Porovnání DAD a RUP – co je v DAD inspirováno RUPem

Semestrální práce ke kurzu 4IT421 Zlepšování procesů budování IS	
Semestr	ZS 2015/2016
Autoři	Jakub Kos, xkosj45 Jan Zaplatílek, xzapj00
Téma	Porovnání DAD a RUP – co je v DAD inspirováno RUPem
Datum odevzdání	1. 1. 2016

Abstrakt

Tato semestrální práce se popisuje metodiky vývoje informačních systému a to konkrétně metodiku Disciplined Agile Delivery, ve zkratce DAD, a metodiku Rational Unified Process, zkráceně RUP. Tyto dvě metodiky jsou v úvodu práce představeny a vysvětleny čtenářům. V následující kapitole jsou metodiky porovnány ve vzájemně souvisejících oblastech vývoje informačních systémů. Poslední kapitola je zaměřena na zodpovězení hlavní otázky této semestrální práce a to, které prvky metodiky DAD se inspirovali v metodice RUP.

Klíčová slova

metodika budování IS, agilní metodiky, Disciplined Agile Delivery, Rational Unified Process, porovnání, inspirace, hybridní metodiky, vývoj

Obsah

1. Úvod

Nárůst počtu softwarových projektů je v dnešní době stále zřetelnější. Vznikají malé a středně velké firmy zaměřující se na vývoj aplikací a systémů. Praxe prokázala, že řízení vývoje projektů není jednoduchou záležitostí. Pokud projekt není řízen alespoň trochu sofistikovaným způsobem, zpravidla nedopadne úspěšně. Snaha zajistit přijatelnou pravděpodobnost úspěchu projektu vedla k tvorbě mnoha metodik softwarového vývoje. Je ale nutné si uvědomit, že žádná metodika nemůže předem zajistit úspěch projektu. (Julinek, 2008)

Tématem této semestrální práce je porovnání dvou metodik využívaných při vývoji nejen informačních systémů, ale software obecně. Práce je zaměřena především na metodiky Discplined Agile Delivery a Rational Unified Process.

Rational Unified Process neboli RUP je zástupcem rigorozních metodik, tedy metodik, které podrobně definují procesy a činnosti při vývoji software, obsahují vysoký počet kontrolních prvků a vysokou úroveň detailu. (Buchalcevová, 2005)

Potřeba rychlejší reakce na odezvu od zákazníka a mnohdy i uživatele daného software vedla k vytváření nových metodik s rychlejším vývojem a s větším zapojením zákazníka do vývoje. Vznikly tedy agilní metodiky. Základním kamenem agilních metodik je iterativní vývoj založený na častém dodávání verzí zákazníkovi a reagování na jeho požadavky. (Buchalcevová, 2005)

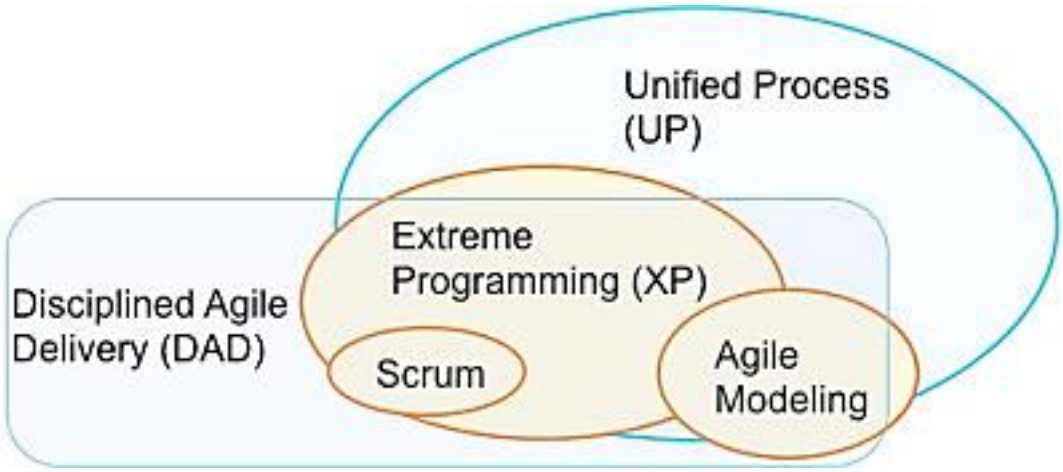
K vývoji metodiky Disciplined Agile Delivery vedly počátky usazování agilních metodik ve větších organizacích. Ty chtěli vést své vývojové týmy s použitím agilní metodiky Scrum, avšak bylo potřeba vést organizovaně i oblasti, které metodika Scrum neřídí. Týmy odpovědné za metodiky začali nahlížet do ostatních metodik a začali je kombinovat. To postupně vedlo až k vytvoření poptávky po takové metodice, která by zajistila agilní vývoj avšak i nadále řídila další oblasti vývoje software. Tuto poptávku následně uspokojila metodika Disciplined Agile Delivery. (DAD, 2015)

