na 8 hodin pro Sprinty o délce trvání jednoho měsíce. Pro kratší iterace je délka přímo úměrná jeho trvání, například dvoutýdenní Sprint bude mít toto setkání dlouhé 4 hodiny.

Plánování se skládá ze dvou částí, z nichž by každá měla zabírat přibližně polovinu délky trvání. Každá z částí přináší odpověď na jednu z těchto klíčových otázek:

1. Co má být doručeno jako Přírůstek následujícího Sprintu?
2. Jakou práci bude nutné vynaložit k doručení Přírůstku?



Obrázek 3.4: Plánování Sprintu.

Co má být doručeno jako Přírůstek následujícího Sprintu?

V této části se provádí předpověď toho, na čem bude tým pracovat v následujícím Sprintu. PO prezentuje členům týmu seřazený PB a vysvětluje význam jednotlivých položek členům týmu tak, aby měl každý představu o významu jednotlivých položek. Vstupem této části plánování je jednak již zmiňovaný PB, ale také Přírůstek produktu vyprodukovaný v předcházejícím Sprintu a ukazatel Výkonnosti týmu, popsaný v podkapitole 3.3.3.

Tým podle aktuální Výkonnosti vybere z vrcholu Produktového katalogu tolik položek, kolik je aktuálně schopen realizovat. V případě, že některé vybrané položky ještě nebyly odhadnuty, provede tým odhad podle pravidel Plánovacího pokeru, který byl popsán v podkapitole 3.3.5. Tato situace může nastat v důsledku náhlé změny priorit ze strany zákazníka.

Pouze Vývojový tým a nikdo jiný, určuje kolik práce je schopen stihnout v daném Sprintu a rozhoduje o tom k jaké části práce, kterou navrhne Vlastník produktu, se zaváže. Tým může například na základě své aktuální Výkonnosti některé PBI odmítnout nebo naopak Vlastník produktu může také vybrat PBI navíc.

Při plánování je také nutné vzít v potaz dostupnost jednotlivých členů týmu v plánovaném Sprintu. Všechny tyto skutečnosti mohou výsledek nepříznivě ovlivnit, a proto je nutné mít na paměti různé dovolené nebo školení, jichž se členové týmu mohou účastnit.

Výsledkem této části je seznam přijatých PBI a definice Cíle Sprintu (viz. podkapitola 3.3.6).

Jakou práci bude nutné vynaložit k doručení Přírůstku?

V rámci této fáze plánování tým rozhoduje jakým způsobem požadované položky z PB přetvoří v Přírůstek splňující definici „Hotovo“ v průběhu nadcházejícího Sprintu. Všechny položky vybrané v první fázi plánování z PB jsou zařazeny do Sprint katalogu. Součástí katalogu je také plán na jejich doručení nebo integraci do produktu.

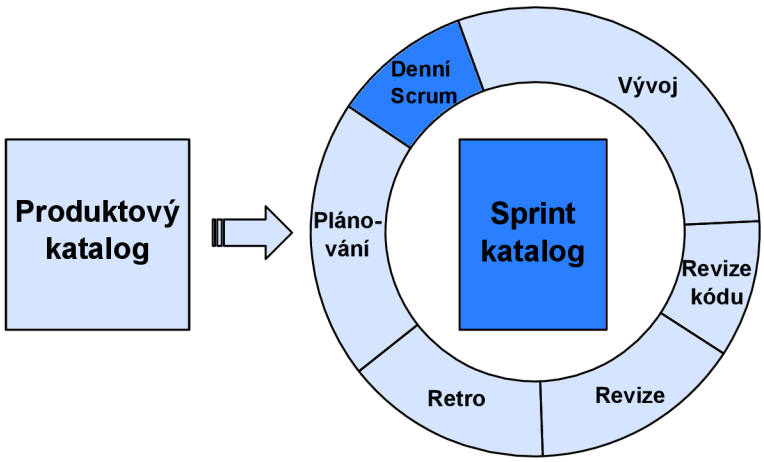
PBI se v této fázi rozpadají na menší jednotky práce. PBI je zpravidla definována jako Uživatelský scénář která se během této fáze plánování rozpadá na další Úkoly tak, jak bylo popsáno v podkapitole 3.3.4.

PO v průběhu této fáze plánování poskytuje podporu při vyjasňování obsahu konkrétní PBI tak, aby byl tým schopen přesně odhadnout objem práce nutný k jejímu splnění. Rovněž může přidat novou PBI z PB nebo naopak některou do něj vrátit zpět, pokud se ukáže, že

je nesplnitelná v daném Sprintu. V závěru plánování by měl být každý člen týmu schopen vysvětlit PO a SM jakým způsobem budou schopni splnit daný Sprint a především jak jsou schopni splnit zvolený Cíl Sprintu a vytvořit odpovídající Přírůstek.

3.5.3 Denní Scrum

Jedná o každodenní událost, která by měla být prováděna vždy ve stejný čas a ideálně také na stejném místě. Délka trvání nesmí přesáhnout 15 minut a smyslem této události je získat přehled o aktivitách ostatních členů týmu a vytvořit plán práce na dalších 24 hodin. Součástí je také kontrola odvedené práce od posledního Denního Scrum.



Obrázek 3.5: Denní Scrum.

Během tohoto setkání každý člen DT vysvětluje ostatním své odpovědi na tyto otázky:

- Co jsem udělal od posledního Denního Scrum?
- Co budu dělat do dalšího Denního Scrum?
- Jaké problémy aktuálně řeším a co mám za překážky v práci?

Všichni členové týmu musí být schopni vysvětlit Vlastníkovi produktu a Mistru Scrum na čem zrovna tým pracují a jak to přispívá k naplnění Cíle Sprintu. Tato setkání napomáhají členům týmu znát odpovědi na tyto otázky.

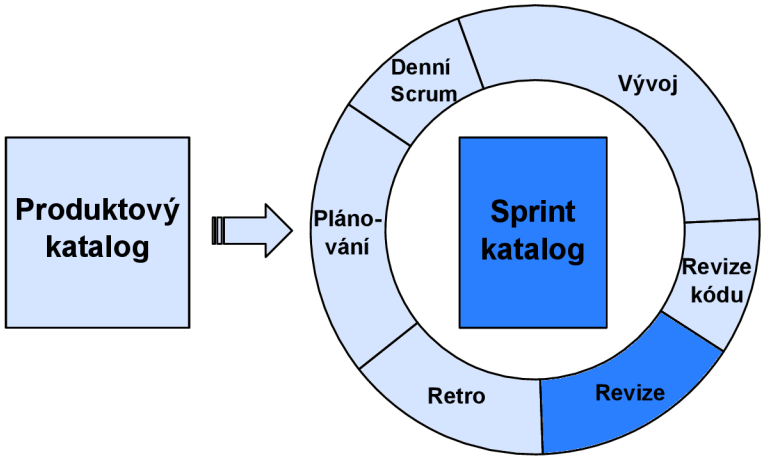
Součástí Denního setkání je zhodnocení aktuálního posunu v rámci Sprintu. Během této aktivity nebo bezprostředně před ní by mělo docházet k aktualizaci SB na základě aktuální stavu vývoje. Pro zhodnocení situace se velmi často používá již dříve popsaný Graf zbývající práce.

Úkolem SM je zajistit, aby docházelo k pravidelnému konání Denního Scrum a aby se jej účastnili výhradně členové DT. Rovněž se snaží hlídat délku trvání, aby nepřesáhla zmiňovaných 15 minut. Členové DT mají za úkol pravidelně provádět Denní Scrum.

Je velmi důležité si uvědomit, že Denní setkání není pouze setkáním s informací o stavu, ale krátkým plánováním a ujasněním úkolů tak, aby tým byl schopen přetavit položky Sprint katalogu ve výsledný Přírůstek na konci iterace. Denní setkání umožňují zvýšit komunikaci v týmu, eliminovat ostatní porady, identifikovat a řešit Překážky. Důraz je také kladen na podporu rychlého rozhodování a zvyšuje se povědomí členů týmu o projektu jako takovém.

3.5.4 Revize Sprintu

Revize Sprintu je prováděna na konci daného Sprintu za účelem kontroly vytvořené Přírůstku DT a případného upravení PB. Smyslem události je informování o tom, co bylo dokončeno tento Sprint za účasti všech Členů vývojového týmu, Vlastníka produktu a všech zainteresovaných stran. Součástí je samotná prezentace Přírůstku a získání odezvy od všech zúčastněných.



Obrázek 3.6: Revize Sprintu.

Délka trvání je stanovena na 4 hodiny pro týmy, které praktikují Sprinty o délce jednoho měsíce. Pokud se jedná o tým, který má kratší Sprinty je doba trvání přímo úměrná délce iterace. Typický průběh revize Sprintu je následující:

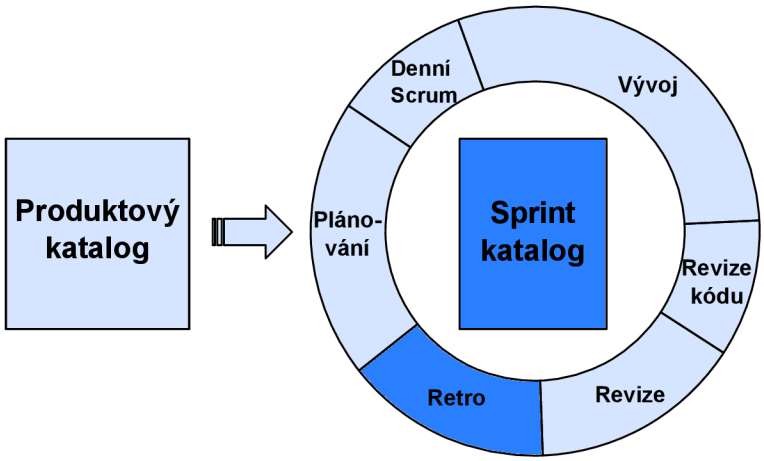
- PO identifikuje, co bylo dokončeno v tomto Sprintu a co se naopak nepodařilo dokončit. Jinými slovy zkoumá, zda všech položky SB odpovídají definici „Hotovo“.
- DT diskutuje, co se podařilo, jaké problémy se vyskytly a jak byly vyřešeny.
- DT představuje výsledek své práce, jsou ukázány všechny položky SB, které splňují definici „Hotovo“ a odpovídá na otázky ohledně Přírůstku.
- PO aktualizuje PB na základě výsledku tohoto Sprintu, rovněž aktualizuje pravděpodobné datum dokončení produktu podle počtu zbývajících položek a aktuální Výkonnosti týmu.
- Celá skupina diskutuje o možném dalším vývoji, což poskytuje cenné informace, které jsou vstupem pro další plánování.

Výsledkem této události je aktualizovaný PB, který již neobsahuje dokončené položky, ale může obsahovat nedokončené položky z aktuálního Sprintu a je základem pro další plánování.

3.5.5 Retrospektiva Sprintu

Po skončení Revize Sprintu a před plánováním nového Sprintu přichází na řadu Retrospektiva. Pro Vývojový tým je to příležitost, jak analyzovat činnost týmu a vytvořit plán pro zlepšení, která budou realizována v rámci nadcházející iterace. Součástí je ohlédnutí za skončeným Sprintem z pohledu týmu a kontrola provedení zlepšení, definovaných v předcházející

Retrospektivě. Retrospektiva by neměla přesáhnout více jak tři hodiny, tato maximální délka trvání se týká týmů s měsíčními Sprinty, pro kratší iterace je délka kratší.



Obrázek 3.7: Retrospektiva Sprintu.

Smyslem Retrospektivy je provést tyto body:

- Zhodnotit skončený Sprint s ohledem na lidi, vztahy v týmu, procesy a nástroje.
- Identifikovat a vyzdvihnout procesy a činnosti, které se podařily a které se naopak nepovedly a diskutovat potenciální zlepšení.
- Vytvořit plán pro zavedení zlepšení při Vývojového týmu.

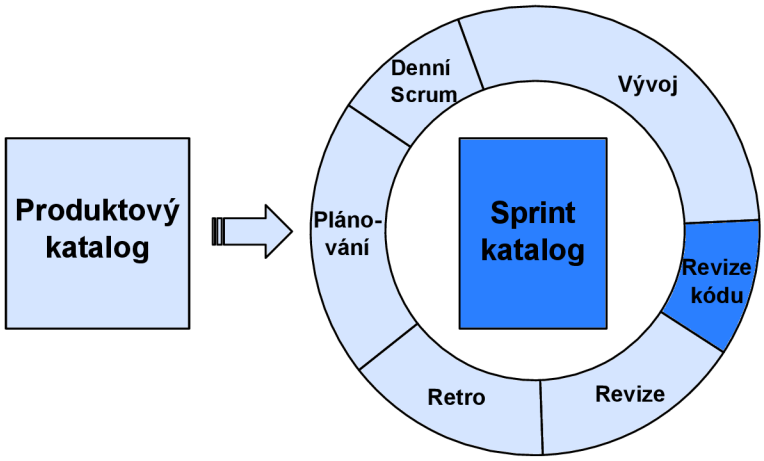
SM podporuje členy týmu v jejich snaze o zlepšování tak, aby se tým stával efektivnějším, což lze například v dlouhodobém měřítku sledovat například podle Výkonnosti týmu. Nalezené nedostatky a způsob jejich odstranění je realizován v dalším Sprintu, ale nemusí tomu tak být vždy. Prostor pro zlepšení se nabízí kdykoliv během iterace a to tak, že může být jednak rovnou implementován nebo zapsán pro příští Retrospektivu. Retrospektiva Sprintu je pouze vhodnou příležitostí.

3.5.6 Revize kódu

Revize kódu je další událostí definovanou metodikou Scrum a jedná se o jeden z nejmocnější nástrojů pro sledování kvality softwaru. Jde o setkání postavené na bázi Vzájemného ohodnocení. Vzájemné ohodnocení (Peer review) je činnost, při které je implementovaný kód zkoumaný jinými osobami než autorem kódu. Této události se může účastnit jeden nebo více členů týmu a měla by být periodicky plánována a systematicky prováděna. Smyslem je odhalit chyby dříve, než budou reportovány ze strany zákazníků. Je statisticky dokázáno, že odstranění chyby je tím dražší, čím později je odhalena, v ideálním případě je nejlepší chybu odhalit již ve fázi vývoje před uvolněním k zákazníkovi, a proto Scrum definuje tuto událost.

Statistika úvádí, že testování softwaru má omezenou účinnost. Tabulka 3.1 uvádí srovnání úspěšnosti nalezení chyby pro jednotlivé typy testů a také Vzájemné ohodnocení. Z tohoto srovnání je patrné, že Vzájemné ohodnocení dokáže odhalit statisticky zdaleka největší množství chyb, a proto je potřeba této činnosti věnovat patřičnou pozornost. [16]

Na závěr bych rád zmínil několik reálných příkladů z knihy Steva McConella, které ukazují, kolik prostředků může správně prováděná Revize kódu ušetřit: [10]



Obrázek 3.8: Revize kódu.

Druh	Úspěšnost odhalení
Testy jednotek	25 %
Funkční testy	35 %
Integrační testy	45 %
Vzájemné ohodnocení	55-60 %

Tabulka 3.1: Úspěšnost odhalení chyby.

- Ve skupině 11 programů vyvíjených stejnou skupinou lidí bylo prvních pět vyvíjeno bez provádění Revize kódu, zatímco zbývajících šest za jeho použití. Po uvolnění všech produktů k zákazníkovi vykazovala první skupina programů průměrnou chybovost 4,5 chyby na 100 řádků kódu, zatímco druhá pouze 0,82 chyb na 100 řádků kódu. Provádění Revize kódu snížilo v tomto případě chybovost o 80pdf%.
- Společnosti AT&T s více než 200 zaměstnanci se díky zavedení Revize kódu do procesu vývoje podařilo podle publikované studie navýšit svou produktivitu o 14 % a snížit chybovost svých produktů reportovanou zákazníky o 90 %.
- Laboratoře Jet Propulsion odhadují, že provádění Revize kódu jim ročně uspoří prostředky ve výši 25 000 $.

3.5.7 Aktualizace Produktového katalogu

Aktualizace Produktového katalogu (Backlog grooming) je událost, kdy dochází za účasti celého týmu k aktualizaci stávajících PB na základě nových skutečností nebo požadavků zákazníka. V rámci této události dochází k detailní specifikaci jednotlivých položek, odhadování pracnosti a řazení položek tak, aby reflektovaly aktuální stav.

Tato událost je realizována v průběhu Sprintu za vzájemné spolupráce Vlastníka produktu a Vývojového týmu. Je velmi důležité, aby čas vyhrazený pro tuto aktivitu nezabíral více než 10 % kapacity vývojového týmu. Může se například jednat setkání jednou týdně, které nezabere více než jednu hodinu.

Klíčovou aktivitou Aktualizace katalogu je odhadování pracnosti jednotlivých PBI. Tento odhad probíhá na základě Planovácího pokeru, který byl podrobněji popsán v podkapitole

3.3.5. Je nutné zdůraznit, že zodpovědnost za správný odhad nese právě Vývojový tým. Vlastník produktu může poskytnout dodatečné informace nebo nabídnout možnosti kompromisu, nicméně za finální odhad jsou zodpovědné ty osoby, které požadavky definované v PBI budou implementovat.

Kapitola 4

Analýza stavu společnosti

Následující kapitola popisuje zkoumanou společnost a přináší závěry provedené analýzy. Rovněž lze zde najít možné oblasti zlepšení tak, aby zavedení těchto vylepšení napomohlo společnosti Siemens k získání požadované úrovně kvality podle modelu CMMI.

4.1 Popis projektu

Jako zkoumaný projekt pro vyhodnocení úrovně Vyspělosti podle modelu CMMI byl za cíl diplomové práce vybrán projekt Sitraffic Office uvnitř společnosti Siemens. Jedná se o projekt z oblasti dopravních systémů, který se zabývá vývojem softwaru pro inteligentní řízení křižovatek. Vývojový tým se aktuálně skládá ze 6 vývojářů, kteří sídlí v Brně. Samotný projekt již běží více než deset let.

Vývoj je řízem interní agilní metodikou společnosti Siemens, která byla představena a podrobně popsána v předchozí kapitole. Jak již bylo dříve rečeno, metodika je postavena nad metodikou Scrum. Zákazníkem pro tento projekt je oddělení uvnitř společnosti Siemens, které sídlí v Německu. Toto oddělení definuje požadavky na vývoj a řídí veškeré směřování produktu. Brněnský tým vývojářů se podílí na vývoji produktu a údržbě stávající verze aplikace.

Organizace již v minulosti získala certifikaci řízení kvality ISO 9001 a nyní usiluje o splnění požadavků modelu kvality CMMI. Cílem této diplomové práce projektu je analýza současného stavu procesů v rámci projektu, konfrontace s požadavky CMMI modelu pro požadovanou úroveň a identifikace slabých míst, která bude potřeba dále rozvíjet tak, aby projekt dosáhl požadované úrovně při budoucím auditu.

4.2 Nerelevantní oblasti

Při konfrontaci požadavků modelu CMMI a aktuálního stavu projektu bylo zjištěno, že některé procesní oblasti nejsou pro zkoumaný projekt relevantní. Jedná se v prvé řadě o procesní oblasti definované modelem pro vyšší úroveň, o níž organizace v současné době neusiluje. Jedná se o následující procesní oblasti:

- Řízení procesního výkonu v organizaci
- Kvantitativní projektové řízení
- Analýza příčin a řešení

- Řízení výkonu v organizaci

Model také definuje procesní oblasti, které sice rozsahem požadavků spadají do úrovně, o kterou společnost usiluje, ale pro zkoumaný projekt nejsou rovněž zajímavé, neboť se o jejich provádění stará kompletně zákazník v Německu, a proto nebudou do zkoumání zahrnuty.

