ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

# ZADÁNÍ BAKALÁŘSKÉ PRÁCE

| | |
|---|---|
| **Název:** | Řízení projektu a infrastruktury portálu pro podporu výuky předmětu BI-DBS |
| **Student:** | Oldřich Malec |
| **Vedoucí:** | Ing. Jiří Hunka |
| **Studijní program:** | Informatika |
| **Studijní obor:** | Softwarové inženýrství |
| **Katedra:** | Katedra softwarového inženýrství |
| **Platnost zadání:** | Do konce letního semestru 2017/18 |

## Pokyny pro vypracování

Cílem práce je revize, reorganizace a nastavení řízení projektu vývoje a infrastruktury portálu DBS pro podporu výuky databázových systémů. Realizační týmy jsou sestaveny ze studentů SP1, SP2 a studentů pracujících na BP.

1. Nastudujte a krátce popište metody řízení vývoje SW systému, kde je třeba koordinovat práci více týmů a jednotlivců.
2. Popište stávající proces řízení projektu, proveďte analýzu s cílem odhalit slabá místa, navrhněte reorganizaci procesu a realizujte ji. Pracujte v roli projektového manažera.
3. Navrhněte, realizujte a zdokumentujte infrastrukturu, která se pro vývoj i samotné nasazení použije. Soustřeďte se zejména na efektivitu vývoje a znovupoužitelnost kódu.

## Seznam odborné literatury

Dodá vedoucí práce.

L.S.

Ing. Michal Valenta, Ph.D.
vedoucí katedry

prof. Ing. Pavel Tvrdík, CSc.
děkan

V Praze dne 28. listopadu 2016

České vysoké učení technické v Praze

Fakulta informačních technologií

Katedra softwarového inženýrství

Bakalářská práce

# Řízení projektu a infrastruktury portálu pro podporu výuky předmětu BI-DBS

***Oldřich Malec***

Vedoucí práce: Ing. Jiří Hunka

24. března 2017

# Poděkování

TODO poděkování

# Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o etické přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. V souladu s ust. § 46 odst. 6 tohoto zákona tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou a veškeré jejich dokumentace (dále souhrnně jen „Dílo“), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené. Každá osoba, která využije výše uvedenou licenci, se však zavazuje udělit ke každému dílu, které vznikne (byť jen zčásti) na základě Díla, úpravou Díla, spojením Díla s jiným dílem, zařazením Díla do díla souborného či zpracováním Díla (včetně překladu), licenci alespoň ve výše uvedeném rozsahu a zároveň zpřístupnit zdrojový kód takového díla alespoň srovnatelným způsobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

V Praze dne 24. března 2017 ……

České vysoké učení technické v Praze

Fakulta informačních technologií

© 2017 Oldřich Malec. Všechna práva vyhrazena.

*Tato práce vznikla jako školní dílo na Českém vysokém učení technickém v Praze, Fakultě informačních technologií. Práce je chráněna právními předpisy a mezinárodními úmluvami o právu autorském a právech souvisejících s právem autorským. K jejímu užití, s výjimkou bezúplatných zákonných licencí, je nezbytný souhlas autora.*

## Odkaz na tuto práci

MALEC, Oldřich. *Řízení projektu a infrastruktury portálu pro podporu výuky předmětu BI-DBS*. Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2017. Dostupný také z WWW: ⟨`https://gitlab.fit.cvut.cz/malecold/bc-thesis`⟩.

# Abstrakt

TODO

**Klíčová slova** TODO

# Abstract

TODO

**Keywords** TODO

# Obsah

# Seznam tabulek

# Úvod

TODO úvod

Kapitola 1

# Co je to DBS portál

Celá tato práce se zabývá vývojem a správou portálu pro předmět BI-DBS na FIT ČVUT. Ještě než se tedy ponoříme do jednotlivých součástí vývoje tohoto portálu, chtěl bych čtenáře seznámit s portálem samotným.