2. Disciplined Agile Delivery

2.1. Obecně o DAD

Jak již bylo zmíněno v úvodu, tak důvodem pro vznik Disciplined Agile Delivery byla především absence metodiky, kterou by se dal zodpovědně řídit agilní vývoj v početnějších týmech a větších organizacích. U těchto větších celků s sebou nese neúspěch větší následky. „Dosavadní převahu malých týmů v agilních postupech dokazuje i průzkum “Agile Practices Survey“ z roku 2009, podle kterého je v 68% agilních týmů pouze 1-10 IT profesionálů.“ (Koudelka, a další, 2015)

DAD je tedy relativně mladá metodika pro řízení vývoje IS v celém jeho rozsahu. Autorem metodiky je Scott W. Ambler, který ji začal vyvíjet již v roce 2007. (DAD, 2015) Disciplined Agile Delivery je popisována jako hybridní metodika, protože vychází z tradičních agilních metodik Scrum, Agilní modelování, Extrémní programování a dalších. Ze všech těchto metodik DAD vybírá a kombinuje nejosvědčenější postupy a prvky. Nad rámec součástí z agilních metodik přidává popis celého životního cyklu dodávky IT řešení, který je inspirován životním cyklem metodiky RUP. (Koudelka, a další, 2015) Pro snadnější představu začlenění jednotlivých metodik do DAD slouží Obrázek 1 - Hybridní DAD.



Obrázek 1 - Hybridní DAD

2.2. Principy DAD

Tato kapitola představuje klíčové principy DAD tak jak je popsal (Koudelka, a další, 2015). Tyto principy společně utváří charakteristiku metodiky jako takové.

LIDÉ

Klíčovým faktorem pro úspěch každého IT řešení jsou považovány lidské zdroje, jejich kvalita, a to jak jsou v rámci řešení organizovány a využívány. Kvalitou rozumíme souhrn vlastností osob zapojených do vývoje IT řešení. Vlastnosti správného člena týmu by měli odpovídat vlastnostem DAD týmu jako takového. Měl by tedy být (Koudelka, a další, 2015):

- Disciplinovaný – dělat jen to co dokáže dokončit a činit tak nejefektivněji

Organizovaný – provádět správné odhady o svých úkolech a spolupracovat s ostatními členy

- Uvědomělý – znát své přednosti i nedostatky, a pracovat na jejich odstranění

Každý člen týmu by nadále měl být bohatý na komplexní znalosti. Být specialistou na jeden obor je v agilním týmu spíše omezení než výhoda, v rychlém vývoji, kdy se jednotlivé fáze vývoje často prolínají, je nutné mít přehled i o ostatních oborech.

Obecná znalost všech částí vývoje zmenšuje riziko nedorozumění při předávkách mezi jednotlivými částmi. Zvyšuje se tak efektivita plynulost vývoje. Dále se snaha o komplexní dovednosti projevuje i v rozdělení týmových rolí, kde se řadoví členové týmu nedělí podle specializace.