- Řízení subdodavatelů
- Řízení požadavků
- Integrace produktu a testování
- Hardwarová implementace a testování

V kontextu zkoumaného projektu je tím, kdo definuje požadavky na vývoj právě zákazník, tým u tohoto projektu pouze implementuje požadované součásti a v rámci svých možností provádí testování nově implementovaných součástí. Vývojový tým v Brně rovněž přichází s podněty pro implementaci, ale tato činnost není jeho primárním úkolem. O komplexnější testování, jakým jsou například integrační testy, se stará testovací tým na straně zákazníka a zpět pouze reportuje nalezené nedostatky.

Uvedeným oblastem nebude nadále věnována pozornost.

4.3 Analýza projektu

Během analýzy současného stavu ve zkoumané společnosti jsem se zaměřil na jednotlivé procesní kategorie modelu CMMI. Pro každou kategorii jsem podrobně rozebral její procesní oblasti a pokusil se na základě dostupných informací o zhodnocení stávajícího situace. Součástí bylo také nalezení možných oblastí pro zlepšení tak, aby se společnost jejich implementací přiblížila k požadované úrovni Vyspělosti podle modelu CMMI. U procesních oblastí, které nejsou vůbec zmíněny jsem nenašel žádné nedostatky, a proto v souhrnu nejsou uváděny.

Model kvality CMMI definuje pro softwarový vývoj celkem 22 procesních oblastí, z tohoto počtu bylo pro zkoumání vyřazeno 8 procesních oblastí z důvodu, které byly uvedeny dříve. Předmětem zkoumání tedy bylo 14 procesních oblastí. Celkem bylo identifikováno 6 procesních oblastí, které je nutné dále rozvíjet tak, aby splňovaly požadavky modelu. Podrobný popis nalezených zjištění je uveden v následující podkapitole. Sumarizace počtu zkoumaných procesních oblastí se nachází v tabulce 4.1

	Počet procesních oblastí
Model CMMI pro SW	22
Vyřazeno	8
Analyzováno	14
Nutno rozvíjet	**6**

Tabulka 4.1: Počet zkoumaných a vyřazených procesních oblastí.

4.3.1 Projektové řízení

Projektové řízení

Jednotlivé role v rámci projektového řízení, stejně jako role jednotlivých členů týmu, jsou ve všech případech kompletně přiřazeny, všechny role zastávají výhradně kvalifikované osoby, které disponují požadovanými znalostmi a zkušenostmi. Osoby zastávající tyto role absolvují pravidelně potřebná školení a jsou rovněž držiteli relevantních certifikátů. Kapacita pro tyto role je správně plánována a sledována. Všechny parametry projektu jsou správně definovány v dokumentu nazývaném jako „Příručka projektu“ (Project Handbook).

Správa a zodpovědnost za PB je v rukou zákazníka. Jedná se především o činnosti spojené s určením priorit, definicí požadavků na nové funkce produktu, ale také na vyjasnění akceptačních kritérií.

Plánování je realizováno agilním způsobem, jednotlivé požadavky zákazníka jsou uvnitř SB děleny na úkoly a řádně odhadovány za účasti členů týmu. Metriky jsou definovány a používány, Graf zbývající práce je používán jako vstup pro Retrospektivu.

Čtvrtletní setkání se zákazníkem jsou dodržována a mají za cíl plánování zdrojů. Zapojení všech zúčastněných stran na projektu je pravidelné a významné. Rovněž je aplikováno řízení rizik.

Možná zlepšení

- **Metody odhadování** – Odhady zákazníka nejsou viditelné na úrovni projektu. Do budoucna by měla být více sledována historická data.
- **Analýza a metriky na úrovni projektu** – Seznam sledovaných metrik společně s jejich atributy je správně uveden v Plánu kvality. Seznam není však úplný, a proto by bylo možné jej dále rozšířit.
- **Řízení a sledování projektu** – Aktuální způsob podávání zpráv by mohl být rozšířen o tabulku s relevantními položkami (umožní vyšší přehlednost). Nedostatečná kontrola zbývajícího objemu práce (námět pro Denní porady). Data z předchozích projektů nejsou používána jako historická data, ale berou se v úvahu zkušenosti z předchozích Sprintů.
- **Řízení rizik** – Je prováděno podle vytyčeného plánu, občas se však zapomíná na jeho dokumentování.
- **Dostupnost členů týmu** – Dostupnost jednotlivých členů týmu je zaznamenávána v průběhu plánování. Způsob zadávání dostupnosti na další Sprint by bylo možné zjednodušit a více zautomatizovat.

Zajištění kvality a Vzájemné ohodnocení

Role Manažera kvality na projektu existuje, jeho dostupnost je plánována a sledována. Jeho činnost zabírá část Revize projektu, Revize Sprintu a také komunikace se zákazníkem. Projektové procesy, procedury a použité postupy jsou pravidelně kontrolovány na Retrospektivě Sprintu.

Na projektu je definován koncept Revize kódu, stejně jako kritéria výběru kódu, navíc je také používána technika párového programování jako způsob revize kódu a zdokonalování členů týmu. Pro revizi plnění plánu je používán revidovací nástroj. Posloupnost Revize

kódu a Plánování je definována. Definice hotové části a akceptační kritéria jsou používána pro kontrolu kompletnosti a akceptovatelnosti výsledků Sprintu.

Možná zlepšení

- **Dodržování procesů** – Změny všech procesů, metod a procedur by měly být zaznamenávány a komunikovány. Občas se vyskytuje bariéra v komunikaci se zákazníkem týkající se připravovaných termínů uvolnění produktu nebo předávání relevantních informací týmu.
- **Zprávy, eskalace a řešení jakosti** – Manažer kvality pro danou oblast by měl být v distribučním seznamu zpráv jakosti pro daný projekt.
- **Vzájemné ohodnocení a revize rozsahu** – Rozhodnutí o revizi a kritéria výběru položek pro revizi musí být jasná. Technické dokumentace jsou považovány za dokumentaci a nejsou revidovány. Kritéria výběru kódu u nových členů týmu by měla být doplněna. Vyhodnocování záznamů z revize by se mělo provádět systematicky.

4.3.2 Procesní řízení

Definice a správa procesů

Role manažera kvality je definována na úrovni celého oddělení, je přiřazena konkrétní osobě a tato osoba je viditelná v organizační struktuře. Všechna potřebná a dostupná školení pro tuto roli jsou prováděna. Firemní procesy jsou spravovány na úrovni společnosti a jsou součástí Příručky procesů. Procesy jsou správně a srozumitelně verzovány a uloženy na firemním serveru, dokumenty specifické pro oddělení pak na Sharepoint.

Pro hlavní firemní procesy jsou uvedeny také varianty popisující celý cyklus vývoje softwaru, jako je tomu například pro agilní vývoj. Na úrovni projektu jsou manažeři procesů, vlastníci a zainteresované strany vždy přizváni ke koordinaci příslušných změn. Existuje aplikace pro hodnocení procesů odpovědnými osobami, výsledky hodnocení jsou pak sledovány na poradě vedení společnosti každý měsíc. Je vyčleněn rozpočet pro zlepšování úrovně kvality, je rovněž vyčleněna celá kapacita manažera kvality pro zlepšování. Možnosti zlepšování procesů jsou kontrolovány audity (ISO, CMMI), přezkoumáním vedením, impuls může také přicházet z hodnotící aplikace.

Ohlasy jsou analyzovány a mohou být předmětem zlepšování. Výsledky výše uvedených kontrol jsou předmětem analýzy při schůzkách na různých úrovních společnosti. V rámci projektu byly kontrolovány CMMI politiky, existuji pravidelné zprávy kvality, stejně jako zprávy o kvalitativních problémech.

Možná zlepšení

- **Institucionalizace definice a správy procesů** – Zlepšení v oblasti kvality jsou definována a částečně sledována. Měl by být zaveden systematický přístup pro odhadování a sledování kvalitativních zlepšení.
- **Plánování a realizace zlepšení** – Hlášení o stavu větších zlepšení procesů by mohlo být dokonalejší (například by se mohlo stát součástí setkání vedení společnosti).

4.3.3 Vývoj

Definice funkcionality a architektury

Na straně zákazníka je definována zodpovědnost za role Vlastníka produktu a Architekta produktu, pozice Softwarového architekta je definována na úrovni projektu. Definice a analýza jednotlivých požadavků je realizována na straně zákazníka. Tato analýza zahrnuje také bezpečnostní a hardwarové požadavky.

Při plánování dochází k explicitnímu přiřazení jednotlivých implementovanách vlastností produktu pro daný Sprint a určení priorit. Přiřazení provádí opět zákazník. Inovace a patenty jsou součástí cílů organizace a jsou sdělovány, lidé jsou školeni a motivováni. Tyto aktivity jsou zahrnuty do plánování.

Možná zlepšení

- **Pochopení a sledovatelnost požadavků a funkcí** – Sledovatelnost testovacích případů pro testování jednotek je k dispozici v systému, nicméně je velmi těžké tyto informace jednoduše získat.
- **Právo duševního vlastnictví** – Komunikace týkající se inovací a patentů v týmu by se měla zlepšit.

Softwarová implementace a testování

Role členů týmu jsou definovány používanou agilní metodikou a jejich přiřazení je uvedeno v Příručce projektu. Jejich práce je plánována ve SB (implementace, testování, . . .). Pro projekt existuje odpovídající Testovací plán, stejně jako plán Konfiguračního řízení. Tým navrhuje zlepšení softwarové architektury zákazníkovi, rozhodnutí o provedení je vždy na straně zákazníka. Pro statickou analýzu kódu jsou používány odpovídající nástroje (ReSharper[1], Sonar[2]).

V souvislosti s touto oblastí budeme hovořit o tzv. Testování jednotek (Unit testing), které spočívá v ověřování správné funkčnosti dílčích částí produktu neboli jednotek zdrojového kódu.

Možná zlepšení

- **Institucionalizace softwarové implementace a testování** – Zodpovědnost a dostupnost jednotlivých kapacit by měla být jasnější.
- **Softwarová architektura** – Technická dokumentace je udržována aktuální v případě funkčnosti použité v projektu. Část zbývající technické dokumentace by měla být také spravována. Definice rozhraní je součástí vestavěné dokumentace. Pro lepší vizualizaci by měly být používány UML nástroje (možnost automatického generování).
- **Softwarový design a psaní kódu** – Jazykově specifické konvence jsou součástí vývojového nástroje, měly by být rovněž uvedeny v projektové dokumentaci.
- **Softwarové testování jednotek a integrační testování** – Testovací strategie je definována, ale měla by být dopracována. Mělo by být vysvětleno, že je specifická pro

[1] http://www.jetbrains.com/resharper/
[2] http://www.sonarsource.org/

agilní vývoj, především by měla být zmíněna orientace na časový úsek ohraničený dobou trvání jednoho Sprintu. Význam testování jednotek a testování součástí by měl být zřejmý v závislosti na agilní vývoji. Omezení zodpovědností pro jednotlivé stupně testování by měly být jasně uvedeny. Kontinuální integrace by měla být popsána explicitně. Výsledky testování a integrace by měly být dokumentovány trvale.

- **Softwarové testování jednotek a integrační testování** – Vzhledem k agilnímu vývoji by mělo být zváženo použití regresních testů.

4.3.4 Podpora

Konfigurační řízení

Kompletní zodpovědnost za konfigurační řízení na tomto projektu je na straně zákazníka. Na úrovni projektu je určena kontaktní osoba a existuje Plán konfiguračního řízení. Aktivity spojené s konfiguračním řízením jsou součástí SB, jsou plánovány a sledovány, dále jsou také součástí Zprávy kvality. Všechny položky jsou předmětem konfiguračních pravidel (zdrojové kódy, dokumentace na wikipedii) a jsou popsány v Plánu konfiguračního řízení.

Plánování uvolnění další verze (Release) je v kompetenci zákazníka, informace o chystaných uvolnění jsou vždy k dispozici všem členům týmu. Popis postupu pro změnové a chybové řízení je k dispozici v Plánu konfiguračního řízení, celková zodpovědnost je opět na straně zákazníka. Chybový stav a změny jsou reportovány na poradě dvakrát týdně, Revizi Sprintu a jsou také viditelné ve SB.

Možná zlepšení

- **Konfigurační položky a systém řízení konfigurací** – Mělo by být přidáno pravidlo, že tyto položky musí být každý den kontrolovány. Jedná se především o výsledky testů, které jsou spouštěny pravidělně s každým sestavením.
- **Podávání zpráv konfiguračního řízení** – Zvláštní zprávy konfiguračního řízení by měly být součástí Denních porad a Retrospektivy Sprintu.

4.4 Závěr analýzy

Z provedené analýzy vyplynulo několik potencionálních oblastí pro zlepšení, které by mohly významně napomoci k získání požadovaného Stupně vyspělosti. Hlavní zlepšení lze rozdělit do dvou kategorií, a to na ta týkající se samotného projektu a pak také na ta zlepšení, která by bylo možné realizovat v rámci organizace jako celku.

Na projektové úrovni bylo zjištěno, že velký potenciál pro zlepšení poskytuje oblast testování, u níž by bylo vhodné doplnit testovací strategii tak, aby více reflektovala specifika agilního vývoje. Rovněž by bylo dobré zvážit použití regresních testů.

Mezi hlavní zlepšení, která lze aplikovat na úrovni orgranizace, lze zařadit sběr a dostupnost historických dat, která by měla být k dispozici všem projektům v organizaci na jednotném místě tak, aby je bylo možné použít pro potřeby plánování, odhadování či definici rizik. Standard vývojového prostředí je aktuálně definován na úrovni celé organizace, vzhledem ke specifikům jednotlivých projektů by bylo vhodnější je definovat na úrovni jednotlivých oddělení.

Pro lepší přehlednost jsem se rozhodl možné oblati zlepšení rozdělit do kategorií. Sumarizaci lze najít v tabulce 4.2.

Příručka projektu	Sumarizace cílových dovedností. Zlepšení eskalačního konceptu a podávání zpráv. Detailní plánování práce, zlepšení odhadů. Revize posloupností provádění na všech úrovních. Koncept revize technické dokumentace. Sumarizace jazykově specifických konvencí (Coding rules).
Plán kvality	Metriky - sledování, vyhodnocování.
Revize	Data, Statistiky
Testování	Použití regresních testů. Velký obrázek zachycující testovací strategii. Popis testování ve vztahu ke specifikům agilního vývoje. Systematické dokumentování výsledků tesků a integrace.
Konfiguračního řízení	Plán kontinuální inregrace ve vztahu k testům.
Inovace a patenty	Lepší komunikace možností v týmu.

Tabulka 4.2: Možné oblasti zlepšení rozdělené do kategorií.

Závěrem je třeba také říci, že aktuální situace ve firmě je po stránce CMMI velmi dobrá a s přehledem splňuje požadavky úrovně o jedna nižší, než si společnost stanovila za cíl získat. Po zapracování uvedených doporučení by nemělo být problémem dosáhnout požadované úrovně nebo ji i překonat.

Kapitola 5

Navrhovaná řešení

V následující kapitole budou prezentována navrhovaná řešení pro oblasti, které vyplynuly z analýzy současného stavu a byly popsány v kapitole 4. Potencionální zlepšení byla diskutována s konzultantem ze společnosti Siemens.

5.1 Možná řešení

Analýza současného stavu ve společnosti poukázala na některá místa či procesní oblasti, které vyžadují další rozvoj tak, aby byly splněny požadavky definované modelem CMMI. Nutno však podotknout, že celková úroveň kvality a procesů ve společnosti Siemens je na velmi dobré úrovni a pro zapracování uvedených zdokonalení by společnost, neměla mít větší problém s dosažením požadovaného Stupně vyspělosti modelu CMMI. Dále budou popsány návrhy možných řešení.

5.1.1 Odhadování Uživatelských scenářů

Odhadování Uživatelských scenářů probíhá v současné chvíli v konkrétních jednotkách, kterými jsou hodiny potřebné pro implementaci daného scénáře. Při plánování se nepoužívá možností, které nabízí metodika Scrum, především se jedná o Plánovací poker.

Bylo by vhodnější sestavit stupnici pro ohodnocení jednotlivých scénářů a za použití kartiček s těmito hodnotami provést kvalifikovaný odhad pracnosti. Tato změna přinese nejen možnost abstraktnějšího přemýšlení o objemu práce, ale především umožní sledovat Výkonnost týmu v jednotlivých iteracích vývoje.

5.1.2 Aktivita Úkolů

Úkoly potřebné pro implementaci konkrétního Uživatelského scénáře jsou správně odhadovány v hodinách a tento odhad je velmi dobrým předpokladem pro zobrazení Grafu zbývající práce.

Je však nutné při plánování pamatovat také na to, že každý úkol by měl mít přiřazen také atribut udávající aktivitu, například vývoj, dokumentace, analýza, testování. Přiřazení aktivity umožňuje sledovat jednak objem práce, který chybí vykonat u dané aktivity v aktuálním Sprintu, ale také nabízí jedinečný pohled na poměr aktivit naplanovaných pro aktuální Sprint.

5.1.3 Graf zbývající práce

V současné chvíli nemá tým možnost sledovat aktuální průběh Sprintu, není schopen jednodušše a přehledně vidět jak moc se mu daří plnit plán aktuálního Sprintu a tím pádem nemá možnost objektivně vidět, zda již náhodou není mimo plán. V případě Grafu zbývající práce se jedná o jednu ze základních možností sledování průběhu vývoje, a proto je velmi důležité tuto možnost mít.

5.1.4 Graf Výkonnosti týmu

Se správným odhadováním Uživatelských scenářů také souvisí možnost zobrazení grafu Výkonnosti týmu, který rovněž není aktuálně k dispozici. Tento graf zcela jistě napomůže týmu v lepším určení objemu práce, který bude schopen během následujícího Sprintu realizovat a také může pomoci lépe sledovat vývoj týmu a bude z něj na první pohled patrné, zda se tým v průběhu času zlepšuji nebo zhoršuje, případně stagnuje.

V neposlední řadě tento ukazatel může být užitečný pro Vlastníka projektu, který s jeho pomocí může mnohem přesněji určovat termíny uvolnění dalších verzí produktu k zákazníkovi.

5.1.5 Vzájemné ohodnocení

Pro tým by bylo přínosné mít možnost zaznamenávat výsledek Revize kódu přímo do šablony na serveru, na němž je realizován systém pro správu kódu. Bylo by tak zajištěno jednoznačné provázání kódu a Revize kódu. Výsledek revize by měl být řádně sledován a přijaté závěry systematicky zapracovány.

5.1.6 Rychlý přístup ke grafům

V rámci Denního Scrum by měl mít tým možné rychlého přístupu ke Grafu zbývající práce. Jelikož je tato denní porada metodikou Scrum definována jako maximálně 15-ti minutová bylo by v rámci zvýšení efektivity dobré, kdyby byl pořízen tablet, který by byl používán pro rychlé zobrazení aktuálního grafu během tohoto setkání.

5.1.7 Kalkulace dostupnosti

Při plánování týmu chybí možnost jednoduchého zadání dostupnosti pro plánovaný Sprint, která by umožnila rychlý výpočet počtu hodin, které mají být vyčleněny pro jednotlivé projektové aktivity (oprava chyb, implementace nový vlastností, neproduktivní hodiny, úprava stávajícího kódu). Procentuální vyjádření těchto aktivit udává zákazník. V současné době tým používá pro tyto výpočty týmovou wikipedii, kde do připravené šablony dopisuje hodnoty, které jsou dopočítávány ručně. Tento proces není příliš efektivní a je zde prostor pro vylepšení.

5.1.8 Seznam bodů pro provádění

Pro jednotlivé činnosti by bylo dobré mít k dispozici seznam bodů, které musí být splněny při provádění dané činnosti. Aktuálně je tento seznam sestaven pro Revizi kódu, ale bylo by velmi užitečné jej mít k dispozici i u dalších projektových aktivit, jako je například Plánování nebo Retrospektiva. V současné době pro tyto aktivity tým používá šablonu, která není zcela kompatibilní s požadavky modelu CMMI.