## 1.1 Historie výuky BI-DBS

Předmět Databázové systémy je vyučován na katedře softwarového inženýrství a v současném studijním programu se s ním setkají studenti bakalářského programu Informatika ve svém druhém semestru. Výuka si klade za cíl přiblížit studentům problematiku ukládání dat, a to především pomocí relačních databází. Důraz je kladen na jazyky RA a SQL. Forma výuky předmětu pochází již z dob výuky na FEL ČVUT a používala se již déle než 10 let.

**Semestrální práce** Každý student si v rámci své semestrální práce vytvoří vlastní databázi, do které si poté vymýšlí dotazy za pomoci různých konstrukcí výše zmíněných jazyků. Typická semestrální práce obsahuje ve své finální podobě alespoň 25 dotazů do databáze a pokrývá všechny standardní klauzule, které jazyk SQL nabízí. Semestrální práce se vypracovávala ve formátu XML, který byl následně pomocí XSLT převeden do HTML pro přehlednější reprezentaci. Zejména studenti si často na nutnost vyplňovat XML šablonu stěžovali.

**Testy v semestru a zkouška** Kromě semestrálních prací popsaných výše jsou také součástí hodnocení studentů průběžné testy v semestru a závěrečná

zkouška. Testy v semestru se týkají především praktické části - tedy používání RA a SQL či modelování schémat databáze - které by student měl mít osvojené ze své semestrální práce. Závěrečná zkouška poté kromě praktických částí může obsahovat i teoretické otázky které byly probírány na přednášce.

## 1.2 Vznik DBS portálu

Obě výše popsané součásti výuku jsou poměrně náročné na korekturu vyučujícím. Typicky nebylo možné zajistit, aby vyučující zkontrolovatl každý jednotlivý dotaz, který student vytvořil ve své semestrální práci. Oprava testů, které se psali na papír poté trvala zbytečně dlouho a prakticky nebylo možné ji automatizovat. Z tohoto důvodu přišel v roce 2013 Jiří Hunka - jeden z vyučujících předmětu BI-DBS - s nápadem, že se realizuje portál zaměřený na podporu výuky Databázových systémů, který bude umožňovat jak efektivnější korekturu studentských semestrálních prací, tak rychlejší opravu testů a zkoušek.

**Řešitelský tým** Vývoj portálu takových rozměrů však nebylo možné financovat běžně dostupnými prostředky, kterými fakulta disponuje. Z toho důvodu bylo rozhodnuto, že bude portál vyvíjen s rámci předmětů BI-SP1 a BI-SP2. Jedná se o předměty vyučované také v oboru Softwarové inženýrství, které si studenti typicky zapisují ve svém 4., respektive 5. semestru studia. Cílem předmětů je vytvořit 3-5 členné týmy, které budou pracovat na softwarovém projektu, počínaje návrhem, analýzou požadavků atp. a konče hotovým softwarem. Bylo tak vypsáno zadání na realizaci DBS portálu, do kterého se přihlásilo 13 studentů. Tento způsob získávání pracovní síly pro další vývoj portálu je používán dodnes. Jelikož SP1 a navazující SP2 má celkové trvání jeden akademický rok, jsou každý rok nabíráni noví studenti. Toto přináší jak obtíže s řízením projektu, tak příležitosti pro jeho změny. Řízení DBS projektu se věnuje kapitola 3.

**Nasazení portálu** Portál byl poprvé nasazen do výuky v LS 2016, kdy byl využíván pouze na cvičeních, která vedl Jiří Hunka a ve zkouškovém období byl otestován také na jednom termínu závěrečné zkoušky. V ZS 2016 portál používali všichni studenti, kteří měli předmět BI-DBS zapsaný, protože jich v tomto semestru bylo pouze 36. Jednalo se tak o ideální testovací vzorek pro příští semestr. Následující semestr - LS 2017 - již byl portál využíván

všemi cvičícími a počet studentů se pohyboval kolem 550. Detailům ohledně nasazování portálu se věnuje kapitola 4.