„Na předávky se lze dívat také jako na potenciální riziko vzniku nedorozumění, a to právě v případě existence dokumentace. Do ní totiž prakticky nelze promítnout tacitní znalosti, které jsou pro porozumění často klíčové, a jež je možné daleko lépe sdílet osobně uvnitř týmu.“ (Koudelka, a další, 2015)

Týmové role definuje DAD tyto:

- Stakeholder – ovlivňován vyvíjeným řešením, nemusí to být přímo uživatel software
- Product owner – zastupuje uživatele, vysvětluje a upřesňuje požadavky na SW během vývoje
- Team lead – zodpovídá za vývoj, organizuje a vede tým
- Team member – člen týmu, podílející se na vývoji
- Architecture owner- dohlíží na architekturu vyvíjeného řešení

CÍL = ŘEŠENÍ

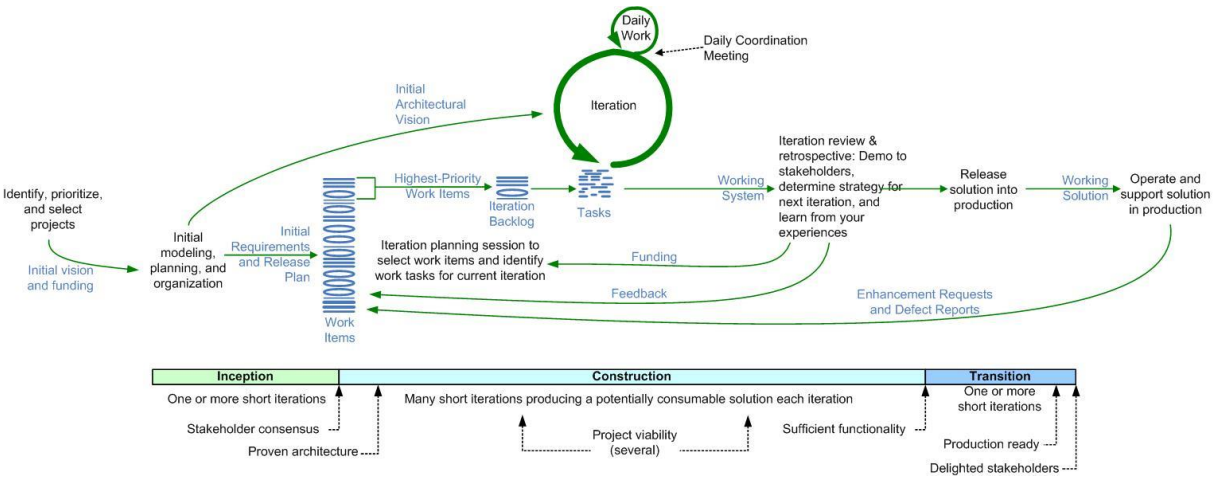
Cíl metody DAD se od ostatních liší tím, že se zaměřuje na kompletní řešení daného problému. Snaží se dodat řešení takové, které nejlépe uspokojí potřeby zákazníka. Nesoustředí se tedy pouze na vyvíjený software, ale zaobírá se kompletním řešením vyvíjeného software. Nahlíží tedy i na hardware a podniky v procesu, a upravuje je tak aby odpovídali vyvíjenému software. „DAD tímto upozorňuje na zcela zásadní poznatek, že IT profesionálové se věnují mnohem více činnostem než jen vývoji softwaru, a zdůrazňuje komplexnost jejich práce.“ (Koudelka, a další, 2015)

2.3. Fáze vývoje v DAD

Hlavně fázemi vývoje se metodika Disciplined Agile Delivery vyčleňuje od ostatních agilních metodik. V metodice DAD jsou popsány i fáze počáteční a koncová. Tradiční agilní metodiky se věnují pouze konstrukční fázi. DAD se tak snaží být střední cestou mezi tradičními a agilními metodikami.

Definované fáze životního cyklu v DAD:

- **Počáteční fáze** (Inception Phase)
- **Konstrukční fáze** (Construction Phase)
- **Přechodová fáze** (Transition Phase)



Obrázek 2 - Životní cyklus DAD

Vývoj v metodice DAD je řízen a zaměřen na cíle, proto jsou v každé fázi navrhovány určité cíle, které je třeba splnit. Metodika navrhuje různé postupy řešení jednotlivých cílů a pomocí jejích plnění sleduje průběh vývoje. (Koudelka, a další, 2015)
Vybrané cíle v jednotlivých fázích:

Tabulka 1 - Cíle v životním cyklu DAD (cs.wikipedia.org)

Fáze zahájení	**Fáze výstavby**	**Přechodová fáze**
Zformovat tým Vytvořit společnou vizi projektu Přizpůsobit firemním potřebám Prozkoumat počáteční rozsah Stanovit počáteční strategii Určit první Release Plan Vytvořit si pracovní prostředí Zabezpečit financování Identifikovat rizika	Vytvářet přijatelná/použitelná řešení Plnit měnící se potřeby zainteresovaných stran Přibližovat se k použitelné provozní verzi Zlepšovat jakost Včas stabilizovat architekturu	Zajistit, že řešení je použitelné. Nasadit hotové řešení.