5.1.9 Sumarizace počtů

Členové vývojového týmu nemají možnost jednodušše a přehledně vidět počet různých ukazatelů, jakými jsou například počet Úkolů, počet Testovacích případů, počet implementovaných testů jednotek. Zvláště důležitá je informace o počtu zbývajích defektů, které vyžadují opravu.

5.1.10 Testovací strategie

Na projektu existuje testovací strategie. Je však nutné ji zdokumentovat více formálněji a měla by být více zaměřena na specifika agilního vývoje. V ideálním případě by strategie mohla být doplněna o velký obrázek názorně zobrazující celý koncept. Velmi důležité je také seznámit všechny Členy vývojového týmu s její přepracovanou podobou.

5.1.11 Vizualizace konfiguračních položek

Konfigurační položky, které jsou pro projekt klíčové by měly být systematicky sledovány alespoň jednou denně. Pro jejich sledování by mohla být vytvořena jednoduchá vizualizace, která by zobrazovala například to, v jakém stavu se aktuálně nachází jednotlivé vývojové větve produktu nebo zda uspěly všechny spouštěné testy.

5.1.12 Plán uvolňování

V rámci plánování budoucích uvolňování produktu by mohl být k dispozici nástroj nebo alespoň jednotné místo, na němž by všichni členové týmu nalezli potřebné informace o termínech plánovaných uvolnění další verze, které plánuje Vlastník produktu.

5.1.13 Historická data

Model CMMI vyžaduje možnost vizualizace a dostupnosti historických dat pro potřeby projektu, proto je velmi důležité mít tato data jednodušše a přehledně k dispozici, ať už se jedná o graf udávající Výkonnost týmu nebo Graf zbývající práce zobrazující průběh předchozích Sprintů.

5.2 Konzultace řešení

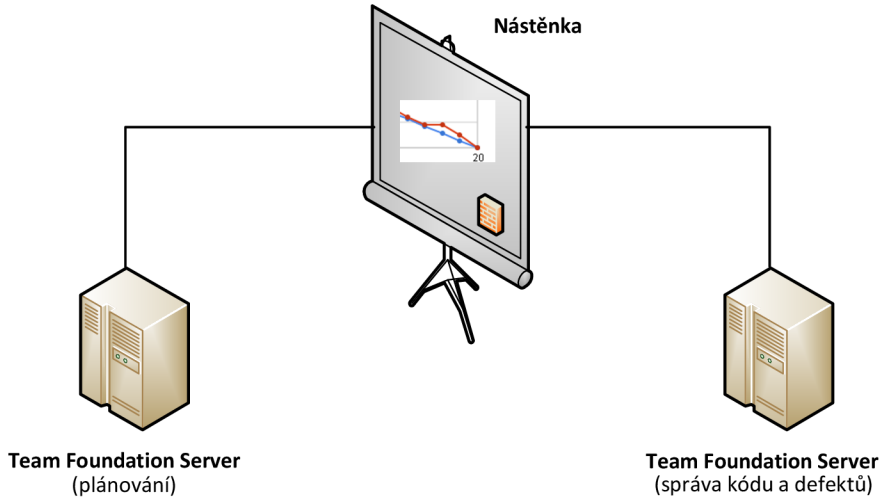
Zjištěné nedostatky, které vyplynuly z analýzy, byly představeny konzultantovi ze společnosti Siemens Ing. Janu Vernerovi a bylo zjištěno, na základě informací získaných z této diskuse, že většinu problémů způsobuje systém dvou serverů pro podporu vývoje, mezi kterými chybí jakákoliv vazba, a tudíž není možné informace uložené na těchto dvou místech, jednoduše spojit dohromady a získat tak ucelený přehled a zobrazit potřebné informace.

Problémem jsou dva Team Foundation Servery[1], z nichž na jednom probíhá plánování, zatímco na druhém jsou uloženy zdrojové kódy a probíhá na něm sledování defektů již existující verze aplikace. První server obsahuje kompletní Produktový katalog a historii všech Sprintů, zatímco druhý kompletní historii vývoje aplikace po stránce kódu.

V dohledné době není sloučení těchto systémů do jednoho technicky možné, a proto bylo na základě konzultace s Ing. Vernerem navrženo dále popsané řešení, které by mělo uvedené systémy propojit a poskytnout tak chybějící informace pro projektový tým, které v současné

[1] http://msdn.microsoft.com/en-us/vstudio/ff637362.aspx

době není možné lehce získat. Vizualizace navrhovaného propojení je uvedena na obrázku 5.1.



Obrázek 5.1: Vizualizace propojení dvou serverů pro podporu vývoje.

Navhované řešení pokryje většinu popsaných zlepšení uvedených v této kapitole.

5.3 Popis řešení

Konzultantovi ze společnosti Siemens byla představena výše uvedená zjištění společně s návrhem možných řešení. Jak již bylo řečeno, velkou část nedostatků způsobují dva systémy pro podporu vývoje.

Bylo rozhodnuto, že v rámci implementační části diplomové práce bude implementována nástěnka (dashboard), která bude poskytovat všechny důležité informace na jednom místě. Bude se jednat o webovou aplikaci, která by měla být primárně začleněna do aplikace Sharepoint[2] formou zásuvného modulu. V případě nemožnosti tento způsob integrace realizovat se pak bude jednat o samostatnou webovou aplikace umístěnou na firemním serveru společnosti Siemens.

Nástěnka by měla všem členům týmu zobrazovat data získaná z obou zmiňovaných systémů. Rovněž by měla týmu poskytovat potřebnou podporu pro plánování. Bude disponovat těmito funkcemi:

- Zobrazení Grafu zbývající práce.
- Plánování dostupnosti členů týmu.
- Zobrazení počtu nevyřešených defektů, celkového počtu Úkolů, Testů jednotek nebo Testovacích případů.
- Zobrazení procentuálního zastoupení jednotlivých aktivit.
- Zobrazení Grafu výkonnosti týmu.
- Zobrazení dalších metriky užitečných pro projektový tým.

[2] http://sharepoint.microsoft.com/cs-cz/Pages/default.aspx

Uvedené funkce budou primárně dostupné pro aktuální Sprint, nicméně aplikace bude poskytovat rovněž možnost výběru libovolného Sprintu z minulosti, čímž bude zajištěna možnost vizualizace historických dat.

Ostatní navrhovaná vylepšení budou zavedena nebo realizována současně s implementací aplikace, čímž by mělo dojít ke zvýšení kvality celé řady procesních oblastí ve společnosti a napomoci k získání požadovaného stupně kvality.

Při implementaci nástroje bude kladen důraz na jeho budoucí rozšiřitelnost a možnost jednoduchého použití pro ostatní týmy uvnitř organizace.

Kapitola 6

Specifikace a návrh aplikace

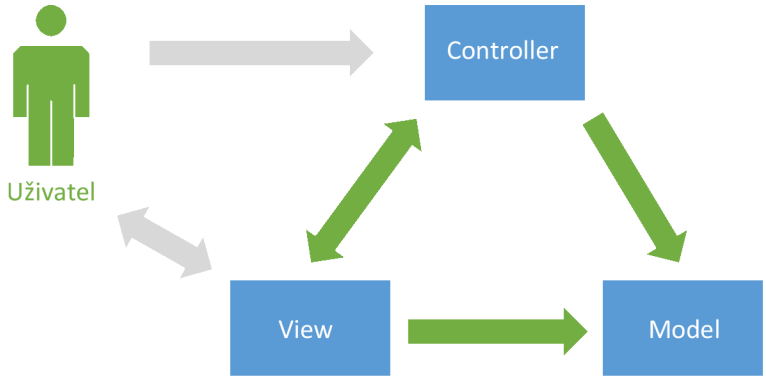
Kapitola v úvodu zmiňuje hlavní použité technologie, zabývá se detailní spefikací a návrhem webové aplikace, která je předmětem implementačním části diplomové práce. Obsahuje kromě specifikace požadavků také schéma databáze a nechybí ani UML diagramy zachycující možnosti uživatele při práci s aplikací.

6.1 Představení technologií

Následující podkapitola představuje tři hlavní technologie, které byly použity při implementaci a je na ně dále v textu odkazováno. Pro vývoj bylo použito prostředí Microsoft Visual Studio 2010, které není dle mého názoru nutné podrobněji představovat.

6.1.1 Model-View-Controller

Architektura MVC (Model-View-Controller) se skládá ze tří oddělených logických částí, které je možné upravovat samostatně tak, aby byl dopad provedených změn na ostatní části co nejmenší. Model reprezentuje datovou vrstvu společně s bussiness logikou aplikace. View se stará o zobrazení grafického uživatelského rozhraní a Controller má na starosti správnou funkci aplikační logiky. Provázání architektury MVC ve vazbě na uživatele lze vidět na obrázku 6.1.



Obrázek 6.1: Architektura Model-View-Controller.

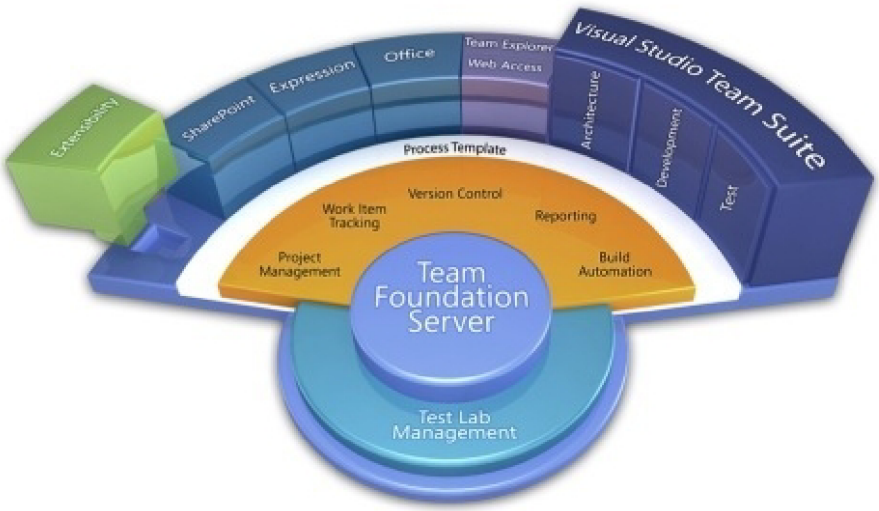
V architektuře MVC existují pouze dva přímé odkazy na Model, a to u Controlleru a View. Controller jej potřebuje pro úpravu dat, zatímco View pro jejich zobrazení. V praxi

a v závislosti na konkrétní implementaci MVC je ještě poměrně častá vazba mezi Controllerem a View (vazba může být jednosměrná nebo také obousměrná). Uživatel vstupuje do tohoto procesu jednak na úrovni View, kde je zpracováván jeho vstup (například validace zadaných hodnot) a dále také na úrovní Controlleru, kde jsou prováděny požadované akce. [2]

Při implementaci aplikace bude použit framework společnosti Microsoft ASP.NET MVC aktuální verze 4. Platforma poskytuje vývojáři technologie pro tvorbu dynamických webových stránek, zahrnuje podporu pro návrhové vzory, vývoj řízený testy (TDD) a v neposlední řadě také podporu pro agilní způsob vývoje. Díky striktnímu dělení kódu aplikace do výše popsaných vrstev se stává kód lépe spravovatelným. Samozřejmostí je respektování aktuálních webových standardů. Rád bych také zmínil, že tato platforma je Open Source.

6.1.2 Team Foundation Server 2010

Team Foundation Server 2010 je platforma společnosti Microsoft pro podporu vývoje a týmovou spolupráci poskytující týmům kompletní možnost řízení vývoje životního cyklu softwaru. Přináší mimo jiné možnosti spojené se správou zdrojového správy kódu nebo podporu pro automatické sestavení. Přehled hlavních funkcí lze vidět na obrázku 6.2. [8]



Obrázek 6.2: Přehled hlavních funkcí Team Foundation Server 2010.

Platforma umožňuje aplikovat některou z nabízených šablon, která server přízpůsobí požadavkům různých vývojových metodik, zkoumaný projekt ve společnosti Siemens využívá šablonu pro agilní metodiku Scrum.

Pro ukládání položek z katalogů využívá Team Foundation Server OLAP[1] databázi, které umožňuje nahlížet na jednotlivé položky zpětně v konkrétním čase. Tato funkcionalita bude využita při implementaci této diplomové práce.

Nástroj pro připojení k serveru je dostupný nejen jako součást produktu Microsoft Visual Studio, ale lze jej používat také jako samostný program. Platforma poskytuje podporu nejen pro jazyky rodiny .NET, ale také například pro jazyk Java. Samozřejmostí je možnost přístupu prostřednictvím webového rozhraní, které přináší stejně možnosti jako připojení k serveru prostřednictvím klienta.

[1] http://datamining.xf.cz/view.php?cisloclanku=2002102808

6.1.3 nHibernate

Jedná se o platformu poskytující programátorovi ORM (Object-Relational mapping) podporu pro mapování datových tříd a jejich vlastností na tabulky SQL databáze a jeho datové typy. Framework je dostupný pro několik programovacích jazyků, včetně platformy .NET. Platforma také poskytuje přímou podporu pro všechny databázové operace včetně podpory transakcí. [11]

Datová třída musí splňovat pouze dva požadavky, a to že bezparametrický konstruktor nesmí být veřejný a všechny vlastnosti jazyka musí označeny jako virtuální, neboť nHibernate za běhu nad nimi vytváří dynamické proxy, které pak používá při práci s datovými objekty.

Pro snadnější zápis mapování mezi třídami a databázovými tabulkami poskytuje Fluent API, které přináší možnost funcionálního zápisu, příklad mapování vztahu One-to-Many pro zákazníka s několika adresami lze najít níže.

```
public class Employee
{
  public virtual int Id { get; protected set; }
  public virtual string Name { get; set; }
  public virtual Ilist Adresses { get; set; }
}

public class Adress
{
  public virtual int Id { get; protected set; }
  public virtual string City { get; set; }
  public virtual Employee Employee { get; set; }
}

public class EmployeeMap : ClassMap<Employee>
{
  public EmployeeMap()
  {
    Id(x => x.Id);
    Map(x => x.Name);
    HasMany(x => x.Adresses).CascadeAll();
  }
}

public class AdressMap : ClassMap<Adress>
{
  public AdressMap ()
  {
    Id(x => x.Id);
    Map(x => x.City);
    References(x => x.Employee).CanNotBeNull();
  }
}
```

6.2 Specifikace požadavků

Na základě analýzy aktuálního stavu procesních oblastí a požadavků modelu CMMI konkrétního stupně, který si analyzovaná společnost stanovila za cíl, byly identifikovány oblasti, které je potřeba dále rozvíjet. Implementace dále uvedené aplikace by měla odstranit většinu těchto nedokonalostí, výrazně napomoci týmu od přílišné administrativy a v neposlední řadě

také přispět k navýšení úrovně kvality procesů v organizaci. Cílem implementační části bylo také co nejvíce stávajících procesů automatizovat.

6.2.1 Požadavky společnosti

Ze strany společnosti nebyly pro implementaci kladena žádná konkrétní omezení, nicméně bylo doporučeno použít stávající infrastrukturu v podobě již existujících databázových a webových serverů a nejčastěji používaných technologií, které ovládají zaměstnanci společnosti. Ovládání aplikace by mělo být co nejintuitivnější a poskytovat potřebné informace rychle a přehledně na jednom místě, ideálně na firemním serveru.

Co se týče funkčních požadavků, bylo požadováno, aby aplikaci bylo možné v budoucnu použít pro ostatní týmy a dále také, aby bylo možné zobrazovat například Graf zbývající práce po zadání potřebných parametrů do adresního řádku bez nutnosti otevírání celé aplikace a použití navigace prostřednictvím menu.

6.2.2 Integrace dvou serverů

Jelikož se data nacházejí na dvou servech, které není v současné době technologické možné propojit, ani se o jejich propojení do budoucna neuvažuje, je nutné pamatovat již na tuto skutečnost při návrhu aplikace, především pak při tvorbě schématu databáze. Data je potřeba ukládat odděleně tak, abychom je byli schopni zobrazit jednak pro konkrétní server, ale také souhrně bez ohledu na to, odkud pocházejí. Vzhledem k tomu, že na obou serverech běží Team Foundation Server 2010, neměla by jejich integrace činit problém, za předpokladu, že bude použito jednotné API poskytované touto technologií.

6.2.3 Graf zbývající práce

Tým nemá v současné době jednoduchou možnost zobrazení Grafu zbývající práce, a proto nemá přehled o průběhu aktuálního Sprintu. Schopnost rychlé reakce na případné odchylky oproti plánu je tak velmi pomalá a neefektivní. Graf zbývající práce by měl zobrazovat souhrnně všechny zbývající hodiny na obou serverech v jednom přehledném grafu. Pro hodnoty v jednotlivé dny se budou používat zbývají hodiny na Úkolech ze Sprint katalogu tak, jak je definováno agilní metodikou Scrum.

6.2.4 Graf Výkonnosti týmu

Pohled na práci týmu napříč Sprinty by měl být zajištěn prostřednictvím Grafu Výkonnosti týmu, který bude zobrazovat všechny uzavřené Sprinty. Tento graf bude zobrazovat poměr naplánovaných Ohodnocení scénářů vůči skutečně splněným. Tým bude mít možnost vidět, zda se postupem času zlepšuje nebo naopak zhoršuje. Dále bude mít možnost srovnat, nakolik se mu daří plnit plán v jednotlivých Sprintech.

6.2.5 Aktuální Výkonnost

S grafem Výkonnosti týmu úzce souvisí také kalkukace aktuální Výkonnosti. Metodika výpočtu této hodnoty se různí, pro potřeby této diplomové práce bude použita ta, která počítá průměrnou hodnotu z posledních tří uzavřených Sprintů. Získaná hodnota slouží jako podklad pro Vlastníka produktu, který má díky ní možnost relativně přesně odhadovat další milníky ve vývoji, jako jsou například termíny uvolnění aplikace. S touto hodnotou také

pracuje tým při fázi plánování, kdy může porovnat aktuální hodnotu s tím, co si právě naplánoval a případně odebrat či naopak přidat položky do Sprint katalogu.

6.2.6 Automatická kalkulace dostupnosti

Při fázi plánování je tým velkou část doby zdržován výpočtem dostupných hodin pro nadcházející Sprint. Aktuálně musí členové či některý člen týmu zjistit, kolik pracovních dnů má další Sprint (nesmí zapomenout na státní svátky a firemní volna), na základě tohoto údaje členové týmu podle svého úvazku zadají do tabulky na týmové wikipedii počet dní přítomnosti pro jednotlivé týdny a poté je nutné ručně provést celkový součet. Prostor pro zanesení chyby v kterékoliv fázi je více než patrný, proto by bylo velkým přínosem, kdyby byly všechny hodnoty spočteny automaticky na základě začátku a konce Sprintu. Členové týmu by jen zadávali svou nedostupnost (dovolené, školení, atd.). Úvazek by neměl být také pevně definován, ale aplikace by měla podporovat zadat pro každý Sprint jiný úvazek u konkrétního člena týmu.

6.2.7 Plánování s podporou aktivit

Nově by plánování mělo zahrnovat také přiřazení aktivit k jednotlivým úkolům. Při plánování by měl mít každý člen týmu přiřazen poměr rozdělení svých hodin na jednotlivé předem definované aktivity. Poměr bude přebírán automaticky z předchozího Sprintu a bude umožňovat manuální úpravu. Na základě rozdělení pro jednotlivé projektové aktivity bude generován souhrn dostupných hodin pro všechny členy týmu, jejich úvazky a definované poměry. Skutečně naplánované Úkoly budou agregovány podle aktivit a aplikace bude zobrazovat součet naplánovaných hodin pro jednotlivé aktivity.

6.2.8 Autentizace

Aplikace bude mít omezený přístup, který bude využívat doménových účtů jednotlivých uživatelů uvnitř společnosti. Přístup bude definován na webovém serveru prostřednictvím definované skupiny uživatelů, všichni uživatelé mimo tuto skupinu nebudou mít do aplikace přístup. Tento způsob autentizace se označuje jako „Single sign-on“[2] a je v korporátní sféře velmi oblíbený, neboť umožňuje uživateli přihlášení k počítači a automatické předávání jeho údajů všem aplikacím v případě potřeby, bez nutnosti dalšího přihlašování.