Kapitola 2

# Metody řízení projektu

V druhé kapitole bych se chtěl věnovat především tématu metodik vývoje softwaru. Cílem je krátce čtenáře seznámit se standardními a agilními metodami vývoje softwaru a zaměřit se na možnosti jejich aplikace pro vývoj DBS portálu, který byl popisován v kapitole 1

## 2.1 Tradiční metody

### 2.1.1 Vodopádový model

Jak zmiňuje [4], nejedná se přímo o metodiku, ale pouze o životní cyklus. Je charakteristický tím, že všechny fáze vývoje jsou prováděny postupně za sebou a není možné se vracet. Případné změny je možní zpracovat až v rámci údržby, která vývoj vždy vrátí do jedné z předchozích etap a následně musí proběhnout i etapy následující.

- Definice problému
- Specifikace požadavků
- Návrh
- Implementace
- Integrace, testování
- Údržba

Tento model vznikl již v roce 1970 a pro vývoj softwaru, který lze typicky nasadit ještě před jeho kompletním dokončením a poté iterativně dále rozvíjet, je nevhodný.

### 2.1.2 Spirálový model

Vzniku Spirálového modelu dala za vznik především kritika Vodopádového modelu. Hlavní novinkou zde byl *iterativní přístup* a opakovaná *analýza rizik*. V tomto modelu se stále opakují fáze *Stanovení cílů*, *Analýza rizik*, *Vývoj a testování* a *Plánování*. Při každém průchodu těmito fázemi se provádí odlišná činnost než v předchozí iteraci (kromě analýzy rizik, ta je prováděna vždy stejně), například fáze vývoje v první iteraci pracuje pouze s koncepty, v příští iteraci specifikuje požadavky, v další navrhuje architekturu a teprve poté implementuje a testuje.
Stále se však jedná převážně o jednosměrnou cestu, která nejen na počátku ale i v průběhu obsahuje velké množství analýz a návrhů. Pro potřeby vývoje DBS portálu, kde se každý rok mění složení studentů, kteří projekt realizují, je tedy nevhodný.

TODO more std methods?

## 2.2 Agilní metody

Agilní metody řízení vznikly především z důvodu neustále se měnících požadavků na software a rychlejší reakci na požadované změny. Tradiční metody nebyly schopné reagovat na změny požadavků dostatečně rychle. Především když se začneme bavit o vývoji softwaru, kde se neustále mění dostupné technologie a požadavky na finální verzi, jsou tradiční metody zcela nevhodným modelem vývoje.

Rozdíl mezi tradiční a agilní metodikou lze popsat také jejich přístupem k *Funkcionalitě*, *Času* a *Zdrojům* [4]. Zatímco u tradičních metod je funkcionalita stanovena na začátku a musí být dosažena pomocí *nějakého* času a *nějakého* množství zdrojů, u agilních metod je to přesně naopak. Typicky je stanoven termín a dostupné zdroje a funkcionalita *nějaká* bude.

Připomeňme si také *Manifesto for Agile Software Development* [1], který vznikl v roce 2001:

> Objevujeme lepší způsoby vývoje software tím, že jej tvoříme a pomáháme při jeho tvorbě ostatním. Při této práci jsme dospěli k těmto hodnotám:
>
> - *Jednotlivci a interakce* před procesy a nástroji
> - *Fungující software* před vyčerpávající dokumentací
> - *Spolupráce se zákazníkem* před vyjednáváním o smlouvě
> - *Reagování na změny* před dodržováním plánu
>
> Jakkoliv jsou body napravo hodnotné, bodů nalevo si ceníme více.
>
> | | | |
> |---|---|---|
> | Kent Beck | James Grenning | Robert C. Martin |
> | Mike Beedle | Jim Highsmith | Steve Mellor |
> | Arie van Bennekum | Andrew Hunt | Ken Schwaber |
> | Alistair Cockburn | Ron Jeffries | Jeff Sutherland |
> | Ward Cunningham | Jon Kern | Dave Thomas |
> | Martin Fowler | Brian Marick | |
>
> © 2001, výše zmínění autoři Toto prohlášení může být volně kopírováno v jakékoli formě, ale pouze v plném rozsahu včetně této poznámky.