3. Rational Unified Proces

3.1. Obecně o RUP

Rational Unified Proces (RUP) je rigorózní metodika pro vývoj IT projektů. Vznikla v roce 1998. RUP je metodika pro vývoj projektů na zelené louce. Tedy pro vývoj nových. RUP se zabývá pouze vývojem projektu, nikoliv jeho provozem a následnou údržbou. Vzhledem k obsáhlosti, detailnosti a taky ceně metodiky je vhodná hlavně pro velké projekty. (Buchalcevová 2005) Metodika dělí vývoj projektu a definuje jednotlivé fáze a disciplíny. Každá disciplína se skládá z jednotlivých rolí, aktivit a artefaktů. (Kroll a Kruchten 2003) Metodika RUP je postavena na následujících 6 nejlepších praktikách (best practiceses) pro vývoj softwaru:

- **Iterativní vývoj** – neboli rozdělení vývoje projektu na několik menších částí. Každí iterace by měla mít jako výstup samostatně spustitelný produkt. Hlavní výhody iterativního vývoje jsou: např. větší šance odhalení rizik v počátečních fázích projektu, snadnější implementace změn, znovu použitelnost, pomáhá učit se z minulých chyb. (Štědroň 2007)

- **Řízení požadavků** – Řízení, někdy také označováno jako správa, požadavků je systematický přístup k přijímání, organizování, dokumentování a sledování požadavků na vyvíjený software. Mezi hlavní výhody používání řízení požadavků patří: lepší kontrola komplexních projektů, lepší kvalita SW a vyšší spokojenost zákazníka, snížení nákladů a času vývoje projektu a lepší komunikace v rámci týmu.(Kruchten 2004)

- **Použití komponentové architektury** - neboli rozdělení vyvíjeného SW na jednotlivé komponenty. Komponent můžeme definovat jako nějakou netriviální kus softwaru, má jasně danou funkci, jasné hranice a lze ho snadno spojit s ostatními komponenty. (Kruchten 2004) Využití komponentové architektury pomáhá lépe zvládnout a vytvořit vnitřní integritu a komplexnost projektu. (Štědroň 2007)

- **Vizuální modelování –** zachycení návrhu a požadavků na vyvíjený software pomocí grafických modelů. Nejčastěji je využívána notace UML. Výhodou tohoto principu je jasné a přehledné popsání modelem vykreslené problematiky komukoli kdo ovládá UML. (Štědroň 2007)

- **Kontrola kvality SW –** průběžní monitorování a kontrole kvality vyvíjeného softwaru. Základní snahou je odhalení chyb, ověření oproti požadavkům a funkčnosti vyvíjeného SW. Cílem je odhalit případné chyby a nekvalitní části co nejdříve. (Štědroň 2007)
- **Řízení změn-** nějaký definovaný způsob řízení a zapracování změnových požadavků. Pomáhá určit dopady změn na projekt a zabraňuje přílišnému a nekontrolovatelnému rozšiřování projektu. (Štědroň 2007)