6.2.9 Cachování

Pro rychlejší odezvu aplikace bude potřeba data cachovat, především protokoly z Plánování a Revize Sprintu. Položky Sprint katalogu, které byly naplánovány, stejně jako položky z Revize nebudou cachovány, neboť zobrazení položek pomocí API je velice rychlá operace a bude realizována pomocí operátou `asOf` z API Team Foundation Serveru. Hodnoty pro jednotlivé dny Grafu zbývající práce budou uloženy pro všechny „uzavřené dny“. Uzavřeným dnem se rozumí všechny dny Sprintu, které předcházejí aktuálnímu dni. V případě potřeby bude aplikace také obsahovat možnost přegenerování cache.

[2] http://en.wikipedia.org/wiki/Single_sign-on

6.2.10 Historická data

Všechna data bude možno zobrazovat nejen pro aktuální Sprint, ale také pro všechny předešlé tak, aby bylo možné používat historická data především pro plánování. Tento požadavek je definován modelem CMMI a tým nemá aktuálně možnost jednoduché vizualizace dat z předchozích Sprintů.

6.2.11 Zvolené technologie

Aplikace bude implementována jako webová aplikace napsaná v jazyce C# za použití Framework ASP.NET. Základem architektury bude koncept MVC (Model-View-Controller). Velmi malá část celkové funkcionality bude implementována pomocí jazyka Javascript.

Při výběru vhodné webové technologie bylo zohledněno několik kritérií. Prvním kritériem byla možnost spravování aplikace ze strany zaměstnanců organizace, kteří jsou odborníky přes jazyk C#. Jelikož se jedná o společnost, která používá především technologie společnosti Microsoft, chtěl jsem se vydat touto cestou, aby se výsledná aplikace příliš neoddálila od firmou používaných, nepsaných standardů. V neposlední řadě bylo mým cílem osvojit si práci s novou technologií a rozšířit si tak své programovací schopnosti, proto jsem zvolil poslední dobou velice aktuální technologii MVC.

Podle původního plánu měla být aplikace implementována jako zásuvný modul do platformu Sharepoint. Při podrobném prozkoumání možností této platformy bylo zjištěno, že nepřináší kromě kalendářové komponenty, kterou by bylo možné použít pro zadávání nedostupnosti členů týmu v průběhu Sprintu, prakticky žádné další benefity, které by byly užitečné pro potřeby implementované aplikace. Na druhou stranu klade omezení na programátora, a proto bylo po dohodě s konzultantem rozhodnuto, že se bude jednat o samostatnou webovou aplikaci, běžící na firemním serveru, která bude v případě potřeby do platformy Sharepoint integrována pomocí `iframe` componenty.

Jelikož společnost provozuje ve své síti Microsoft SQL Server, bylo rozhodnuto, že pro persistenční vrstvu bude použita technologie dostupná v organizaci, a proto budou data ukládána do databáze na zmiňovaném serveru. Samotná aplikace bude umístěna na již existujicím webovém serveru a bude přístupná všem oprávněným uživatelům v podnikové síti.

Pro přístup k databázi bude používán Framework nHibernate, který umožňuje velice pohodlnou práci s databází, speciálně při použití jeho rozšíření označovaného jako Fluent API. Alternativou k tomutu frameworku se nabízelo použití Entity Frameworku, ale při porovnání programátorské přívětivosti a možností jsem se rozhodl na základě vlastních zkušeností použít právě nHibernate.

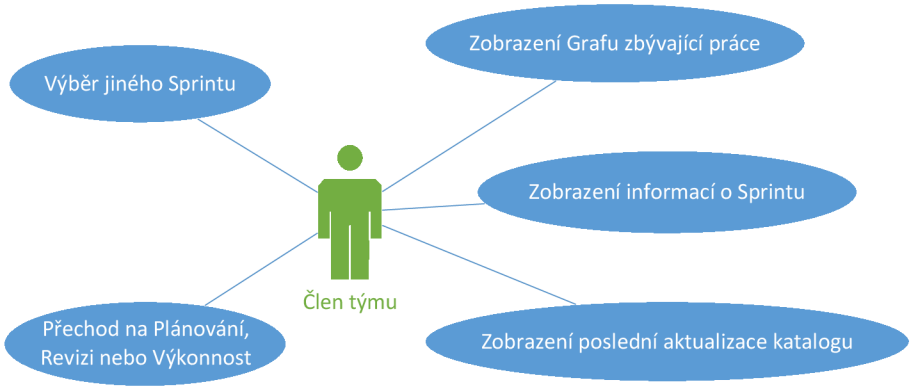
6.3 Specifikace případu užití

Při zadání webové adresy se uživateli zobrazí seznam všech dostupných týmů. Výběrem týmu dojde k přesměrován na první stránku. Od této chvíle se veškeré informace zobrazují pro zvolený tým a není možné jej měnit. Hlavní jádro aplikace bude nabízet čtyři stránky:

1. Graf zbývající práce
2. Plánování
3. Revize
4. Výkonnost

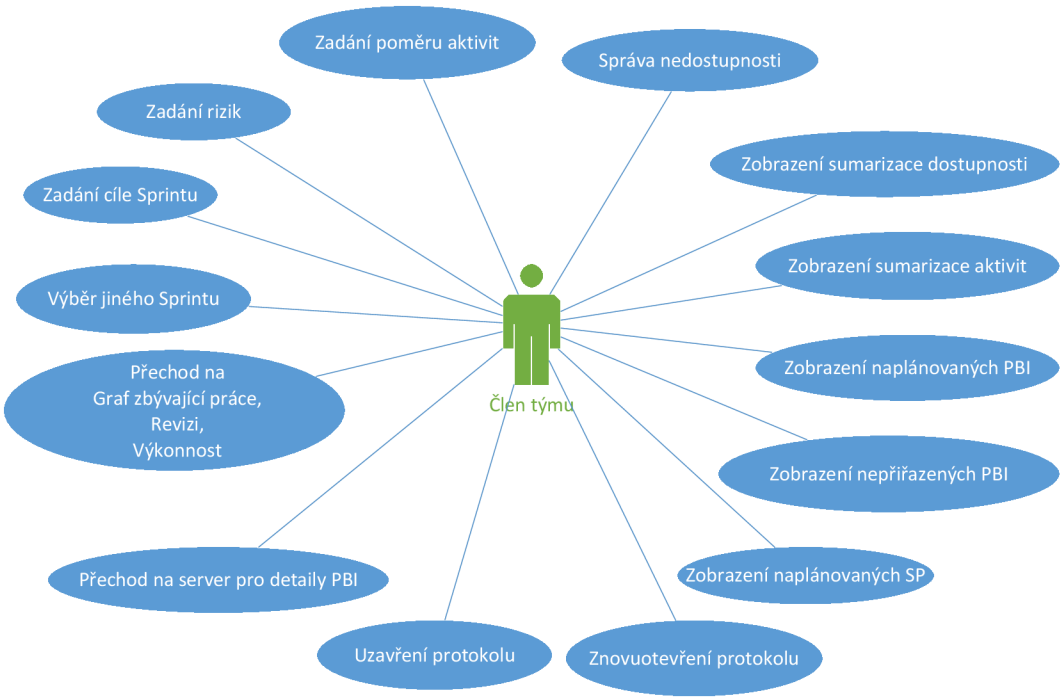
Hlavní stránkou byl zvolen Graf zbývající práce, neboť tato stránka bude zobrazována nejčastěji v průběhu Sprintu. U prvních tří uvedených stránek bude mít uživatel možnost vybrat požadovaný Sprint z nabídky, čímž bude docíleno možnosti vizualizace historických dat. Implicitně bude vybrán poslední Sprint z databáze. Pro jednotlivé stránky aplikace byly nadefinovány diagramy případu užití z pohledu člena týmu, které zachycují veškeré možné akce, které bude moct provádět na jednotlivých stránkách.

6.3.1 Graf zbývající práce



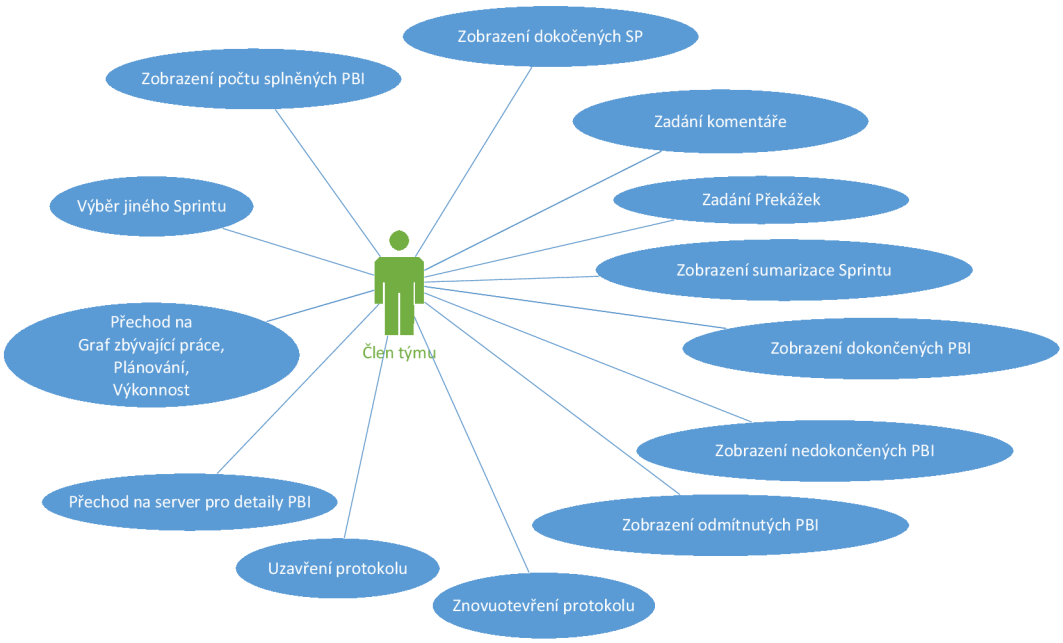
Obrázek 6.3: Diagram případu užití pro Graf zbývající práce.

6.3.2 Plánování



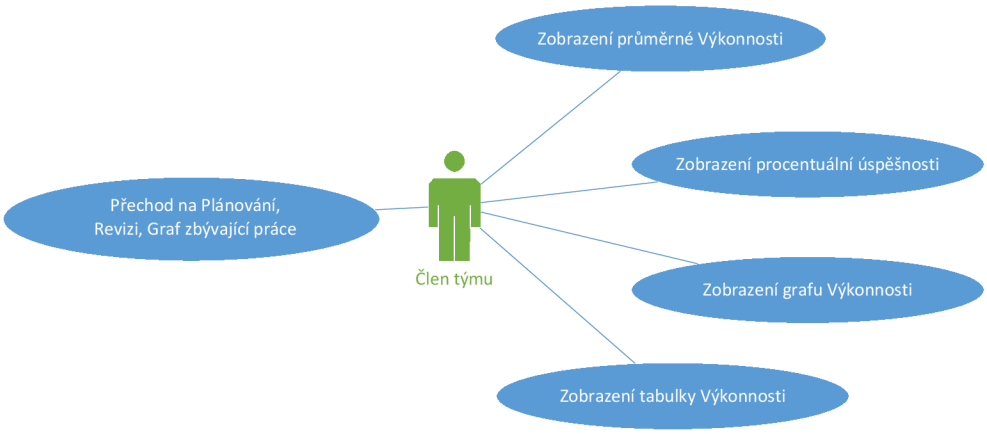
Obrázek 6.4: Diagram případu užití pro Plánování.

6.3.3 Revize



Obrázek 6.5: Diagram případu užití pro Revizi.

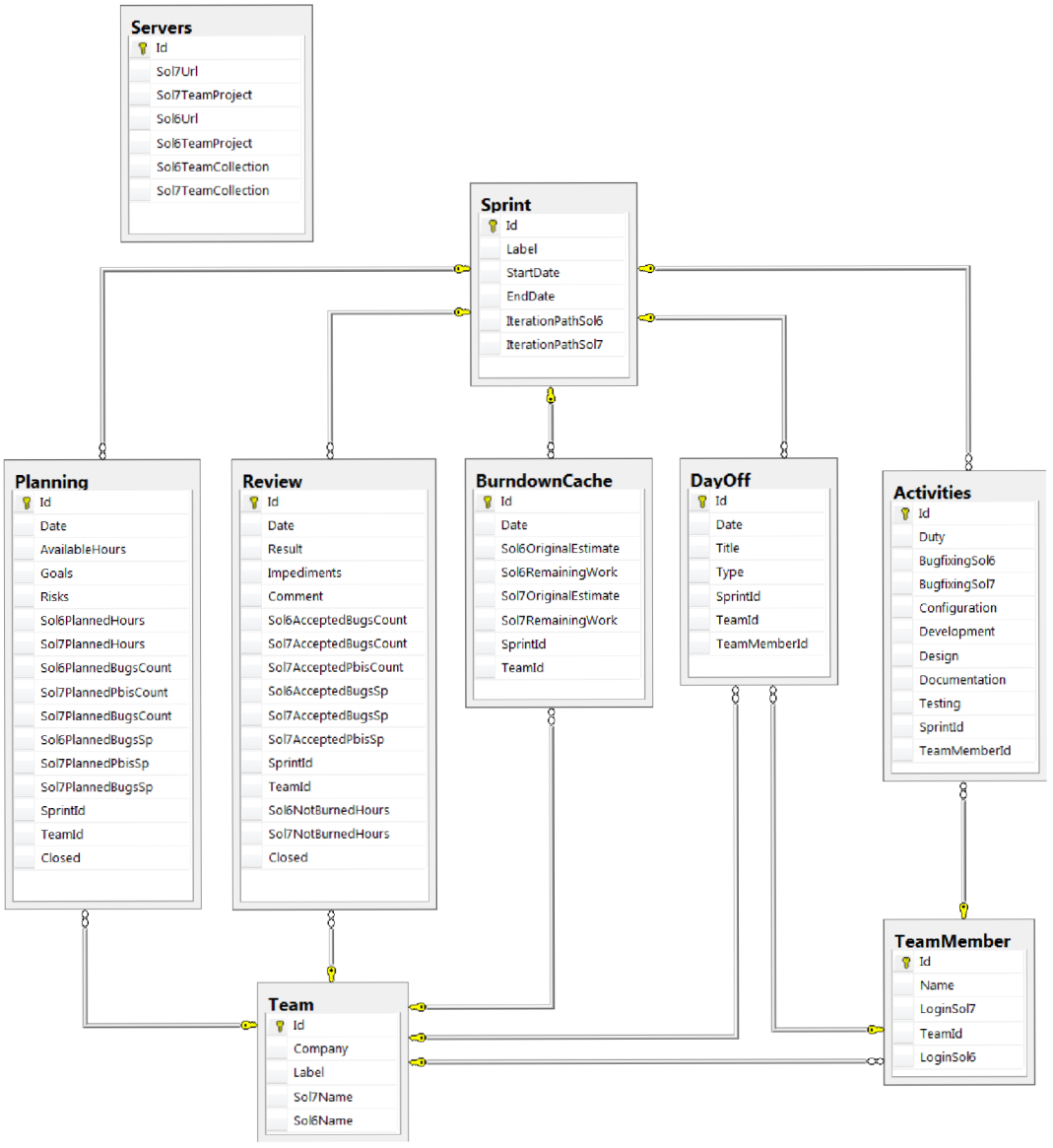
6.3.4 Výkonnost



Obrázek 6.6: Diagram případu užití pro Výkonnost.

6.4 Schéma databáze

Při analýze potřeb aplikace na uchovávání a cáchování dat bylo vytvořeno schéma databáze vyobrazené na obrázku 6.7. Vytvořené schéma pokrývá všechny požadavky specifikace aplikace uvedené v podkapitole 6.2.



Obrázek 6.7: Navržené schéma databáze.

Kapitola 7

Implementace a testování

Následující kapitola popisuje detaily implementace aplikace, představuje funkcionalitu nejen z pohledu uživatele, ale také z pohledu vývojáře webových aplikací. V závěru je podrobně rozepsána fáze testování.

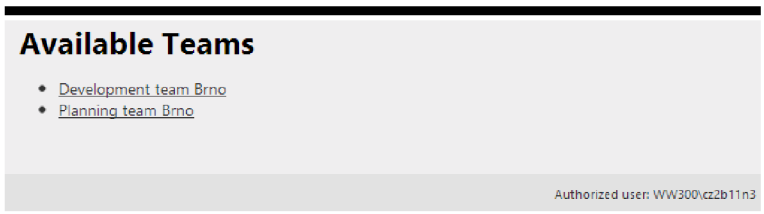
7.1 Popis aplikace

Implementace respektuje specifikaci požadavků uvedenou v kapitole 6.2 a obsahuje všechny požadované vlastnosti. Oproti specifikaci přináší několik dílčích zlepšení, které se ukázaly jako užitečné při uvádění aplikace do provozu, především pak při testování a sběru odezvy od reálných uživatelů uvnitř společnost. Databázová vrstva je implementována podle schématu z obrázku 6.7, rovněž byly dodrženy všechny zvolené technologie.

Programová dokumentace společně s demoverzí aplikace se nachází na přiložené CD, přesné umístění jednotlivých souborů je uvedeno v příloze této diplomové práce.

U všech tabulek s hlavičkou je implementována možnost řazení podle libovolného sloupce.

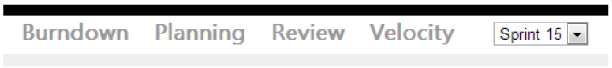
Při zadání adresy aplikace do adresního řádku se zobrazí úvodní obrazovka, která obsahuje seznam všech dostupných týmů. Po výběru konkrétního týmu je uživatel přesměrován na nástěnku daného týmu, na které se již všechna data vztahují ke zvolenému týmu. V pravém dolním rohu lze pak vidět informaci o aktuálně přihlášeném uživateli. Úvodní obrazovku lze vidět na obrázku 7.1.



Obrázek 7.1: Úvodní stránka aplikace.

Nástěnka týmu se skládá ze čtyř webových stránek, z nichž každá obsahuje informace specifické pro jinou část průběhu daného Sprintu. Zmíněné webové stránky lze rozdělit do dvou kategorií. Do první kategorie, která zahrnuje aktivity spojené s konkrétním Sprintem, spadá Graf zbývající práce, Plánování a Revize. Do druhé kategorie lze zařadit stránku zobrazující Graf výkonnost týmu, která je zobrazována pro všechny Sprinty najednou.

Každá obrazovka v pravém horním rohu obsahuje navigační menu, které umožňuje přechod mezi jednotlivými sekcemi. Webové stránky první kategorie navíc umožňují volbu požadovaného Sprintu. Při spuštění aplikace je implicitně vybrán poslední neuzavřený Sprint z databáze, uzavřeným Sprintem se rozumí Sprint s uzavřenou revizí a jako první je vždy načtena stránka s Grafem zbývající práce. Vizualizace položek menu se nachází na obrázku 7.2.