Je tedy zřejmé, že Agilní metodiky více pracují s vývojáři samotnými, a umožňuji i potlačení některých postupů pro dobro celku a dovolují ”zanedbat” i náležitosti jako například dokumentaci. Následně se na některé agilní metodiky zaměříme více:

### 2.2.1 Extrémní programování

[1]

[1]Někteří autoři (Wysocki [9]) oddělují XP od agilních metod, pravděpodobně proto, že vzniklo o 2 roky dříve než *Agile manifesto* [1] z roku 2001. Jelikož však XP postavilo základ pro agilní metodiky, zahrnu jej v tomto textu, stejně jako například Kadlec [4] do této kapitoly.

XP rozvíjí standardní postupy, avšak posouvá je až do extrémů:

- Vždy budeme pracovat na co nejjednodušší verzi, která splňuje požadavky
- Pokud se osvědčuje *revize kódu*, budeme neustále revidovat kód …
- Pokud se odvědčil *návrh*, budeme neustále navrhovat a vylepšovat specifikaci …
- Pokud se osvědčilo *testování*, budeme neustále testovat …
- Narozdíl od tradičních metodik, které najednou dodají velký balík nových funkcionalit, bude XP dodávat i ty nejmenší funkční kousky. Toto vede na *krátké iterace.*

Tyto postupy přivedené do extrémů jsou reprezentovány v *12 základních postupech XP*, se kterými se seznámíme níže.
Ještě než se dostaneme k samotným postupům, potřebujeme se seznámit s pojmem *karta zadání.*

**Karty zadání a karty úkolů** Karty zadání vznikají během plánování návrhu softwaru. Tohoto plánování se účastní všichni vývojáři, vedení a i zákazník. Typicky jedna karta popisuje jednu funkcionalitu či jeden požadavek. Kartě náleží vlastnosti jako *datum vytvoření*, *stav*, *priorita*, *odhad vyřešení* a také samozřejmě vlastní zadání úkolu a dodatečné poznámky. Když se poté na jedné kartě pracuje, je aktualizována průběžnými poznámkami, které popisují které části již byly splněny, co je potřeba udělat atp.
Kromě karet zadání existují v XP také *karty úkolů.* Tyto vznikají nikoliv na začátku celého procesu vývoje, ale na začátku jednotlivé iterace.

**12 základních postupů XP**

1. *Plánovací hra:* Probíhá jednak na začátku vývoje v rámci návrhu celé aplikace, tak na začátku jednotlivých iterací. Jejím výstupem jsou *Karty zadání* a *Karty úkolů*, popisované v předchozím odstavci.
2. *Malé verze:* Verze jsou dodávány neustále. I nejmenší změna funkcionality, která přináší užitek pro zákazníka je vydána jako nová verze.

3. *Metafora:* Celý systém je popisován jednoduchým příběhem, jak má ve výsledku fungovat. Tento příběh je sdílen jak vývojáři a managementem, tak zákazníkem a případně dalšími zainteresovanými osobami. Tato metafora u XP nahrazuje tradiční popis architektury.

4. *Jednoduchý návrh:* XP navrhuje vždy pouze to, co je přímo požadováno. Žádné přidáné vlastnosti nechce do návrhu zahrnovat, snaží se udělat minimum pro to, aby byly splněny požadavky.

5. *Testování:* Během vývoje jsou neustále psány i testy pro nově vznikající komponenty. Nikdy nesmí existovat jednotka, která nemá vlastní unit test. Testy dokonce píší i sami zákazníci, kteří tvoří testy funkcionality.

6. *Refaktorizace:* Refaktorizace je změna zdrojového kódu, která ovšem nemění jeho chování. Je prováděna z důvodu zvýšení čitelnosti kódu pro ostatní programátory či jeho zefektivnění. V XP by měl programátor vždy buď pouze implementovat novou funkčnost, nebo refaktorizovat stávající kód.