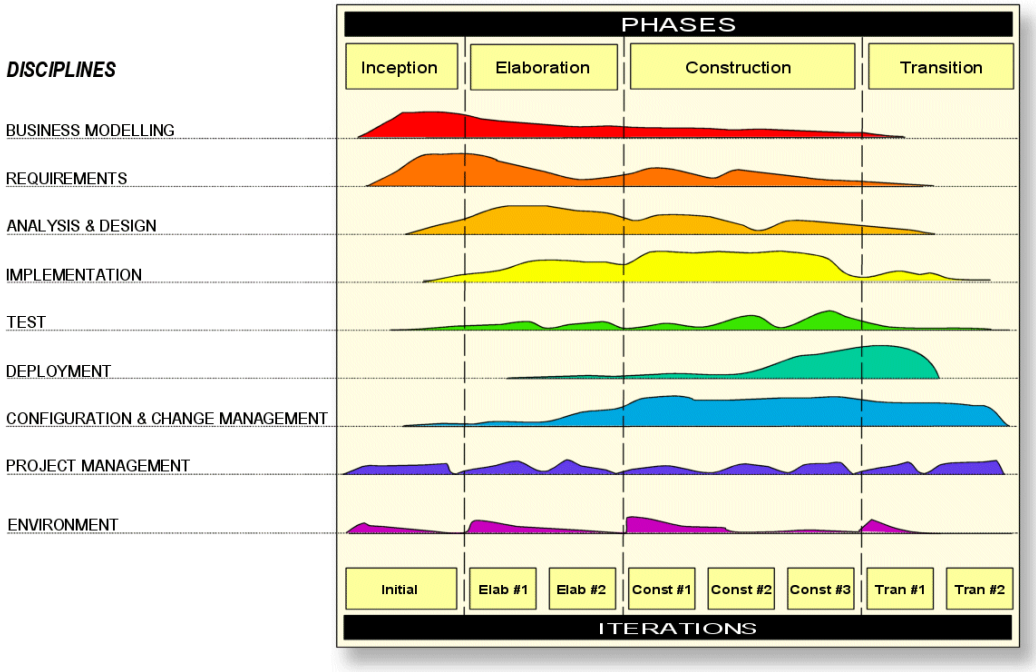
V současnosti je metodika RUP vlastněna a dále vyvíjena IBM. Rovněž existuje odlehčená verze pro menší projekty.

3.2. Životní cyklus vývoje

Na životní cyklus metodiky RUP je možno nahlížet ze dvou dimenzí:

1. Dynamický pohled – Dynamický pohled (v obr.1 představuje vodorovnou osu) vyjadřuje a popisuje jednotlivé fáze, cykly, iterace a milníky při vývoji. (Kroll a Kruchten 2003)

2. Statický pohled – Na obr. 1 vertikální osa. Tento pohled reprezentuje jednotlivé procesní elementy (aktivity, role, artefakty) sdruženy do jednotlivých disciplín.

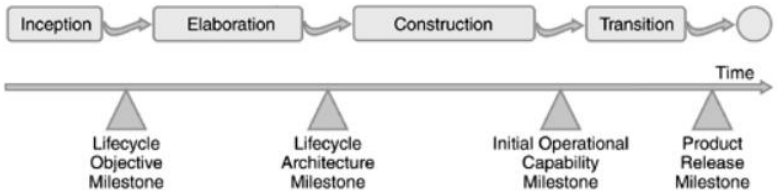


Obrázek 3 Rozdělení projektu na jednotlivé fáze, iterace a disciplíny v metodice RUP (Staffordshire University Enterprises Ltd. 2008)

Obr 1 dále zobrazuje důležitost a míru provádění jednotlivých disciplín v daných iteracích a fázích projektu.

3.3. Fáze

Životní cyklus metodiky rup se dělí, na čtyři fáze. Každá fáze je zakončena milníkem (milestone) tedy časovým bodem do kterého je potřeba splnit metodikou dané požadavky. (Kroll a Kruchten 2003) Obrázek 2 zobrazuje tok jednotlivých fázi a jejich milníky v čase.



Obrázek 4 Fáze RUP a jejich milníky v čase (Kroll a Kruchten 2003)

1. Inception – Cílem počáteční fáze je pochopení a definování co vlastně máme vyvíjet, co je konečným cílem. Měli by se definovat požadavků, cílů, rizik projektu. Dále by se měl vytvořit časový plán projektu a odhad nákladů. Tato fáze by měla končit rozhodnutím, zda je projekt možno za daných podmínek realizovat (Buchalcevová 2005)
2. Elaboration – Elaborační fáze se zabývá vytvářením základní architektury projektu, na které se bude dále stavět v konstrukční fázi. (Buchalcevová 2005)
3. Construction – Konstrukční fáze se zabývá samotným implementací projektu a jeho testováním. Výstupem této části je konečný software (Buchalcevová 2005)
4. Transition – poslední fáze se zabývá nasazením. Neboli přípravě a zavádění projektu do provozu. DO této části patří instalace řešení, školení, předání dokumentace, help-desku atd.. (Buchalcevová 2005)

3.4. Disciplíny

Metodika rup obsahuju 9 disciplín. Každá disciplína je logickým seskupením jednotlivých elementů metodiky RUP (aktivit, rolí a artefaktů). Které se mají v průběhu vývoje provést. (Kroll a Kruchten 2003) Míru využívání jednotlivých disciplín v průběhu vývoje zobrazuje obr. 1.