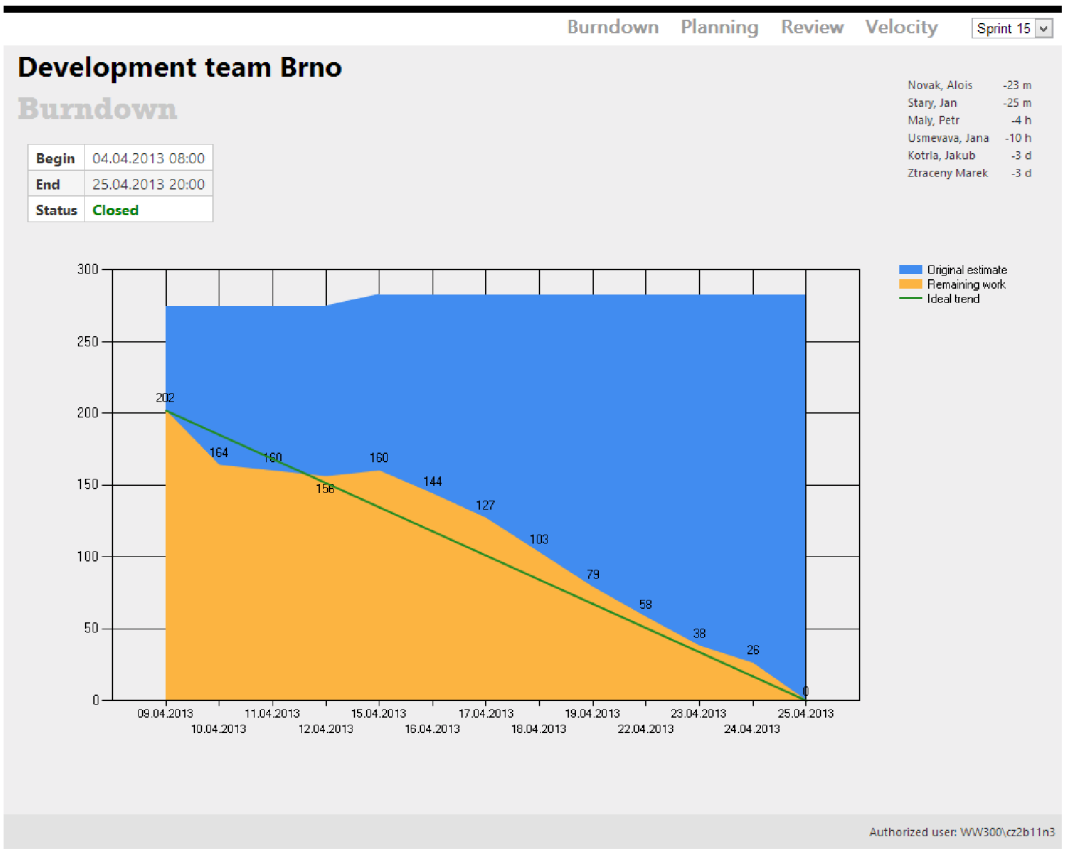


Obrázek 7.2: Menu aplikace s možností výběru Sprintu.

V následujících podkapitolách budou rozebrány jednotlivé webové stránky s podrobným popisem jejich funkcionality a uvedení jejich použití v návaznosti na fáze vývoje.

7.1.1 Graf zbývající práce

Stránka obsahuje zobrazení základních informací o zvoleném Sprintu a je používána ze všech stránek nejčastěji, konkrétně nejvíce při denních setkáních, méně časteji pak při Retrospektivě. Z tohoto důvodu je nastavena jako hlavní po výběru konkrétního týmu.



Obrázek 7.3: Obrazovka s Grafem zbývající práce.

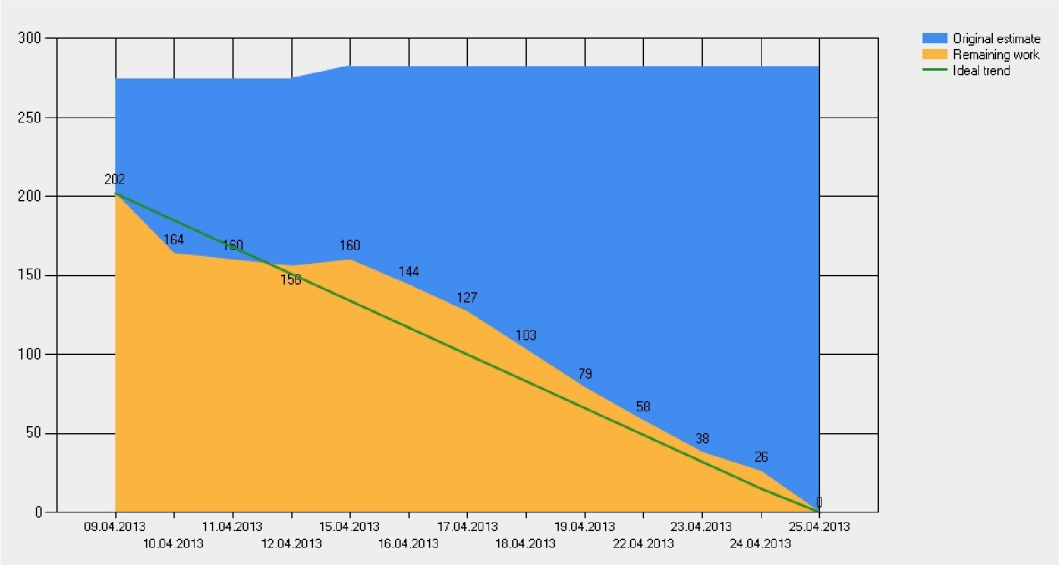
Mezi základní zobrazované informace patří datum začátku, konce a stavu zvoleného Sprintu. Sprint se může nacházet ve čtyřech různých stavech – Plánování, Vývoj, Revize,

Uzavřený. Stavy jsou pro lepší orientaci barevně odlišeny. Pokud se Sprint nachází ve stavu vývoje, je zde uvedena také informace o počtu zbývajících dní do konce vývojové fáze. Vzhled sumarizační tabulky je vyobrazen na obrázku 7.4.

Begin	04.04.2013 08:00
End	25.04.2013 20:00
Status	**Closed**

Obrázek 7.4: Tabulka s informacemi o zvoleném Sprintu.

Pod sumarizací základních informací se nachází Graf zbývající práce, který agreguje zbývající hodiny na Úkolech ze Sprint katalogu pro oba servery dohromady, které jsou zobrazeny formou grafu. Graf ukazuje poměr původně naplánované práce pro zvolený Sprint vůči aktuálně zbývající společně s trendem, kterým by se měl vývoj ubírat v průběhu Sprintu. Pro lepší orientaci v grafu je trend vyznačen zelenou barvou. Graf zbývající práce lze vidět na obrázku 7.5.



Obrázek 7.5: Graf zbývající práce pro skončený Sprint.

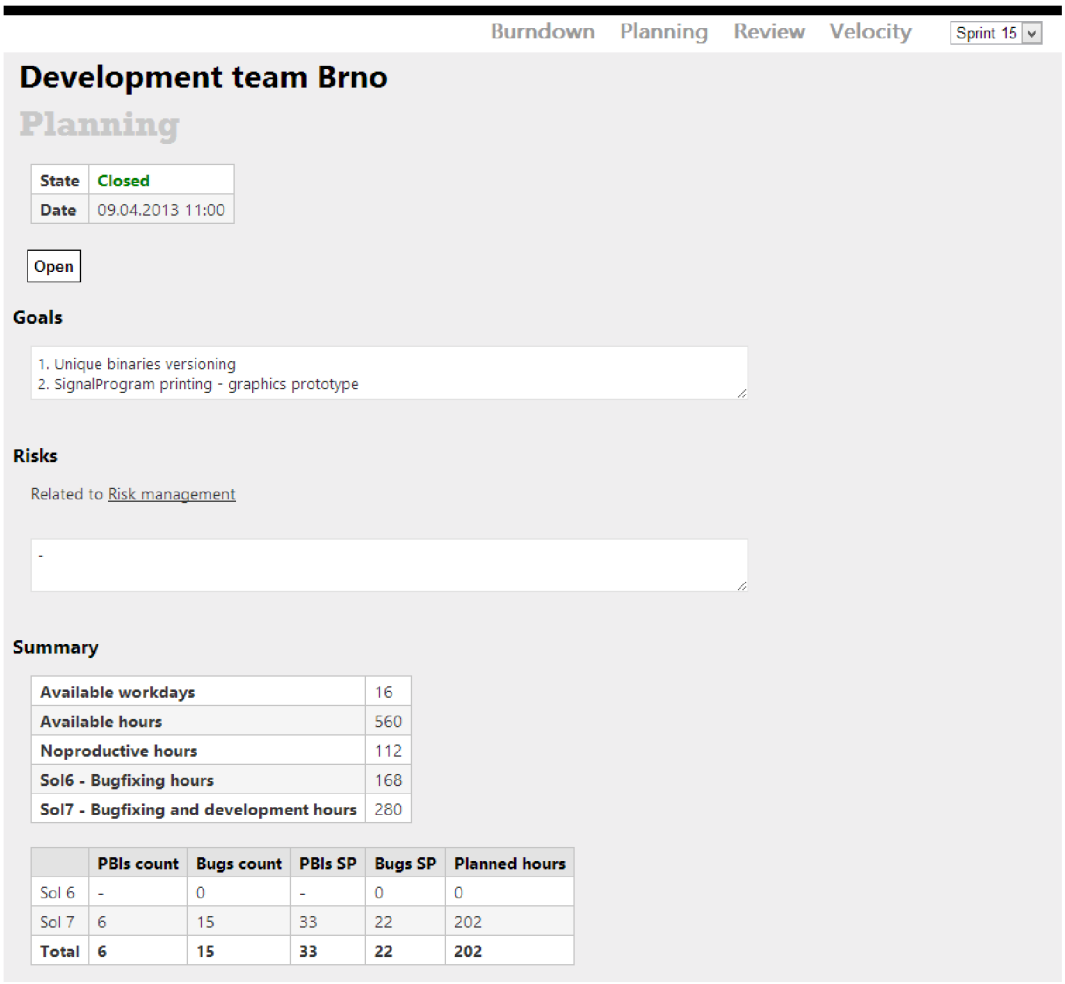
Dále tato obrazovka ukazuje interval od poslední aktualizace Úkolů ve Sprint katalogu u jednotlivých členů týmu. Díky tomuto pohledu je možné identifikovat možné problémy či zdržení ve vývoji a vhodným způsobem na ně reagovat. Snahou každého člena týmu by měla být aktualizace Sprint katalogu alespoň jednou denně.

7.1.2 Plánování

Stránka s plánováním je bezesporu nejkomplexnější částí webové aplikace. Kombinuje v sobě automatickou agregaci dat z dvou různorodých serverů společně a uživatelským vstupem v podobě zadávání nedostupnosti členů týmu v průběhu Sprintu. Dále také umožňuje zadání rozložení dostupnosti mezi jednotlivé aktivity, včetně definice úvazku, pro jednotlivé členy týmu.

Novak, Alois	-23 m
Stary, Jan	-25 m
Maly, Petr	-4 h
Usmevava, Jana	-10 h
Kotrla, Jakub	-3 d
Ztraceny Marek	-3 d

Obrázek 7.6: Přehled poslední aktualizace Sprint katalogu členy týmu.



Obrázek 7.7: První část stránky Plánování.

Na obrázku 7.7 lze vidět první část této stránky, která obsahuje kromě informace o tom, že plánování pro tento Sprint je již uzavřené (lze vidět také datum uzavření) mimo jiné Cíle Sprintu společně s Riziky, která byla shledána ve fázi plánování. Tyto informace byly do systému zadány manuálně některým členem týmu.

Dále lze na obrázku vidět počet dostupných pracovních dní (automaticky se odečítají svátky a firemní volna) a počet dostupných hodin, které jsou spočteny na základě úvazku člena týmu a jeho dostupnosti. Dostupné kvantum hodin je dále podle pravidel projektu rozděleno definovaným dílem mezi údržbu stávající verze aplikace a vývoj nové verze.

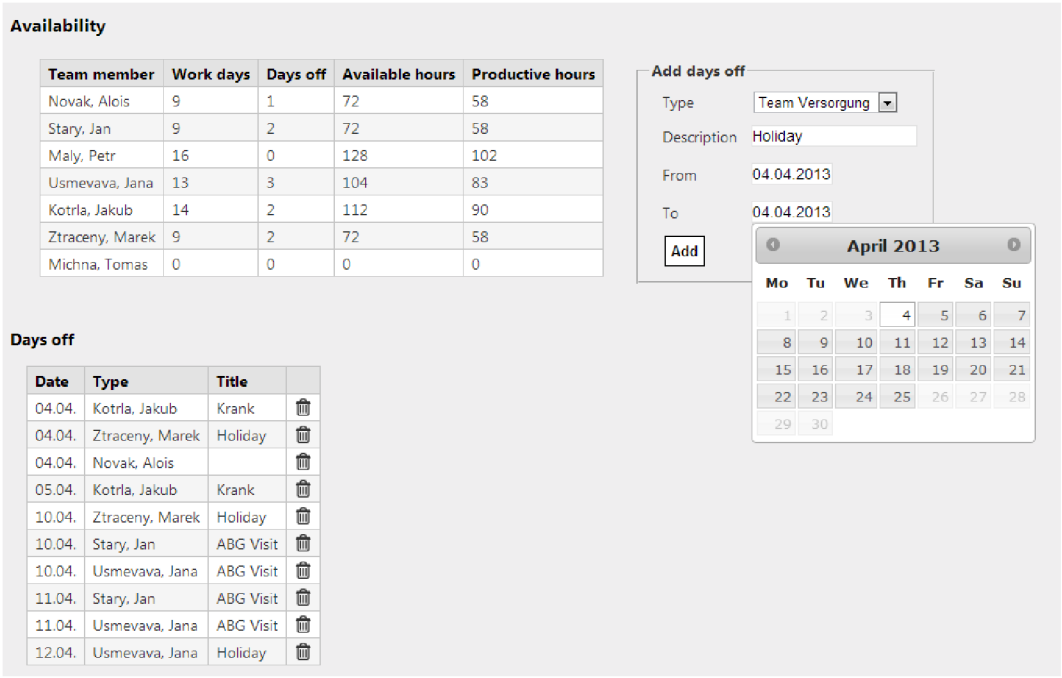
Poslední tabulkou sumarizace plánování je počet Defektů a položek ze Sprint katalogu, které jsou naplánovány pro nadcházející Sprint. Tabulka obsahuje také počet hodin a Ohodnocení těchto položek.

Dále budou popsány dvě hlavní části této obrazovky, kterými jsou Nedostupnost a Rozložení aktivit.

1. Nedostupnost

Nepřítomnost některého člena týmu v průběhu nadcházejícího Sprintu lze zadávat v další části stránky s Plánováním. V tabulce lze vidět souhrn počtu pracovních dní podle aktuálního úvazku zaměstnance a neproduktivní hodiny, které jsou rovněž předmět dohody se zákazníkem.

Z tabulky na obrázku 7.8 je lze vidět, že několik členů týmu má zadánu nedostupnost. Tuto skutečnost si lze ověřit také v tabulce pod sumarizací, kde se nachází seznam všech dnů nedostupnosti pro členy týmu nebo celý tým. Rovněž jsou součástí tabulky státní svátky nebo firemní volna.



Obrázek 7.8: Zobrazení nedostupnosti členů týmu v průběhu Sprintu.

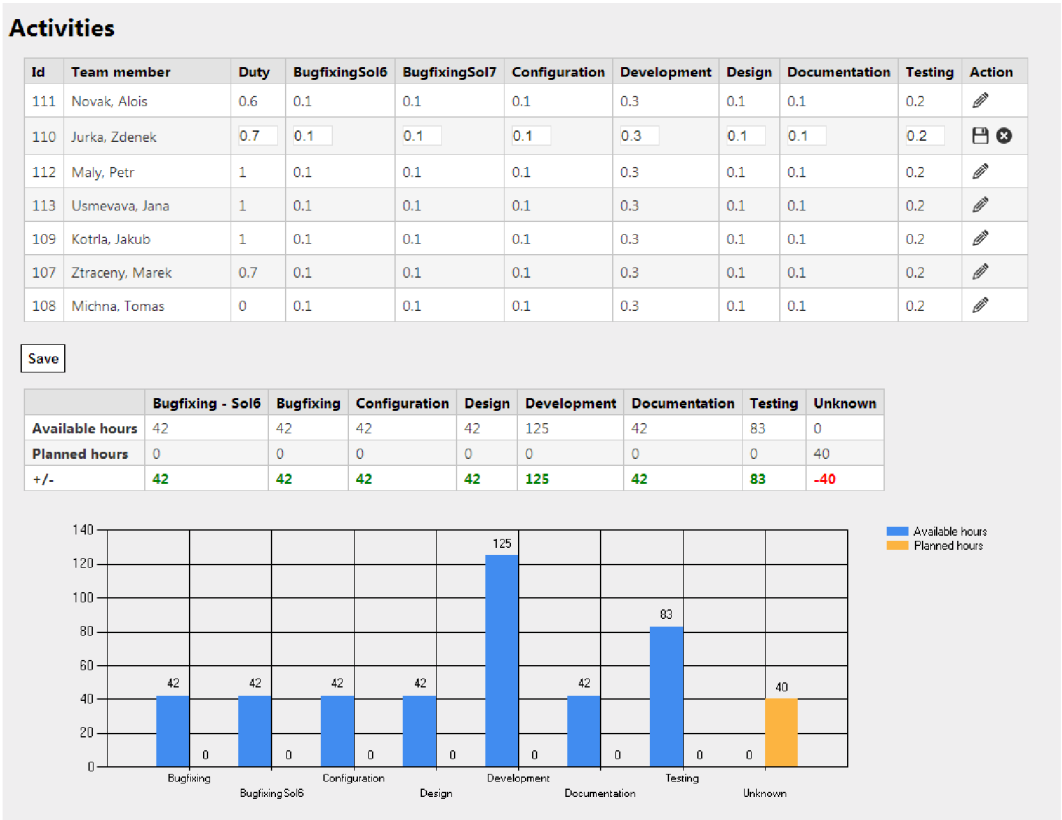
Týmovou nedostupnost nebo nedostupnost člena týmu lze smazat prostřednictvím ikony koše uvedené u každého záznamu. Firemní volna a státní svátky nelze odebírat, neboť se definují globálně pro celou aplikaci. V pravé části lze zadat nepřítomnost jednak členovi týmu, ale také celému týmu. Aplikace umožňuje zadat rozsah a kontroluje, aby nedostupnost nepřesahovala hranice Sprintu. Nepřítomnost může přidat kdokoliv kterémukoliv členovi týmu. V případě přidání nové nepřítomnosti je tabulka dostupnosti automaticky přepočítána stejně jako sumarizace hodin pro celý Sprint.

2. Rozložení aktivit

Jednotlivé Úkoly ze Sprint katalogu jsou při Plánování nově díky této diplomové práce rozdělovány do kategorií označovaných jako Aktivity. Tato kategorizace umožňuje lepší rozdělení práce a především sledování složení práce v rámci Sprintu.

Pro každého člena týmu lze zadat jeho úvazek pro plánovaný Sprint. Zadaná hodnota je používána pro již dříve zmíněnou dostupnost a počet hodin. Jednotlivé aktivity jsou pevně definovány a oprávněná osoba, zpravidla vedoucí projektu, definuje pro daný Sprint poměr rozložení aktivit pro jednotlivé členy. Na základě tohoto poměru je spočteno dostupné kvantum práce vyhrazené pro konkrétní aktivitu. Při plánování nového Sprintu se poměr rozložení aktivit získává z nastavení předcházejícího Sprintu.

Naplánované Úkoly ze Sprint katalogu jsou roztříděny do kategorií a počty naplánovaných hodin sečteny. Díky tomu lze na první pohled vidět, jak si tým stojí a zda například nenaplánoval příliš mnoho Úkolů na Konfigurační řízení, když lidé, kteří jsou oprávnění tuto činnost provádět, mají právě naplánovánu dovolenou. Lepší vizualizaci napomáhá zelená nebo červená hodnota, které udává rezervu či chybějícího hodiny pro danou projektovou aktivitu.



Obrázek 7.9: Správa aktivit na stránce pro Plánování.

Pro ještě lepší zobrazení těchto hodnot jsou data z tabulky reprezentována také sloupcovým grafem, který na první pohled ukazuje zastoupení projektových aktivit, resp. jejich hodin práce v kontrastu s dostupným kvantem. Vzhled části stránky pro správu Aktivit lze vidět na obrázku 7.9.