7. *Párové programování:* Zdrojový kód je vždy psán dvěma programátory u jednoho počítače. Zatímco jeden píše kód, druhý sleduje postup a přemýšlí nad dalšími souvislostmi: Je tato funkce napsána správně? Zapadá do celkového konceptu? Změní tato úprava chování jiného modulu? Je potřeba napsat nový test? …

8. *Společné vlastnictví:* Zatímco některé jiné metodiky přiřazují jednotlivé třídy určitým programátorům, které za ně poté mají zodpovědnost, v XP je to právě naopak. Všichni mají zodpovědnost za celý kód a kdokoliv tak může měnit libovolnou součást systému.

9. *Nepřetržitá integrace:* Systém je sestaven a otestván vždy, jakmile je to možné, typicky tedy ihned po dokončení nové funkcionality.

10. *Čtyčicetihodinový pracovní týden:* XP klade důraz i na spokojenost lidských zdrojů. Snaží se tedy o nepřetěžování programátorů, jelikož při jejich únavě se razantně snižuje kvalita jejich výstupu.

11. *Zákazník na pracovišti:* Součastí týmu by měl být i sám zákazník, neboli expert na cílovou doménu. Jeho účelem je odpovídat na dotazy, tvořit testy či určovat priority.

12. *Standardy pro psaní zdrojového kódu:* Jelikož XP nedefinuje žádné standardní dokumenty, jsou veškeré informaci o projektu obsaženy ve zdojových kódech samotných. Proto je potřeba, aby byly rozumně strukturované, čitelné a pochopitelné.

Některé z těchto postupů jsou jistým způsobem používány i při vývoji DBS portálu, více v sekci 2.3.

### 2.2.2 Scrum

Scrum je charakterizován především pojmy *Sprint* a *Scrum meeting*. První z nich má trvání většinou jeden měsíc a opakuje se několikrát. Jeho cílem je přinést do výsledného produktu nové funkcionality. Naopak Scrum meeting je *každodenní* schůzka, na které se shrnou nově dokončené úkoly a stanoví se, na čem se bude dále pracovat. Schůzku vede tzv. *Scrum master*. Typicky probíhá ve stoje a netrvá déle než 30 minut. Dalo by se říci, že tyto každodenní schůzky jsou stěžejní aktivitou metodiky Scrum. Od těchto schůzek se také odvíjí velikost týmu: Scrum preferuje malé týmy mezi třemi až šesti členy.

Pro potřeby DBS projektu je tedy Scrum také nevhodný, především jelikož předpokládá každodenní schůzky.

### 2.2.3 Lean development

TODO ?

### 2.2.4 Feature driven development

Jak název napovídá, FDD se soustředí především na *featury*, neboli jednotlivé vlastnosti, funkce či rysy výsledného softwaru. Na počátku je vytvořen základní model softwaru, který je dále členěn na jednotlivé vlastnosti. Vývoj probíhá zpravidla v dvoutýdenních iteracích, jejichž výsledkem by měl vždy být fungující software, který je možné představit zákazníkovi.

Klíčový pojem *feature (vlastnost)* lze popsat jako samostatnou část systému, která splňuje následující parametry:

- *měritelnost*: vlastnost je jasně uchopitelná komponenta systému, kterou lze porovnat s požadavkem zákazníka
- *srozumitelnost*: programátor musí rozumět tomu, jak má vlastnost fungovat a co je jejím cílem
- *realizovatelnost*: programátor musí vědět, zda je schopen vlastnost realizovat v odpovídajícím časovém úseky (typicky iterace 2 týdny). V opačném případě je většinou potřeba vlastnost rozdělit na více menších, a na těch pracovat samostatně.

V tuto chvíli by se mohlo zdát, že FDD vůbec nedisponuje *návrhem a modelováním*. Ve skutečnosti je tomu právě naopak, společně s FDD je často zmiňován ještě *model-driven approach*. Jelikož vlastnosti jsou klíčovou stavební jednotkou FDD, je potřeba je kvalitně navrhnout a namodelovat.