Disciplíny v metodice RUP můžeme rozdělit na dvě skupiny. První skupina popisuje 6 hlavních disciplín:

1. Business modelling (obchodní modelování)
2. Requirements (požadavky)
3. Analysis and design (analýza a návrh)
4. Implementation (implementace)
5. Test (Testování)
6. Deployment (Zavádění)

Druhá skupina jsou tři podpůrné disciplíny:

1. Configuration and change management (Konfigurační management)
2. Project management (Řízení projektu)
3. Enviroment (Prostředí)

Názvy jednotlivých disciplín jsou jasné a není potřeba je detailněji představovat.

3.5. Role, Artefakty a Aktivity

Role

Role v metodice Rup představuje souhrn činností, povinností a odpovědností, které má představitel role v týmu. Není však dané že jedna osoba musí představovat pouze jednu roli. Velmi často má jeden člověk více rolí a jednu konkrétní roli vykonává více osob. (Kroll a Kruchten 2003) Rolí má metodika RUP definováno více než 30. Např**. Business Process Analyst, Systems Analyst, Implementer, Tester, Tech Writer, Course Developer a další.** (Crain 2005)

Artefakty

Artefakt v metodice RUP představuje nějakou informaci, kterou role používá, nebo vytváří při vykonávaní dané aktivity. Artefaktem může být: Model (Use case, design model), element modelu, dokument, zdrojový kód, samostatně spustitelná část projektu (např. prototyp) a další (Kroll a Kruchten 2003)

Aktivity

Aktivita v metodice RUP představuje jasně danou činnost, kterou má vykonat role. Aktivita má jasně daný účel a cíl. Velmi častým cílem aktivity je vytvořit nebo upravit nějaký artefakt.

4. Porovnání DAD a RUP

Tato část práce se zabývá porovnáním dvou metodik pro vývoj software, které jsou popsány v předchozích částech. Tedy o metodiku Rational Unified Proces a metodiku Disciplined Agile Delivery. Cílem této části je ukázat čtenáři přímo rozdíly mezi oběma metodikami a také ukázat v jaké oblasti se nejvíce metodika DAD inspiruje a čerpá z metodiky RUP. Porovnání je rozděleno na tři části. První část porovnává obecné principy metodik, jejich zaměření, dostupnost a další. Druhá část porovnává životní cyklus projektu a jeho rozdělení a poslední část této kapitoly podrobněji rozebírá v čem se DAD inspiruje metodikou RUP.

4.1. Obecně

Hlavní rozdíl mezi porovnávanými metodikami je znát na první pohled. RUP je rigorózní metodikou. Tedy má jasně definované procesy, činnosti, aktivity. DAD je metodikou agilní. JE více volnější, nemá tak podrobně popsané své části a dává lidem větší svobodu. Jelikož ovšem jedním z důvodů pro vznik DAD byla přílišná volnost agilních metodik (především Scrumu a Extrémního programování), které v podstatě popisovali pouze metodiku práce a nebyly efektivní pro řízení a práci ve větších týmech a neobsahovaly finální předání produktu.(DAD 2015) Z tohoto důvodu není DAD tak volný pružný jako ostatní agilní metodiky S předcházejícího textu je jasné ž DAD je novější metodikou, byl vytvořen v roce 2012, o čtrnáct let později co vznikla metodika RUP.

Obě metodiky mají společné to, že se zaměřují pouze na vývoj nového softwaru až do okamžiku předání. Ani jedna z porovnávaných metodik se nezabývá následným provozem a údržbou softwaru. Rozdíl je u typu projektů, které se vyvíjí těmito metodikami. Zatímco RUP je pouze pro velké projekty, důležité a komplexní (díky své detailnosti, počtu rolí, artefaktů, aktivit atd.).(Kroll a Kruchten 2003) DAD lze použít jak na velké, tak i malé projekty. To je možní díky jeho klíčové vlastnosti, škálovatelnosti. Metodika DAD se dá přizpůsobit jednotlivým vyvíjeným projektům, ať podle velikosti, důležitosti, komplexnosti, náročnosti atd. (DAD 2015).