Poslední část stránky s Plánováním je věnována seznamu naplánovaných položek Sprint

katalogu agregovaných z obou serverů. U každé položky má uživatel možnost zobrazení kompletních informací o položce, které se provádí kliknutím na její titulek.

Za zmínku také stojí zobrazení všech položek, které nejsou korektně přiřazeny a musí být ručně zkontrolovány. Zpravidla se jedná o počet nepřesahující tři položky, ale cílem je mít všechny položky správně při uzavření přiřazeny tak, jak lze vidět na obrázku 7.10. Pokud by tomu tak nebylo, tým může vidět ve Sprint katalogu jiný objem práce než ten, který bude zobrazován Grafem zbývající práce.

Work items with incorrect path

all workitems are correctly assigned

Planned Workitems

Sol6

any workitems planned

Sol7

Id	Type	Title	Stack rank
11135	Bug	Sitraffic.Office.profiles file is not created correctly	
11136	Bug	UserProperties.cfg file cannot be found	
11234	Bug	SG editor: error messages are not translated	
11120	Bug	Core: Config wizard: Following pages are not grayed out when changing a value in detail editor of SG-Editor	
11121	Bug	Core: Config wizard: Following pages are not grayed out when changing a value in ZZ matrix	
10555	Bug	TLX-Ex-/Import: keine Übernahme des Flags für die Defaultwert-Anzeige	151
10606	Bug	Exception: Component change with corrupt data	180
9582	Bug	Core SG Editor: Eingabe bei Anzahl Geber führt zum Hängenbleiben des Core	209
11175	Bug	[Core] Printing attachment pref is not set to KnotenVersion	210
9462	Bug	Core: SG Editor: for SG types the minimum and transfer times cannot be edited	220
10707	Bug	Core: SG Editor: Transfer times cannot be edited correctly	420
11002	Product Backlog Item	CM Support	500
10654	Bug	The default username should be the windows login when starting the first time	800
10922	Bug	Order of sisi-properties in printing should be the same as in sisi wizard page	980
10661	Bug	Core: Print order and preview for SiSi wizard page	980
10939	Bug	Print preview should be available for SiSi Wizard page	980
9411	Product Backlog Item	Office bietet eine Rückfallmöglichkeit zum "Serverless" Profil beim Start, falls das gewählte Profil nicht funktioniert	2300
8786	Product Backlog Item	Printing of Signalprograms	2600
6248	Product Backlog Item	Unique versioning of binaries (P2 editors, OCX, InterOp assemblies, C++ projects, ...)	2700
11123	Product Backlog Item	Topology and Lageplan Inital Analysis	2700
7217	Product Backlog Item	Isolate MIE business logic	3100

Obrázek 7.10: Naplánované a nesprávně přiřazené položky ze Sprint katalogu.

Pokud je protokol z Plánování otevřen, může libovolný člen týmu provést jeho uzavření v horní části stránky. Po provedení této akce je Sprint automaticky převeden do stavu vývoje, na úvodní stránce se začně zobrazovat Graf zbývající práce a od této chvíle není možné dále protokol z Plánování upravovat. V případě, že je potřeba protokol dodatečně upravit, mají členové týmu možnost jej znova otevřít pomocí tlačítka `Open`.

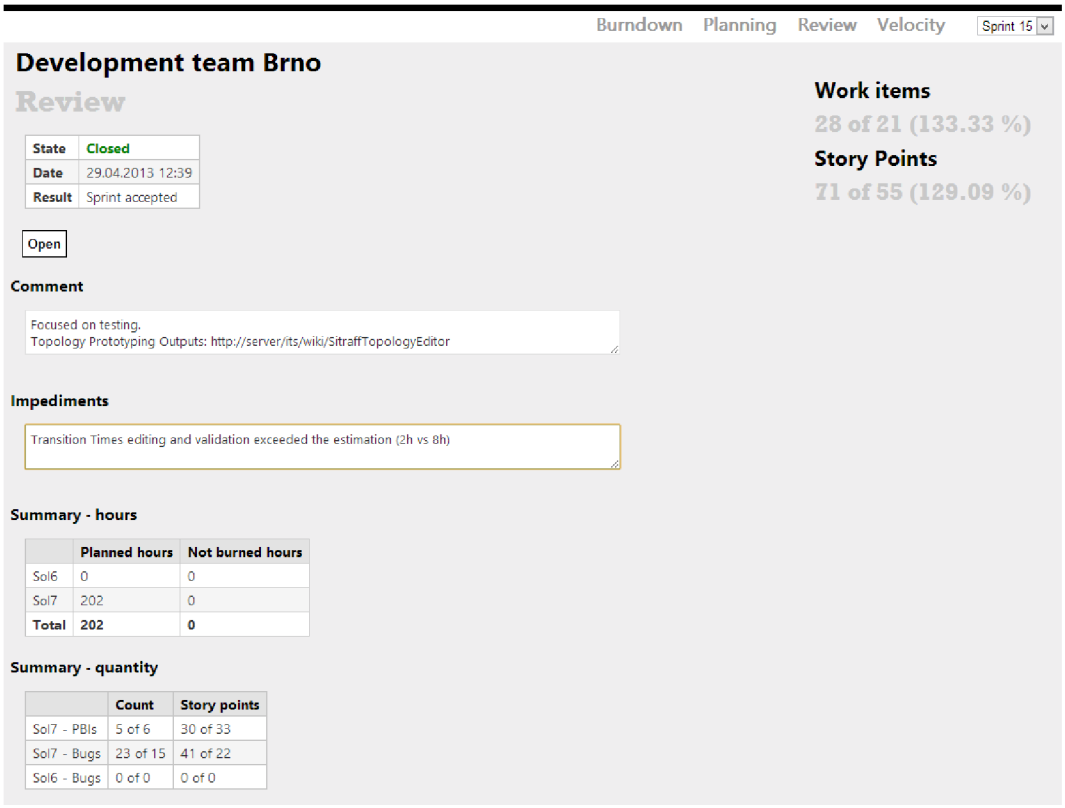
7.1.3 Revize

Uzavření již skončeného sprintu se pak provádí na stránce označené jako Revize. Tento protokol je automaticky generován na základě informací získaných z obou serverů k aktuálnímu datu a času, v případě, že je zvolen již dříve uzavřený Sprint, vážou se tato data ke dni a hodině jeho uzavření. Možné vyplnění protokolu je vyobrazeno na obrázku 7.11.

Člen týmu provádějící Revizi Sprintu má možnost připsat k protokolu vlastní komentář nebo zde poznačit všechny Překážky, které se v průběhu objevily. Jedná o jediné dva uživatelské vstupy na této stránce, ostatní informace jsou agregovány zcela automaticky.

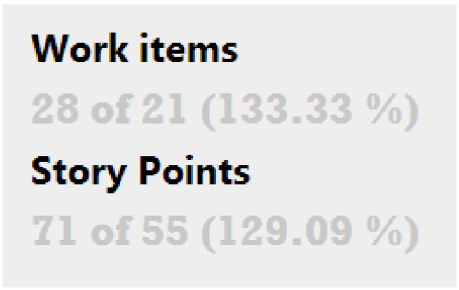
Zadávání komentářů a Překážek je dostupné i v průběhu Sprintu a je ukládáno do databáze.

Součástí protokolu jsou informace o zbývajících hodinách a sumarizace počtů položek pro jednotlivé servery tak, aby měl tým dostatečnou informaci o tom, co se mu ve Sprintu podařilo či nepodařilo. Tento protokol slouží jako vstup pro Retrospektivu, kdy dochází k hodnocení skončeného Sprintu.



Obrázek 7.11: Příklad vyplnění protokolu Revize Sprintu.

V pravé horní části se pak nachází rychlá sumarizace klíčových informací z revize Sprintu, jak lze vidět na obrázku 7.12. Jako první je zde uveden celkový počet dokončených položek ze Sprint katalogu vůči původně naplánovaným, souhrně pro Defekty i Uživatelské scénáře. Rovněž lze zde najít Výkonnost týmu pro tento Sprint opět ve vazbě na původně naplánovanou práci. Nechybí ani procentuální vyjádření udávající úspěšnost či neúspěšnost týmu.

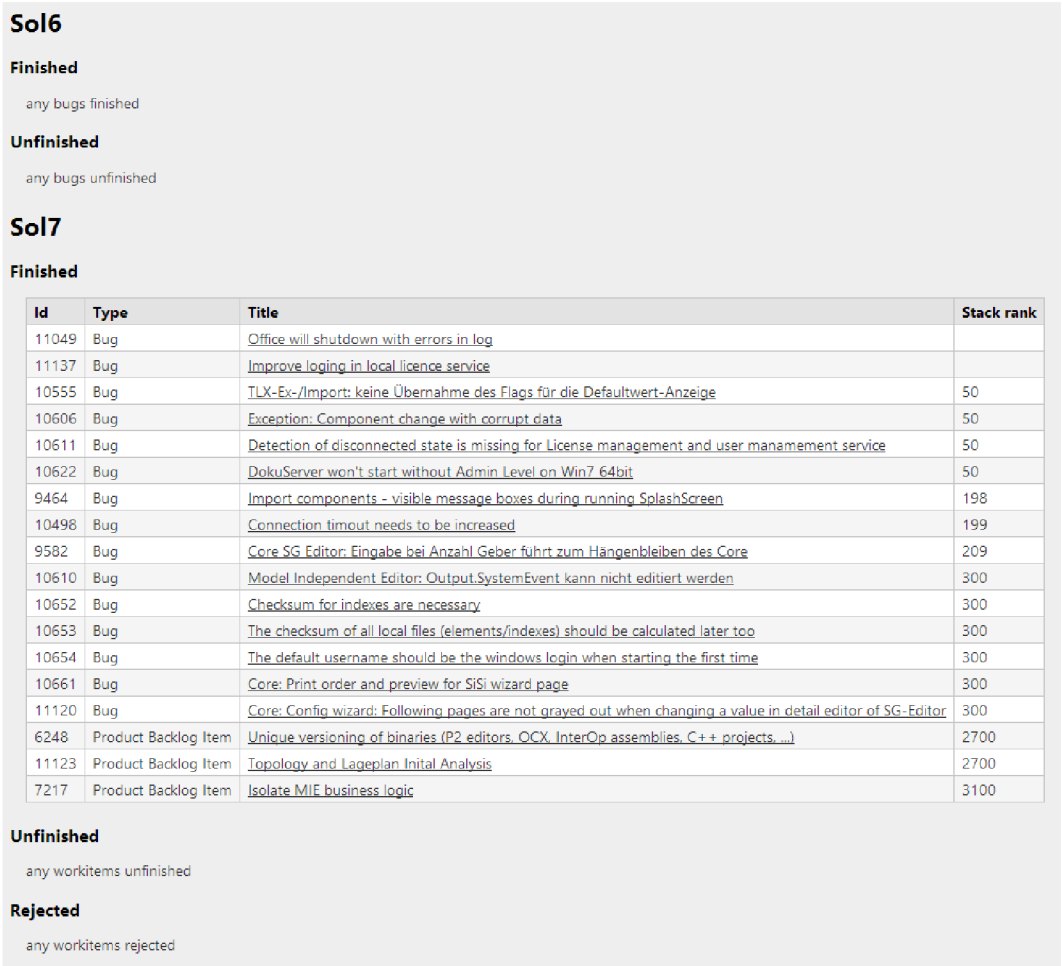


Obrázek 7.12: Vizualizace počtu dokončených položek.

Ve spodní části stránky se pak nachází sumarizace položek ze Sprint katalogu rozlišená

podle toho, zda se je podařilo dokončit, nedokončit nebo zda byly odmítnuty zákazníkem. Toto shrnutí výsledku Sprintu lze vidět na obrázku 7.13. U každé položky z katalogu lze kliknutím na její titulek zobrazit detailní informace na příslušném serveru.

Stejně jako u Plánování mají členové týmu možnost uzavřít protokol z Revize pomocí tlačítka `Close` v horní části stránky. Pokud je potřeba informace dodatečně upravit, lze tuto akci provést teprve po stisku tlačítka `Open`, čímž dojde ke znovuotevření dokumentu.



Sol6

Finished

any bugs finished

Unfinished

any bugs unfinished

Sol7

Finished

Id	Type	Title	Stack rank
11049	Bug	Office will shutdown with errors in log	
11137	Bug	Improve loging in local licence service	
10555	Bug	TLX-Ex-/Import: keine Übernahme des Flags für die Defaultwert-Anzeige	50
10606	Bug	Exception: Component change with corrupt data	50
10611	Bug	Detection of disconnected state is missing for License management and user manamement service	50
10622	Bug	DokuServer won't start without Admin Level on Win7 64bit	50
9464	Bug	Import components - visible message boxes during running SplashScreen	198
10498	Bug	Connection timout needs to be increased	199
9582	Bug	Core SG Editor: Eingabe bei Anzahl Geber führt zum Hängenbleiben des Core	209
10610	Bug	Model Independent Editor: Output.SystemEvent kann nicht editiert werden	300
10652	Bug	Checksum for indexes are necessary	300
10653	Bug	The checksum of all local files (elements/indexes) should be calculated later too	300
10654	Bug	The default username should be the windows login when starting the first time	300
10661	Bug	Core: Print order and preview for SiSi wizard page	300
11120	Bug	Core: Config wizard: Following pages are not grayed out when changing a value in detail editor of SG-Editor	300
6248	Product Backlog Item	Unique versioning of binaries (P2 editors, OCX, InterOp assemblies, C++ projects, ...)	2700
11123	Product Backlog Item	Topology and Lageplan Inital Analysis	2700
7217	Product Backlog Item	Isolate MIE business logic	3100

Unfinished

any workitems unfinished

Rejected

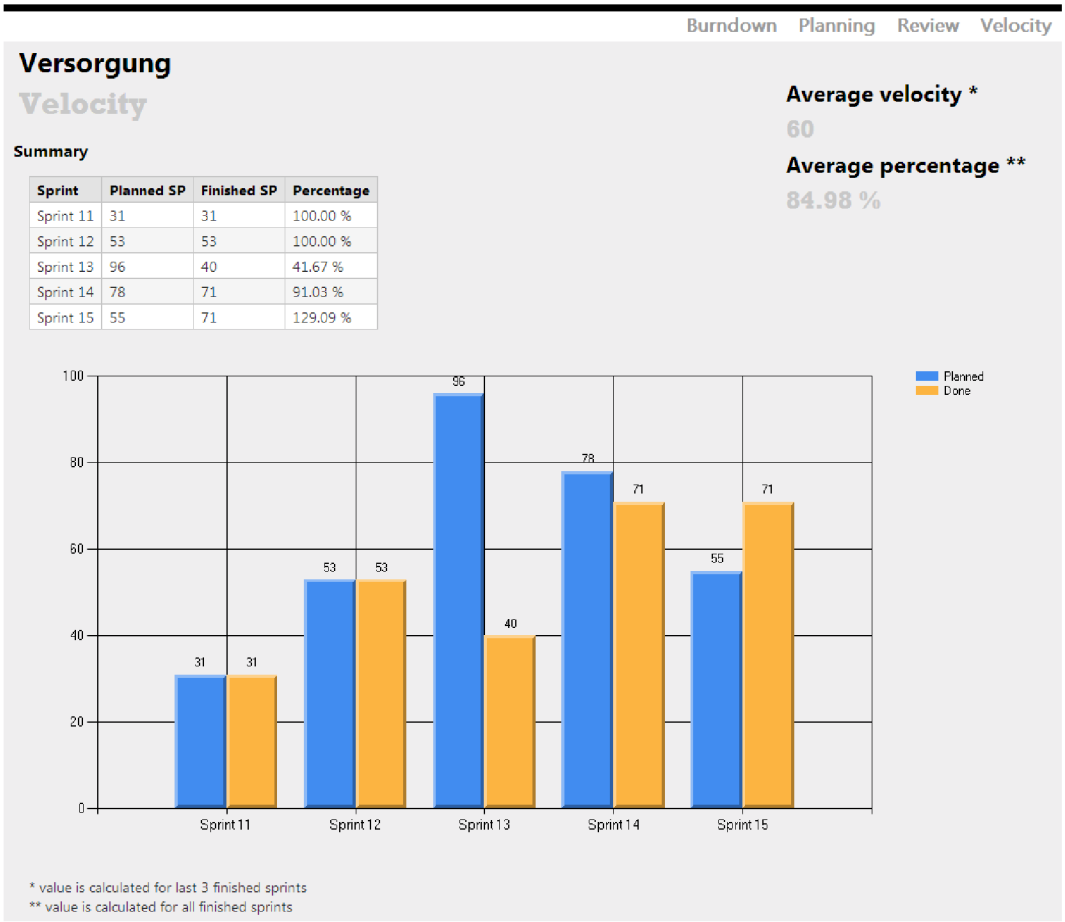
any workitems rejected

Obrázek 7.13: Seznam dokončených, nedokončených a odmítnutých položek.

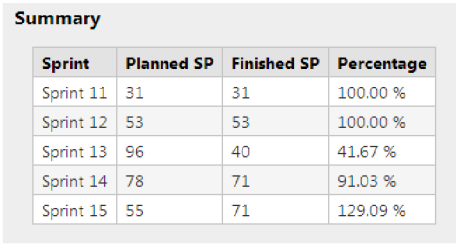
7.1.4 Výkonnost

Jak již bylo uvedeno dříve, obrazovka zobrazující Výkonnost týmu neumožňuje volbu konkrétního Sprintu, neboť zobrazuje informace o všech uzavřených Sprintech najednou. Stránka opět agreguje informace z obou serverů a slouží jako podklad pro sledování Výkonnosti týmu v dlouhodobém horizontu. Tuto část lze použít například pro plánování uvolnění další verze aplikace Vlastníkem produktu, kdy se pro stanovení dalších milníku vývoje používá právě aktuální Výkonnost týmu.

Pro každý Sprint je zde uveden v tabulce počet naplánovaných Ohodnocení položek, počet skutečně dokončených Ohodnocení a procentuální úspěšnost naplánovaných vůči skutečně splněným. Detail tabulky lze vidět na obrázku 7.15.



Obrázek 7.14: Rozložení stránky s data o Výkonnosti týmu.

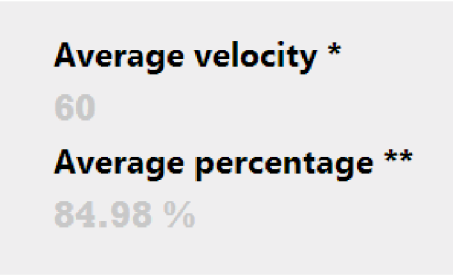


Obrázek 7.15: Detail tabulky s ohodnocením pro jednotlivé Sprinty.

V pravé části obrazovky lze pak vidět dva ukazatele označující Výkonnost týmu, které mají následující význam:

- **Aktuální výkonnost** – Hodnota je vypočítána z posledních tří uzavřených Sprintů a je průměrem dokončených položek ze Sprint katalogu, resp. součtu jejich ohodnocení.
- **Procentuální úspěšnost** – Ukazatel toho, jak se týmu daří plnit plán v souhrnu všech Sprintů, vypočítáno jako průměr úspěšností všech Sprintů.

Příklad reálných hodnot se nachází na obrázku 7.16. Obě vyobrazené hodnoty slouží týmu jako podklad pro Plánování. Podle aktuální Výkonnosti může již při Plánování na



Obrázek 7.16: Aktuální Výkonnost a procentuální úspěšnost.

Stránka	Bez cache	S cachí	Urychlení
Burndown	122 099 302	3 160	38 639 %
Planning	43 806 707	441	99 334 %
Review	12 690 814	440	28 842 %

Tabulka 7.1: Porovnání rychlosti načítání při zapnutém a vypnutém cachování.

základě Ohodnocení plánovaných položek vidět, zda je reálné naplánovanou práci splnit nebo nikoliv. Druhý ukazatel pak týmu říká, u kolika procent z naplánovaných položek hrozí, že nemusí být úspěšně dokončeny.