Posloupnost kroků při vývoji pomocí FDD zachycuje následující seznam:

1. vytvoření celkového (globálního) modelu
2. vypracování podrobného seznamu vlastností
3. plánování podle vlastností
4. návrh podle vlastností
5. implementace

Dvě poslední se poté stále opakují, dokud existují další vlastnosti, které je potřeba pro softwaru přidávat.

Vzhledem k vývoji DBS portálu zde stojí za zmínku především fáze návrhu a implementace:

**Návrh** Hlavní programátor v této fázi vybere vlastnosti, které budou realizovány v následující iteraci. Vlastnosti volí podle jejich priority, vzájemných návazností a dlaších kritérií. Poté jsou stanoveny třídy, kterých se vlastnost týká a je předána programátorům, kteří za danou třídu zodpovídají.

Jelikož různé vlastnosti se vždy týkají různých tříd, jsou programátoři děleni do týmů dle potřeby a jednotlivé týmy se tak pro každou iteraci obměňují.
Tento nově sestavený tým má poté za úkol připravit návrh implementace vlastnosti, na jehož základě je poté naprogramována.

**Implementace** Týmy sestavené v předchozí fázi pracují na samotné funkcionalitě nové vlastnosti. Společně s programovým kódem jsou tvořeny i testy, a to především *unit testy*.
Jakmile je vlastnost dokončena, je vložena do sdíleného prostoru, u FDD nazývaného *class repository*
Hlavní programátor poté dokončené vlastnosti integruje do předchozí verze hlavní aplikace.

Typicky na projektu pracuje několik týmů a hlavní progragramátor poté pouze volí nové vlastnosti k implementaci a integruje hotové vlastnosti do výsledné aplikace.

Výše popsaný průběh práce hrubě odpovídá i organizaci vývoje DBS projektu, které je popsáno v sekci 2.3.

TODO popsat týmy podle vlastností? (Kadlec 194)

### 2.2.5 Test driven development

TODO

## 2.3 Zvolené řešení pro DBS projekt

Jak bylo zmíněno v sekcích o *Extrémní programování* a *Feature driven development*, DBS projekt je vyvíjem pomocí metody na pomezí těchto dvou metodik.
TODO

Kapitola 3

# Řízení DBS projektu

Kapitola popisuje řízení DBS projektu, jeho předešlý stav, návrhy na vylepšení a zhodnocení aplikovaných řešení. Zabývá se především efektivností řízení lidských zdrojů, komunikace a použitých technologií.

## 3.1 Řízení lidských zdrojů

Projekt je vyvíjen již od roku 2014 v rámci předmětů BI-SP1 a navazujícího BI-SP2. Někteří studenti po dokončení BI-SP2 pokračují v práci na systému v rámci své bakalářské práce. Výjimečně jsou také vypsány samostatné bakalářské či diplomové práce zaměřující se na specifickou komponentu systému. Ze studentů předmětů BI-SP1, případně BI-SP2 jsou typycky sestaveny 2-3 týmy po 4-5 členech. TODO vedoucí týmu atp.

## 3.2 Řízení komunikace

Hlavním stěžejním bodem komunikace byla vždy pravidelná týdenní schůzka, na které se sešly řešitelské týmy s vedoucím práce - Ing. Jiřím Hunkou. Kromě těchto schůzek byl pro komunikaci využíván systém projektového řízení Redmine (popsáno v 4.1.0.2). Další komunikace nebyla jednotně stanovena a probíhala tak v různých komunikačních kanálech. TODO rozvoj, Slack...