Konečně posledním porovnávaným parametrem v této části je dostupnost obou metodik. Ani jedna z metodik není volně dostupná. RUP je vlastněn IBM a za jeho použití se musí zaplatit. Po zaplacení zákazník dostane nejenom veškerou potřebnou dokumentaci metodiky ale i vývojové nástroje, poradenské služby a další podporu (IBM 2014). U metodiky DAD to takto jednoduché není. Neexistuje žádný poplatek za používání metodiky. Ta je poměrně dobře popsána na oficiálních stránkách metodiky. Avšak pokud chceme vyvíjet podle DAD, informace z této stránky nestačí a je nutno zakoupit knihu Scotta Amblera a Marka Linese *Disciplined Agile Delivery: A Practioner's Guide to Agile Software Delivery in the*

Enterprise kde je metodika popsána detailně. Na oficiálních stránkách DAD jsou i odkazy na firmy poskytující poradenskou činnost a různé tréninkové programy. Tyto služby však nejsou, jako u RUPu, součástí jedné transakce za získání metodiky a musí se platit zvlášť. (DAD 2015)

4.2. Životní cyklus projektu

Životní cyklus projektu je klíčovou částí pro každou metodiku. Životní cyklus u námi porovnávaných metodik je na první pohled velmi podobný. Právě v této části vychází DAD nejvíce z RUPu. Stejně jako RUP využívá DAD dělení vývoje na části a využívá milníky. Na rozdíl od RUPu nemá DAD fázi elaborační . I když milník splňující tuto část může být obsazen v konstrukční fázi, avšak není povinný. Na rozdíl od RUP, kde je tato fáze povinná (Lines 2012). Ostatní tři části: 1) Inception 2) Construction 3) Transition jsou v obou porovnávaných metodikách.

Níže jsou porovnány jednotlivé části vývoje projektu v obou metodikách mezi sebou:

1. Inception Phase – Úvodní fáze je u obou metodik prakticky stejná. V této fázi se provádí úvodní modelování, specifikují se cíle, provádí se analýza rizik a další. Fáze v obou metodikách končí stejným milníkem. Tedy získání souhlasu s dalším pokračováním projektu. (Lines 2012).
2. Construction Phase – Tato fáze má v obou metodikách společný pouze název, rozdělení na několik iterací a celkový cíl vytvořit konečný produkt. Samotný způsob implementace softwaru je v obou metodikách značně odlišný. RUP má jasně definované disciplíny podle aktivit a artefaktů a jasně říká v jaké fázi a iteraci vývoje a v jaké míře se má daná disciplína provádět. DAD žádné disciplíny, aktivity a artefakty popsané nemá. Konstrukční fáze v DAD je v podstatě stejná jako v metodice Scrum (používá však jinou terminologii). Tedy tato fáze v metodice DAD popisuje pouze metodiku práce a rozhodnutí, co se v které iteraci bude provádět, nechává na členech týmu. (DAD 2015)
3. Transition Phase - Poslední fáze zabývající se předáním a zaváděním vyvinutého softwaru zákazníkovi (v DAD stakeholderovi). DAD tuto fázi zcela převzal z RUPu, je tedy v obou metodikách stejná. (Lines 2012)

4.3. Inspirace DAD v RUP

V předchozích kapitolách práce bylo často zmíněno, že metodika DAD je hybridní. Tedy metodika, která si jde svou vlastní cestou na pomezí agilních a tradičních těžkých metodik. Z obou skupin si vybírá a kombinuje různé části a aspekty užívaných metodik. Zde se zaměříme a shrneme, čím se DAD inspirovalo od metodiky RUP.