7.2 Cachování

Jelikož aplikace pracuje s velkým množstvím dat, která jsou agregována ze dvou dedikovaných serverů, bylo nutné pro rychlou odezvu aplikace implementovat cachování u historických dat, které se v čase již nemění. Při prvním použití aplikace umožňuje uložit historická data. V dalším průběhu používání jsou data cachována na základě událostí uživatele. V zásadě jsou implementovány dva způsoby cachování:

1. **Graf zbývající práce** – Při zobrazení aplikace komunikuje s databází a snaží se získat data z již skončených dní aktuálního Sprintu. Pokud nejsou dostupná v databázi, dochází k jejich agregaci s obou serverů a jejich uložení do databáze. Pokud jsou uložena v databázi, aplikace se již serverů vícekrát nedotazuje. Výjimku tvoří aktuální den, který je vždy získáván z aktuálních dat a není cachován.

2. **Plánování a Revize** – Do chvíle, kdy protokol není uzavřen uživatelem, jsou data agregována ze serverů pro aktuální datum, pokud však dojde k uzavření protokolu je uložen do databáze aktuální stav a v budoucnu se již používají data uložená v databázi.

Díky implementaci cachování došlo k průměrnému zrychlení načítání aplikace o 55 604 %. Měření bylo provedeno postupně na všech stránkách vyjma Výkonnosti. Měření může být zatíženo chybou, která může být způsobena rychlostí sítě uvnitř společnosti, nicméně rozdíl je natolik významný, že by případná chyba výsledek zřejmě neovlivnila. Naměřené hodnoty rychlost načítání při zapnutém i vypnutém cachování uvádí tabulka 7.1.

Pro zobrazení jednotlivých položek ze Sprint katalogu není používáno cachování, neboť se jedná o velice rychlou operaci. Podrobný popis výkonnostního testu lze nalézt v projektu `WebBurndown.Tests.PerformanceTest`.

7.3 Testování

Aplikace byla od počátku vývoje nasazena v reálném prostředí společnosti, konkrétně na projektu, pro něhož byla vyvíjena. Správnost implementace byla otestována především automatizovaně prostřednictvím sady automatizovaných testů. Proběhlo také manuální testování pro předem definované scenáře. V rámci vývoje byly prováděny následující typy testů:

- Testy jednotek (Unit tests)
- Zahořovací testy (Smoke tests)
- Regresní testy (Regression tests)

Kromě výše uvedených testů byly také prováděny tzv. Opičí testy (Monkey tests), které měly za cíl uvést aplikaci do nekonzistentního stavu. Díky těmto testům byly odstraněny výjimky a neošetřené stavy, na které se zapomnělo při vývoji aplikace. Všechny typy testů byly prováděny od počátku implementace aplikace a velice napomohly k rychlému odstranění nalezených problémů.

7.3.1 Testy jednotek

Hlavním sadou testů aplikace jsou automatizované testy jednotek, které pokrývají hlavní funkcionalitu aplikace. Celkem soubor obsahuje 17 testů, které pokrývají všechny stránky aplikace. Díky těmto testům bylo dosaženo otestování kódu ze 42 % (ukazatel označovaný jako Code Coverage byl zjištěn pomocí zkušební verze nástroje dotCover[1] společnosti JetBrains). Testy jsou zaměřeny nejen na automatickou agregaci dat z obou serverů, ale také na kalkukace spojené s uživatelským vstupem ve fází plánování. Výčet testů jednotek s krátkým popisem se nachází dále, Podrobnější popis chování jednotlivých testů lze nalézt ve zdrojovém kódu v projektu `WebBurndown.Tests`.

Graf zbývající práce (`BurndownTest.cs`)

- `SprintStatus` – Správné zobrazení stavu Sprintu.
- `DaysToTheEnd` – Algoritmus pro výpočet počtu zbývajících dnů.
- `BurndownCache` – Správná činnost cache (načtení, ukládání).
- `LastUpdate` – Zobrazení poslední aktualizace členy týmu.

Plánování (`PlanningProtocolTest.cs`)

- `PlanningSummary` – Generování plánovacího protokolu.
- `WorkitemsSummary` – Správné zobrazení naplánovaných položek.
- `UnassignedPath` – Funcionalita nesprávně přiřazených položek.
- `ActivitiesHours` – Algoritmus pro výpočet hodin pro jednotlivé aktivity.
- `SprintTotalHours` – Výpočet celkového souhrnu dostupných hodin.

[1] http://www.jetbrains.com/dotcover/

- `SprintMemberHoursWithDayOff` – Výpočet nedostupnosti členů týmu.
- `SprintWorkdays` – Algoritmus pro výpočet pracovních dní pro daný Sprint.

Revize (`ReviewProtocolTest.cs`)

- `ReviewSummary` – Generování protokolu Revize.
- `FinishedWorkitems` – Správné zobrazení dokončených položek.
- `UnfinishedWorkitems` – Správné zobrazení nedokončených položek.
- `RejectedWorkitems` – Správné zobrazení odmítnutých položek.

Výkonnost (`VelocityTest.cs`)

- `SummaryTable` – Generování sumarizační tabulky pro jednotlivé Sprinty.
- `AverageValues` – Správné zobrazení průměrné hodnoty.

7.3.2 Zahořovací testy

Tento typ testů se zaměřuje pouze na hlavní funkce programu, které nebývají příliš často upravovány. Lze v podstatě říci, že tyto testy jsou prováděny každodenně při používání aplikace, kdy je členy týmy zobrazován Graf zbývající práce, neboť již tato činnost ověří, zda je aplikace na serveru dostupná a zda funguje správně. Pro tuto kategorii byly definovány tyto testy:

1. Zobrazení Grafu zbývající práce pro aktuální Sprint.
2. Zobrazení záložky Review pro aktuální Sprint.
3. Zobrazení záložky Review a kliknutí na dokončené položky.
4. Zobrazení záložky Planning pro aktuální Sprint.
5. Zobrazení záložky Velocity pro aktuální Sprint.
6. Výběr různých Sprintů z nabídky a správné přegenerování aktuální stránky.

7.3.3 Regresní testy

Regresní testy mají za úkol ověřit funkcionalitu aplikace z pohledu provedení předem definovaného scénáře a zjistit, zda je možné při současném stavu scenář kompletně realizovat.

U tohoto typu testů se nabízí možnost automatického testování grafického uživatelského rozhraní, nicméně jsem se rozhodl tyto testy provádět manuálně, neboť úsilí vložené do jejich implementace by nebylo přímo úměrné velikosti projektu a benefitům, které by to do budoucna přineslo.

Scénář č. 1

Kontroluje funkcionalitu na počátku nového Sprintu, pokrývá otestování uživatelských vstupů při plánování, správné uzavření protokolu z plánování a zobrazení Grafu zbývající práce.

1. Zadání URL aplikace do prohlížeče – zobrazí se aktuální Sprint.
2. Sprint se musí nacházet ve stavu Planning a nesmí být zobrazen Graf zbývající práce
3. Kliknutí na záložku Planning.
4. Přidání komentáře a cílů, kliknutí na tlačítko `Save`.
5. Přidání nedostupnosti členům týmu a přegenerování tabulky.
6. Úprava rozdělení projektových aktivit a přegenerování tabulky.
7. Uzavření plánování kliknutím na tlačítko `Close`, zobrazení stavu `Closed` a data uzavření.
8. Přechod na záložku Burndown, musí být zobrazen graf a stav `Development`.
9. Při přepnutí na stránku Planning zůstaly vyplněné informace, stav `Closed` a je zakázána editace.

Scénář č. 2

Kontroluje funkcionalitu aplikace používanou na konci Sprintu, kdy skončil vývoj a Sprint je před Revizí. Má za cíl otestovat korektní uzavření Sprintu a zobrazení Výkonnosti.

1. Zadání URL aplikace do prohlížeče – zobrazí se Graf zbývající práce pro aktuální Sprint se stavem `Reviewed`.
2. Přechod na záložku Review, protokol je ve stavu `Opened`.
3. Vyplnění komentáře a Překážek, uložení pomocí tlačítka `Save`.
4. Přechod na záložku Velocity – aktuálně revidovaný Sprint není zobrazen.
5. Přechod zpět na stránku Review – uzavření protokolu pomocí tlačítka `Close`.
6. Zobrazení záložky Burndown – Sprint je ve stavu `Closed`, Graf zbývající práce je zobrazen.
7. Při přepnutí na stránku Review zůstaly vyplněné informace, stav `Closed` a je zakázána editace.
8. Záložka Velocity ukazuje nově právě uzavřený Sprint.

Kapitola 8

Zhodnocení a další vývoj

Kapitola se zabývá zhodnocením implementovaných řešení ve vazbě na úroveň kvality procesů uvnitř zkoumané společnosti. Rovněž zde lze nalézt výsledky interního auditu, který objektivně hodnotí realizovaná vylepšení stávajících procesů. Závěrem kapitoly jsou nastíněny oblasti možného budoucího vývoje aplikace a také rozvoje úrovně kvality po stránce modelu CMMI.

8.1 Zhodnocení implementace

V rámci implementace aplikace se podařilo realizovat veškerou funkcionalitu, která byla naplánována na počátku vývoje. Vzhledem k tomu, že průběžně byly nové součásti nasazeny do ostrého provozu na běžícím projektu, bylo možné všechny funkce a zobrazení vyladit tak, aby odpovídaly skutečným potřebám organizace. Během vývoje se rovněž objevila další potencionální vylepšení, která vyplynula z užívání aplikace a byla implementována nad rámec plánované funkcionality. Z těchto části lze například zmínit zobrazení položek z katalogu s nesprávně přiřazenou cestou nebo počet dní do konce Sprintu.

Za velký přínos aplikace považuji skutečnost, že protokoly z Plánování a Revize Sprintu již tým nemusí vyplňovat ručně, ale jsou generovány automaticky a odpovědný člen týmu pouze doplňuje drobné komentáře či další vstupy. Proces Plánování se podařilo z velké části automatizovat, především výpočet pracovních dnů a dostupnosti členů týmu. Dříve byla právě tato oblast velmi náchylná k chybám, které byly způsobeny ruční kalkulací. Dnes je na základě začátku a konce Sprintu dopočítáván celkový počet pracovních dní, v tomto počtu nejsou zahrnuty svátky ani firemní volna. Každý člen týmu si zadá svou nedostupnost v daném Sprintu a na základě poměru přiřazení k jednotlivým aktivitám je vypočten celkový souhrn dostupných hodin pro jednotlivé projektové aktivity.

Zde bych rád podoktnul, že rozdělení objemu práce mezi více aktivit u jednotlivých členů týmů disponuje funkcionalitou, která není podporována ani v nové verzi Team Foundation Server 2012 od firmy Microsoft. Jejich implementace umožňuje přiřazení konkrétního člověka pouze k jedné aktivitě, což přímo odporuje principům agilního vývoje, kdy by každý člen týmu měl být schopen zastávat jakoukoliv roli. Implementovaná aplikace touto funkcí disponuje, čímž mnohem více reflektuje specifika agilního vývoje.

Vzhledem k tomu, že od počátku vývoje byla aplikace nasazena v reálném prostředí, bylo dosaženo jejího vyladění konkrétním potřebám týmu, ale také k intenzivnímu beta testování. Jelikož byla aplikace ihned používána, byly všechny nalezené chyby či nedostatky průběžně odstraněny. Ke kvalitě aplikace přispěly také implementované Testy jednotek, díky kterým

bylo dosaženo automatické otestovatelnosti aplikace v rozsahu 42 %.

Na závěr bych rád zdůraznil, že jsem velice rád, že se podařilo implementovat aplikaci, která je již nyní plně využívána v prostředí reálné firmy a poskytuje rozsáhlou podporu vývojovému týmu ve všech fázích agilního vývoje. Myslím si, že používáním tohoto nástroje došlo k úspoře 8 hodin práce jednoho člena týmu, který je možné investovat jiným způsobem. V neposlední řadě také díky této aplikaci došlo ke zdokonalení těch procesních oblastí, které před započetím práce zcela nesplňovaly požadavky modelu CMMI vytyčené úrovně, což bylo hlavním cílem práce.

8.2 Interní audit

Před započetím realizace této diplomové práci proběhl ve společnosti Siemens interní audit, který byl proveden osobou, která je oprávněna hodnocení stavu kvality podle modelu CMMI provádět. Z tohoto auditu vzešlo ohodnocení stavu procesů před implementací dříve uvedených zlepšení.

Po dokončení všech vylepšení procesního charakteru, implementaci a nasazení aplikace byl proveden opětovný interní audit s cílem zjistit, nakolik tato diplomová práce přispěla ke zvýšení úrovně kvality podle modelu CMMI.

Díky této diplomové práci společnost v současné době dosahuje úrovně definované modelem CMMI, kterou si stanovila za cíl získat a nemělo by tak pro ni být problémem získat certifikaci při ostrém auditu, který se uskuteční v budoucnu. Tato práce přispěla k navýšení úrovně kvality procesů o 0,26 bodu na stupnici modelu od 1 do 5. Dokladem této hodnoty je také text zhodnocení odvedené práce konzultantem, Ing. Janem Vernerem, uvedený v příloze.

8.3 Možný další vývoj

Další rozvoj úrovně kvality procesů uvnitř zkoumané společnosti za cílem rozvoje modelu CMMI by bylo možné ubírat směrem zavedení firemních standardů napříč všemi odděleními a různými projekty uvnitř organizace. Rovněž by bylo možné sjednotit historická data z dříve uskutečných projektů do globální databáze, která by poskytla nově začínajícím projektům možnost čerpat z jejich výsledků.

Pokud bychom hovořili o možných zlepšením stávajících procesů, bylo by možné u zkoumaného projektu zavést do všech etap agilního vývoje softwaru pokročilé techniky pro analýzu a podporu rozhodování.

Při zamyšlení nad budoucím vývojem aplikace se nabízí několik oblastí rozšíření. Většina z nich by mohla přinést odstranění další administrativní zátěže, které je kladena na členy týmu nebo naopak poskytnout benefit v podobě možnosti jednoduchého zobrazení, v současné době těžko získatelných informací.

8.3.1 Použití pro ostatní týmy

Již od počátku vývoje byla aplikace navržena tak, aby umožňovala rozšíření i na ostatní týmy uvnitř společnosti. Díky tomuto rozšíření by bylo možné dosáhnout zvýšení úrovně kvality i v řadě jiných oddělení, která v současné době sice neusilují o certifikaci kvality podle modelu CMMI, ale do budoucna jim tento nástroj může významně pomoci právě ve splnění požadavků definovaných modelem CMMI.

Na rozšíření o další týmy je již nyní aplikace připravena, proces by spočíval v pouhém doplnění informací do databáze aplikace, konkrétně by bylo nutné přidat nový tým a do něj přidat členy týmu.

8.3.2 Nahrazení protokolu setkání se zákazníkem

Dalším možným rozšířením, které se přímo nabízí, je zautomatizování procesu dalších událostí definovaných agilní vývojovou metodikou Scrum.

V současné době tým dvakrát do týdne vyplňuje zprávy na týmové wikipedii, které slouží jako podklad pro telekonference s druhou částí týmu na Německé straně a jedná se z velké části o nadbytečnou práci, kterou by mohlo být možné zautomatizovat. Pověřený člen týmu vždy před tímto setkáním zkontroluje Sprint katalog a zaznačí do protokolu ty položky, které byly od minulého setkání dokončeny a které jsou aktuálně rozpracovány. Součástí tohoto dokumentu je také hlášení Překážek či upozornění na nutnost rozhodnutí ze strany zákazníka. Tento dokument pak slouží jako základ pro toto setkání.

8.3.3 Podpora pro Revizi kódu

Ve společnosti je v pravidelně stanovených intervalech prováděna Revize kódu a její výsledek je zaznamenáván na týmovou wikipedii. U výsledného dokumentu však chybí potřebná vazba na konkrétní části revidovaného kódu. Proto se nabízí možnost obohacení nástroje o možnost nejen tvorby zápisu z revize, ale také o propojení s kódem přímo na serveru tak, aby bylo možné revidované úseky jednoduše dohledat. Tuto funkcionalitu by mohla poskytovat právě jako rozšíření implementovaná webová aplikace.

Otázkou však zůstává, zda má toto rozšíření aplikace v současnou chvíli smysl, neboť nová verze Team Foundation Server 2012 již podporu pro Revizi kódu obsahuje nativně, a tudíž by s přechodem na novou verzi bylo implementované rozšíření nadbytečné. Pokud by se však o přechodu neuvažovalo, přímo se tato integrace do aplikace nabízí.

8.3.4 Graf zbývající práce pro jednotlivé položky

Pro potřeby Retrospektivy právě skončeného Sprintu by bylo velmi přínosné mít možnost zobrazit Graf zbývající práce nejen pro celý Sprint, ale také pro jednotlivé položky ze Sprint katalogu. Díky tomu by bylo možné velice efektivně a především objektivně zhodnotit důvod nesplnění dané položky a přijmout taková opatření, která by podobným situacím v budoucnu zabránila.

Graf zbývající práce na dané položce v průběhu Sprintu může například napomoci odhalit, že práce byly zahájeny příliš pozdě, ačkoliv se jednalo o Cíl Sprintu nebo naopak by z něj bylo možné vidět, že se v průběhu Sprintu nedařilo uzavírat položky ze Sprint katalogu tak úspěšně jak bylo plánováno.

Kapitola 9

Závěr

Cílem této diplomové práce bylo zanalyzovat stav procesů ve společnosti Siemens, nastudovat požadavky modelu kvality CMMI a konfrontovat tyto požadavky s již používanou agilní metodikou Scrum. Pro nalezené nedokonalosti se podařilo najít odpovídající vylepšení stávajících procesů, která napomohla společnosti k získání požadované úrovně kvality podle modelu CMMI, což bylo hlavním cílem práce.

Během důkladné analýzy bylo zjištěno, že pro získání požadované úrovně CMMI je nutné získat ze dvou serverů další informace. Bylo proto v součinnosti s konzultantem rozhodnuto, že bude vytvořen nástroj, webová aplikace, který tyto servery spojí přehledně na jednom místě a bude vizualizovat chybějící data.

Vytvořená webová aplikace přinesla týmu potřebnou podporu ve všech fázích agilního vývoje. Díky tomu, že aplikace byla od samotného počátku vývoj nasazena na běžícím projektu, se podařilo splnit nejen požadavky modelu CMMI, ale také konkrétní potřeby projektového týmu. Veškerá implementovaná funkcionalita poskytuje týmu potřebnou podporu v celé fázi vývoje produktu.

Za další benefit považuji fakt, že aplikaci je možné do budoucna bez problému rozšířit na ostatní projektové týmy, což napomůže ke zvýšení úrovně kvality oddělení jako celku. Do aplikace by bylo možné do budoucna zaintegrovat další používané dokumenty a zautomatizovat jejich vytváření tak, aby byli členové týmu co nejvíce odstíněni od nadbytečné administrativy a mohli se soustředit na vývoj.

Efekt zavedených vylepšení byl zhodnocen nezávislým interním auditem certifikovanou osobou uvnitř společnosti Siemens. Výsledkem tohoto auditu bylo zjištění, že realizací zlepšení došlo k navýšení úrovně vyspělosti o 0,26 bodu, díky čemuž společnost aktuálně dosahuje požadované úrovně Vyspělosti modelu CMMI a je tak připravena na ostrý audit, který se uskuteční v budoucnu.

Na závěr je také říci, že práce byla v měsíci dubnu prezentována a úspěšně obhájena na studentské konferenci EEICT, na které se v konkurenci ostatních účastníků a jejich příspěvků umístila na druhém místě v kategorii magisterských projektů oboru Informační systémy, což považuji za velký úspěch a ocenění mnou odvedené práce.