### 3.2.1 Zpětná vazba

Zpětná vazba probíhá standardně ze strany vedení projektu. Před realizací této práce tak byla veškerá zpětná vazba realizována především na pravidelných týdenních schůzkách, příležitostě také v systému probjektového řízení - Redmine. Týmy také měly stanoveny své vedoucí, kteří řídili tým interně. Vedoucí týmu na schůzce prezentuje výstupy jeho týmu a popisuje, co který člen realizoval. Týmy se samy interně hodnotí v několika iteracích rozložených do celého semestru. TODO rozvoj, bodování

## 3.3 Nástroje

Pro jakékoliv řízení projektu je potřeba využívat určitých nástrojů. Správě infrastruktury v projektovém řízení je v této práci věnována samostatná kapitola ??.

Kapitola 4

# Správa infrastruktury

DBS projekt je webová aplikace, která je dostupná na adrese `https://dbs.fit.cvut.cz/`. Aplikace běží na Nette Frameworku (PHP) a je servírována pomocí apache. TODO

## 4.1 Infrastruktura projektu před realizací práce

#### 4.1.0.1 Git

Git [8] je distribuovaný systém pro správu verzí, navržený jak pro malé tak pro velmi velké projekty, kladoucí důraz na rychlost a efektivitu. TODO více popsat git V DBS projektu byl využíván TODO

#### 4.1.0.2 Redmine

## 4.2 Analýza a návrh zlepšení infrastruktury

**Git** Jako první nedostatek byla identifikována neexistence oddělení vývojové verze od produkční. TODO

## 4.3 Realizované řešení

**Git** V Gitu je nově dodržována přísnější struktura a například k produkční verzi má přístup pouze projektový manažer a ověření vývojáři, kteří již prošli předměty SP1 a SP2.

TODO

# Závěr

TODO závěr

# Zdroje


1. BECK, Kent et al. *Manifesto for Agile Software Development* [online]. 2001 (cit. 2017-03-15). Dostupné z: `http://agilemanifesto.org/`.
2. HRONČOK, Miroslav. *Diplomová práce* [online]. 2016 (cit. 2017-03-05). Dostupné z: `https://github.com/hroncok/diplomka`.
3. HRONČOK, Miroslav. *FIT ČVUT thesis tips* [online]. 2016 (cit. 2017-03-05). Dostupné z: `https://github.com/hroncok/fit-thesis-tips`.
4. KADLEC, Václav. *Agilní programování: metodiky efektivního vývoje softwaru*. Brno: Computer Press, 2004. ISBN 80-251-0342-0.
5. LANG, Jean-Philippe. *Overview - Redmine* [online]. 2006 (cit. 2017-03-05). Dostupné z: `http://www.redmine.org`.
6. ŘEHÁČEK, Petr. *Projektové řízení podle PMI*. Praha: Ekopress, 2013. ISBN 978-80-86929-90-3.
7. SCHWALBE, Kathy. *Řízení projektů v IT: kompletní průvodce*. Brno: Computer Press, 2011. ISBN 978-80-251-2882-4.
8. SOFTWARE FREEDOM CONSERVANCY. *Git* [online] (cit. 2017-03-05). Dostupné z: `https://git-scm.com/`.
9. WYSOCKI, Robert K. *Effective project management: traditional, agile, extreme*. Indianapolis: Wiley, 2012. ISBN 978-1-118-01619-0.

Dodatek A

# Seznam použitých zkratek

| | |
|---|---|
| BI-DBS | Databázové systémy |
| BI-SP1 | Softwarový týmový projekt 1 |
| BI-SP2 | Softwarový týmový projekt 2 |
| DBS | Databázové systémy |
| FDD | Feature driven development |
| FEL | Fakulta elektrotechnická |
| FIT | Fakulta informačních technologií |
| HTML | HyperText Markup Language |
| KOS | Komponenta studium |
| LS | letní semestr |
| PMI | Project Management Institute |
| RA | Relational algebra / Relační algebra |
| SQL | Structured query language |
| TDD | Test driven development |
| XML | Extensible Markup Language |
| XP | Extrémní programování |
| XSLT | Extensible Stylesheet Language Transformations |
| ZS | zimní semestr |
| ČVUT | České vysoké učení technické |