Fáze projektu

Metodika DAD měla být určena nejen pro malé agilní týmy ale i pro použití na velkých projektech ve velkých firmách. Proto bylo potřeba pojmout vývoj komplexněji. Tato snaha tak vedla k zakotvení všech důležitých fázi vývoje řešení. V DAD se tedy objevují fáze inspirované od metodiky RUP. Podrobněji byly fáze porovnány v kapitole 4.2.. K zahrnutí fází Inception a Transition vedl i jeden z principu DAD, a to zaměření na řešení. Pro vývoj kvalitního řešení bylo nutné zahrnout jednak počáteční plánovací a analytickou část vývoje, tak i fázi předávání, v níž je často nutné upravit hardware uživatele a i odpovídající procesy. Milníky užívané ke sledování průběhu vývoje byly přeneseny a upraveny i do DAD, zde jsou však zastoupeny coby cíle odpovídajících fází. (Lines, 2012)

Přístup k vývoji

„V tomto ohledu DAD rozšiřuje dosavadní agilní metodiky, které se zaměřují na hodnotu (snaha dodat potenciálně použitelný software v každé iteraci, upřednostňování nejdůležitějších požadavků zákazníka apod.), o větší pozornost věnovanou rizikům.“ (Koudelka, a další, 2015) Jak kolegové zmínili, snaží se DAD řešit otázku rizik již v úvodních fázích projektu. Snaží se tak minimalizovat možné dopady těchto rizik později v průběhu vývoje. Tento přístup se rovněž inspiroval u metodiky RUP a jejího principu zmírňování rizik testováním vyvíjené architektury funkčním řešením, a to co nejdříve v průběhu projektu. (Lines, 2012)

5. Závěr

Cílem této práce bylo porovnat metodiky Disciplined Agile Delivery a Rational Unified Process a najít spojitost, v čem se metodika DAD inspirovala v metodice RUP. V předešlých kapitolách této práce byly obě metodiky představeny. Následně bylo provedeno stručné srovnání zaměřené na obecné specifikace metodik a následně srovnání životního cyklu vývoje v obou metodikách. Zaměření na fáze vývoje je odůvodněno nejvyšší mírou podobnosti DAD a RUP. V následující kapitole byla shrnuta a sepsána odpověď na otázku „V čem se DAD inspiruje u RUPu?“. Odpověď na tuto otázku se nám v této práci podařilo najít.

Bibliografie

Buchalcevová, Alena. 2005. *Metodiky vývoje a údržby informačních systémů : kategorizace, agilní metodiky, vzory pro návrh metodiky.* Praha : Grada, 2005. ISBN 80-247-1075-7.

DAD. 2015. Introduction to DAD. *Disciplined Agile 2.0.* [Online] 2015. [Citace: 28. 12 2015.] http://www.disciplinedagiledelivery.com/introduction-to-dad/.

IBM. 2014. IBM developerWorks. *Download.* [Online] IMB Rational Method Composer, 2014. [Citace: 30. 12 2015.] http://www.ibm.com/developerworks/downloads/r/rup.

Julinek, Pavel. 2008. *Diplomová práce: Použití RUP pro malé SW projekty.* Praha : Masarykova univerzita, 2008.

Koudelka, Tomáš a Ondřej, Sklenář. 2015. *Disciplined Agile Delivery.* 2015.

Lines, Mark. 2012. Comparing the Disciplined Agile Delivery (DAD) framework to RUP. *Comparing the Disciplined Agile Delivery (DAD) framework to RUP.* [Online] 12. Listopad 2012. [Citace: 30. 12 2015.] https://www.ibm.com/developerworks/community/blogs/c914709e-8097-4537-92ef-8982fc416138/entry/comparing_the_disciplined_agile_delivery_dad_framework_to_rup?lang=en.

KROLL, Per a Philippe KRUCHTEN, 2003. *The Rational Unified Process Made Easy: A Practitioner's Guide to the RUP*. B.m.: Addison-Wesley Professional. ISBN 978-0-321-16609-8

STAFFORDSHIRE UNIVERSITY ENTERPRISES LTD., 2008. RUP Summary. *RUP Summary* [online]. Dostupné z: http://projects.staffs.ac.uk/suniwe/project/rup.html

ŠTĚDROŇ, Bohumír, 2007. *Manažerské řízení a informační technologie*. B.m.: Grada Publishing a.s. ISBN 978-80-247-2052-4.

CRAIN, Anthony, 2005. Understanding RUP roles. *Understanding RUP roles* [online] [vid. 28. prosinec 2015]. Dostupné z: http://www.ibm.com/developerworks/rational/library/apr05/crain/

KRUCHTEN, Philippe, 2004. *The Rational Unified Process: An Introduction*. B.m.: Addison-Wesley Professional. ISBN 978-0-321-19770-2.