Literatura

[1] BECK, e. a., K.: Manifest Agilního vývoje software [online]. 2001, [cit. 2012-10-22].
URL http://agilemanifesto.org/iso/cs/principles.html

[2] BOREK, B.: Úvod do architektury MVC [online]. 2009, [cit. 2013-05-02].
URL http://www.zdrojak.cz/clanky/uvod-do-architektury-mvc/

[3] CARNEGIEMELLON: *Capability Maturity Model Integration* [online]. Carnegie Mellon University, August 2002, [cit. 2012-11-03].
URL http://www.sei.cmu.edu/reports/02tr029.pdf

[4] CHRISSIS, M. B.: *CMMI for development: guidelines for process integration and product improvement.* Addison-Wesley, třetí vydání, 2011, ISBN 978-0-321-71150-2, 650 s.

[5] COHN, M.: *Agile Estimating and Planning.* Prentice Hall, 2007, ISBN 0-1314-7941-5.

[6] GILB, T.: *COMPETITIVE ENGINEERING: A Handbook for Systems Engineering, Requirements Engineering and Software Engineering.* Middlesex University, 2005, ISBN 0-7506-6507-6.

[7] JAMES, M.: Scrum Training Series [online]. [cit. 2012-12-22].
URL http://scrumtrainingseries.com/

[8] JOY, A. A.: 5 Quick Steps to Get Introduced with Visual Studio Team System and Team Foundation Server 2010 [online]. 2009, [cit. 2013-04-27].
URL http://weblogs.asp.net/ashraful/archive/2009/06/03/5-quick-steps-to-get-introduced-with-visual-studio-team-system_and-team-foundation-server-2010-beta-1.aspx

[9] KIŠOŇOVÁ, E.; MIARKA, R.: agileSEM: an agile System Development Method at Siemens in CEE [online]. January 2012, [cit. 2013-01-01].
URL http://www.software-quality-days.com/uploads/media/1130_Eva_Kisonova_Ralph_Miarka_agileSEM_an_agile_System_Development_Method_at_Siemens_in_CEE.pdf

[10] MCCONNELL, S.: *Code Complete: A Practical Handbook of Software Construction.* O'Reilly Media, 2004, ISBN 978-0735619678.

[11] NHIBERNATE FORGE: The official new home for the NHibernate for .NET community [online]. 2012, [cit. 2013-05-11].
URL http://nhforge.org/

[12] PDQM: CMMI – FAQ (Časté otázky) [online]. 2012, [cit. 2012-11-12].
URL http://www.pdqm.cz/Blog/CMMI-FAQ.html

[13] SCHWABER, K.; SUTHERLAND, J.: *The Scrum Guide* [online]. Scrum.org, October 2011, [cit. 2013-01-01].
URL http://www.tutorialspoint.com/cmmi/

[14] SHORE, J.; WARDEN, S.: *The Art of Agile Development*. O'Reilly Media, 2008, ISBN 978-0-596-52767-9.

[15] SZALVAY, L.: Scrum Impediments [online]. [cit. 2012-01-02].
URL http://scrummethodology.com/scrum-impediments/

[16] WIEGERS, K.: *Peer Reviews in Software: A Practical Guide*. Addison-Wesley, 2001, ISBN 978-0201734850.

Seznam použitých zkratek

CAR	Analýza příčin a řešení
CM	Konfigurační řízení
CMMI	Stupňovitý model vyspělosti
DT	Vývojový tým
IPM	Integrované projektové řízení
MA	Analýza a metriky
OPD	Firemní definice procesů
OPF	Firemní procesní zaměření
OPM	Řízení výkonu v organizaci
OPP	Řizení procesního výkonu v organizaci
OT	Firemní vzdělávání
PB	Produktový katalog
PBI	Položka produktového katalogu
PI	Integrace produktu
PMC	Monitorování a kontrola projektu
PO	Vlastník produktu
PP	Projektové plánování
PPQA	Zajišťování kvality produktu a procesů
QPM	Kvantitativní projektové řízení
RAD	Rozhodovací analýza a řešení
RD	Požadavky na vývoj
REQM	Řízení požadavků
RSKM	Řízení rizik
SAM	Řízení subdodavatelů
SB	Sprint katalog
SM	Mistr Scrumu
TS	Technické řešení
VAL	Validace
VER	Verifikace

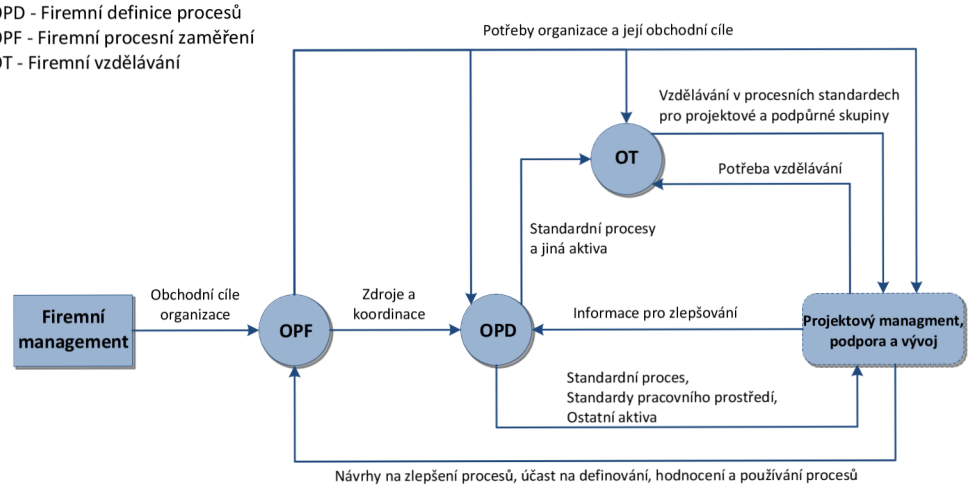
Příloha A

Provázání procesních oblastí CMMI

A.1 Procesní řízení

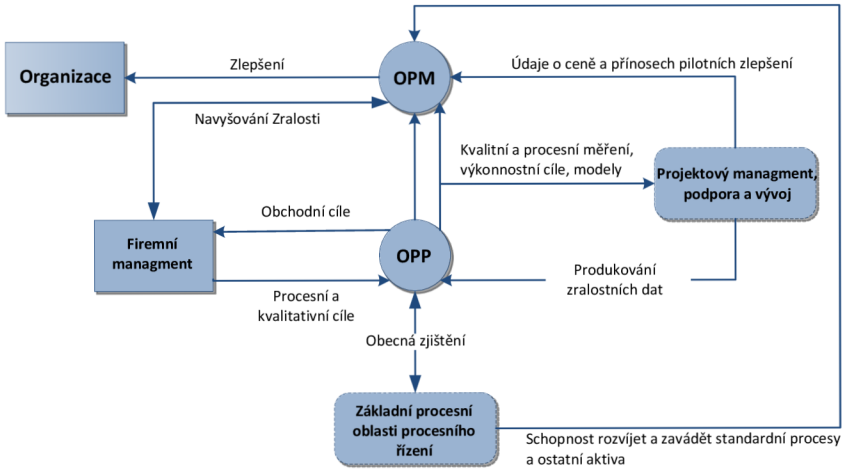
Základní procesní oblasti

OPD - Firemní definice procesů
OPF - Firemní procesní zaměření
OT - Firemní vzdělávání



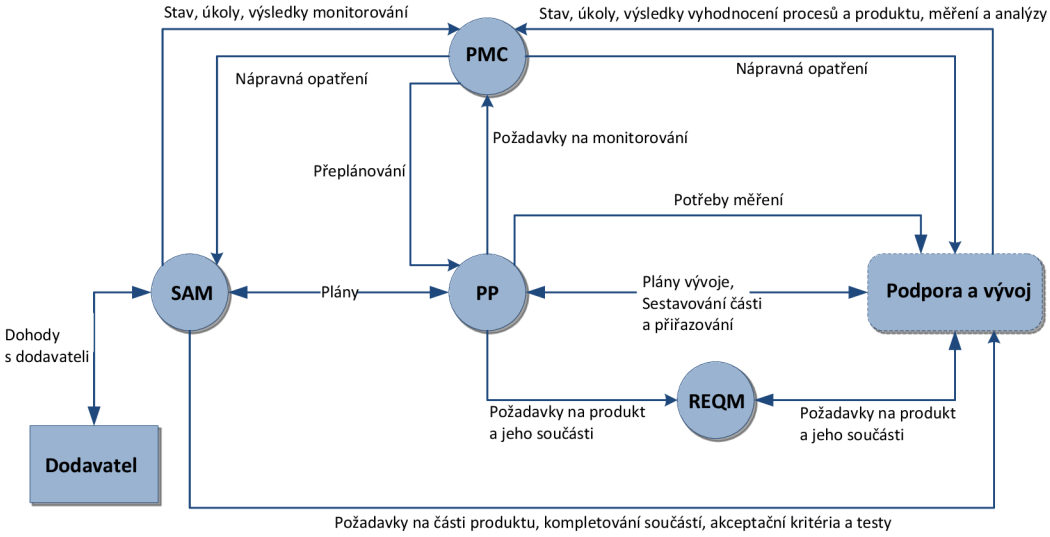
Pokročilé procesní oblasti

OPM - Řízení výkonu v organizaci
OPP - Řízení procesního výkonu v organizaci

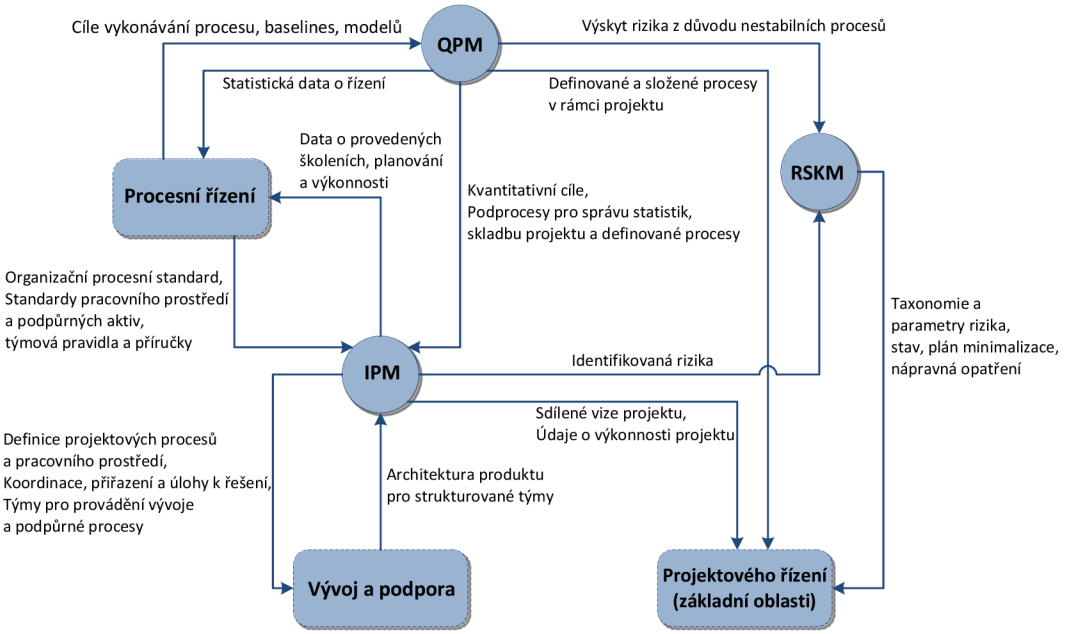


A.2 Projektové řízení

Základní procesní oblasti
PMC - Monitorování a kontrola projektu
PP - Projektové plánování
REQM - Řízení požadavků
SAM - Řízení subdodavatelů

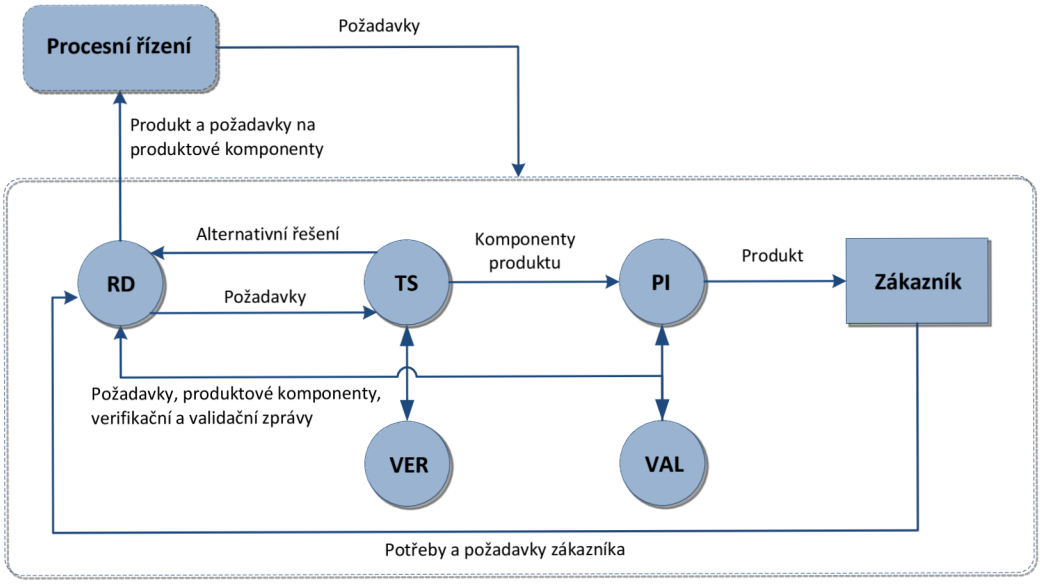


Pokročilé procesní oblasti
IPM - Integrované projektové řízení
QPM - Kvantitativní projektové řízení
RSKM - Řízení rizik



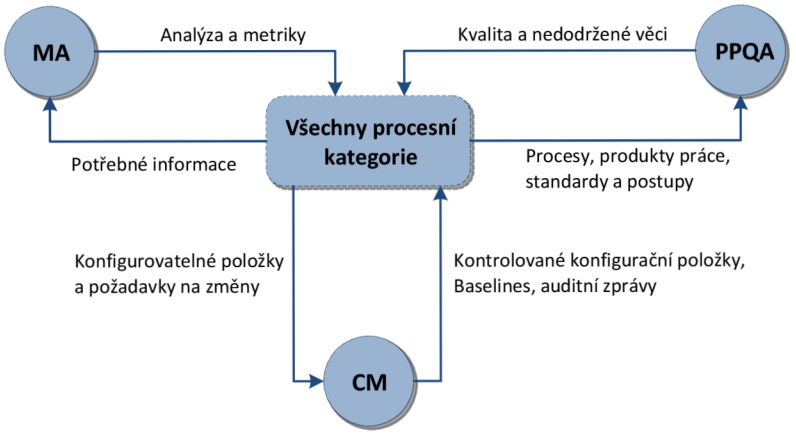
A.3 Vývoj

Procesní oblasti
PI - Integrace produktu
RD - Požadavky na vývoj
TS - Technické řešení
VAL - Validace
VER - Verifikace

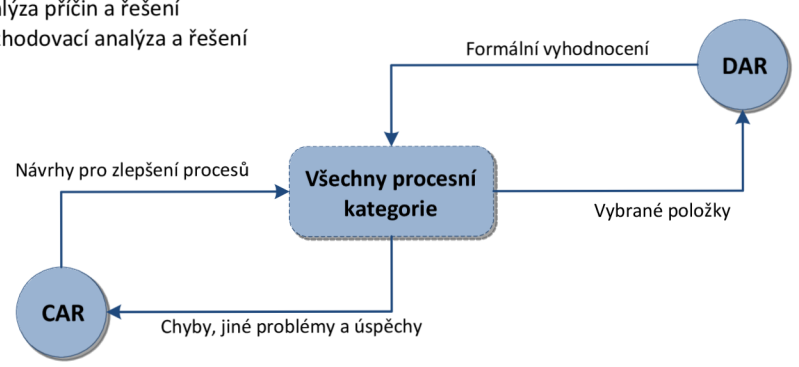


A.4 Podpora

Základní procesní oblasti
CM - Konfigurační řízení
MA - Analýza a metriky
PPQA - Zajišťování kvality produktu a procesů



Pokročilé procesní oblasti
CAR - Analýza příčin a řešení
DAR - Rozhodovací analýza a řešení



Příloha B

Obsah CD

Elektronická verze textu diplomové práce ve formátu PDF se nachází v kořenovém adresáři přiloženého CD. Na tomto médiu se dále nachází tři adresáře, jejichž obsah je následující:

Adresář `doc`

- `doc/html` – Programová dokumentace k aplikaci ve formátu HTML.
- `doc/pdf` – Programová dokumentace k aplikaci ve formátu PDF.

Adresář `src`

- `src/app` – Zdrojové soubory aplikace psané v jazyce C#.
- `src/doc` – Zdrojové soubory programové dokumentace psané v jazyce LaTeX.
- `src/thesis` – Zdrojové soubory textu diplomové práce psané v jazyce LaTeX.

Adresář `demo`

V tomto adresáři se nachází demoverze webové aplikace, která nevyžaduje připojení na servery v podnikové síti.

SIEMENS

Za společnost Siemens Jan Verner, konzultant.

Společnost Siemens se celosvětově dlouhodobě zaměřuje na neustálé zvyšování kvality procesů. Naše oddělení Corporate Technology v Brně, ve kterém pan Gajdušek realizoval svoji diplomovou práci, pomáhá plnit tento cíl mimo jiné také realizací požadavků normy CMMI, která zasahuje všechny procesní oblasti a má velmi vysoké nároky. Vývoj produktů v našem softwarovém oddělení je již velmi dlouho postaven na principech agilních metodik, které ovšem nejsou zcela vždy kompatibilní s normou CMMI. Nalezení takového řešení, které by splňovalo požadavky obou, do jisté míry protichůdných přístupů, bylo výzvou této diplomové práce.

Na úvodním setkání byly stanoveny následující cíle:

- Prostudovat požadavky normy CMMI a agilních vývojových metodik.
- Seznámit se detailně s výsledky interních auditů a předložit analýzu aktualního stavu.
- Na základě analýz navrhnout a realizovat takové technické řešení, které bude vyhovovat požadavkům obou norem a bude splňovat dodatečná kritéria stanovená konzultantem.

Musím ocenit pana Gajduška za velmi dobrou spolupráci a zodpovědný přístup při realizaci jeho diplomové práce. Cením si zejména proaktivního přístupu ve všech fázích projektu, častých konzultacích, které byly vždy velmi konstruktivní a bylo z nich zřejmé, že se v dané problematice velmi dobře orientuje.

Během první fáze realizace se pan Gajdušek seznámil velmi rychle s procesy a požadavky a byl schopen navrhnout první řešení, které také po konzultaci naimplementoval v podobě webové aplikace. Fakt, že již během této rané fáze projektu byla k dispozici testovací verze jeho nástroje pro podporu procesů, hodnotím jako velmi pozitivní. Je nutné také zmínit, že takto brzká realizace zcela předčila původně stanovený časový harmonogram a výrazně přispěla k vysoké kvalitě výsledné diplomové práce.

Interní audit provedený před i po úpravách realizovaných v rámci práce pana Gajduška ukazuje zlepšení ukazatelů hodnot CMMI o +0,26 na stupnici od 1 do 5. Tento výsledek lze považovat za nadstandardní. Osobně pak hodnotím míru splnění předem stanovených cílů na 130%. Výsledek práce ve všech ohledech splňuje požadavky společnosti, oddělení i produktových týmů a navíc přináší nové inovativní přístupy. Práce byla také úspěšně obhájena na soutěži vysokoškolských prací EEICT, což jen dokládá její kvalitu.

Vzhledem k náročnosti zvoleného tématu, které vyžaduje velmi dobré teoretické základy společně s technickými schopnostmi nutnými pro realizaci aplikace, zodpovědnému přístupu a vynikajícím výsledkům, které převyšují původně stanovené cíle, považuji diplomovou práci pana Gajduška za výbornou. Jeho diplomová práce výraznou měrou přispěla při realizaci náročných požadavků normy CMMI, za což bych mu rád vyjádřil i svůj osobní dík.

Jan Verner, Siemens

V Brně 16.5.2013