Univerzita Karlova v Praze

Filozofická fakulta

Ústav informačních studií a knihovnictví

Studijní program: informační studia a knihovnictví

Studijní obor: informační studia a knihovnictví

Diplomová práce

BBS Jan Machynka

Vývoj front-endových aplikací

Development of Front-end Applications

Praha 2013

Vedoucí práce: Ing. Martin Souček, Ph.D.

Prohlášení:

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně a výhradně s použitím citovaných pramenů, literatury a dalších odborných zdrojů.

V Praze, dne 8. srpna 2013

……..

Jan Machynka, BBS

Poděkování

Rád bych poděkoval Ing. Martinu Součkovi, Ph.D., za odborné vedení diplomové práce a cenné rady, které mi poskytl při jejím zpracování.

Identifikační záznam

MACHYNKA, Jan. Vývoj front-endových aplikací: Development of Front-end Applications. Praha, 2013-08-01. 92s. Diplomová práce. Univerzita Karlova v Praze, Filozofická fakulta, Ústav informačních studií a knihovnictví. Vedoucí diplomové práce Ing. Martin Souček, Ph.D.

Abstrakt (česky)

Diplomová práce „Vývoj front-endových aplikací" se zaměřuje na návrh automatizace vývoje podnikových procesů zastřešených jednotnou front-end aplikací. Pomocí explorativního výzkumu odborné literatury práce navrhuje postupy, jak uchopit technologicko-rozvojovou strategii podniku a vyhodnotit potřebu front-end aplikace pomocí podnikové architektury. V dalších kapitolách je předkládán souhrn základních metodik vývoje použitelných pro vývoj automatizovaných podnikových procesů. Pro užší vazbu na obchodní části podniku práce inovativně navrhuje roli front-end specialisty, který nepotřebuje znát základy programování. Práce doporučuje analytické standardy a vývojové nástroje pro nově zavedenou roli, které automatizují podnikové procesy a vytváří front-end aplikaci. Na modelovém příkladu práce ukazuje relativní jednoduchost a praktickou proveditelnost nabízených nových přístupů ve vývoji front-end aplikací.

Klíčová slova (česky)

vývoj front-end, analýza front-end, front-end, automatizace podnikových procesů, efektivní vývoj front-end, podniková architektura, uml a podnikový proces, bpmn, optimalizace podnikových procesů

Abstract (in English):

The diploma thesis on Development of Front-end Applications focuses on the design of computerization of enterprise processes covered by a unified front-end application. Exploratory research was used to suggest working methods to solve technological growth of the enterprise and how to evaluate business need of a front-end application. Later chapters present basic summary of development methodology used for automation of business processes. The diploma thesis innovatively proposes a role of a front-end specialist who does not need to create a program code and has closer relation to business departments. There are recommend analytical standards and development tools for automation of business processes as well as front-end implementation. Finally the work attaches a model example demonstrating relative simplicity and practical realization of new techniques for front-end application development.

Klíčová slova (anglicky):

Front-end development, front-end analysis, front-end, computerization of enterprise processes, automation of enterprise processes, efficient front-end development, enterprise architecture, uml and enterprise process, bpmn, business process reengineering

OBSAH

1 ÚVOD 1

2 ZHODNOCENÍ STÁVAJÍCÍCH POSTUPŮ A NÁSTROJŮ PRO ANALÝZU, VÝVOJ, TESTOVÁNÍ A NASAZENÍ 1

2.1 ZÁKLADNÍ POJMY 2

2.2 OBLAST VYUŽITÍ 3

2.3 ČINNOSTI A PROFESE VEDOUCÍ K FUNKČNÍMU FRONT-ENDU 5

2.4 SPECIALIZOVANÝ TÝM – FE SPECIALISTA 5

2.5 FUNKČNÍ FRONT-END 6

3 VÝVOJOVÝ CYKLUS 7

3.1 PODNIKOVÁ ARCHITEKTURA 7

3.1.1 Zachman 8

3.1.2 TOFAG (The Open Group Architecture Framework) 12

3.1.3 FEA (Federal Enterprise Architecture) 17

3.2 METODIKY VÝVOJE 36

3.2.1 Vodopádový vývoj a rigorózní metodiky 37

3.2.2 Unified Proces 39

3.2.3 Agilní metodiky a techniky 42

3.2.4 SCRUM 43

3.2.5 Zhodnocení vývojových metodik 50

4 ANALÝZA 51

4.1 STANDARDY 52

4.1.1 UML 52

4.1.2 Případy užití 55

4.1.3 Diagramy aktivit 56

4.1.4 BPMN 58

4.1.5 Symboly a sémantika 58

4.1.6 Zhodnocení standardů procesní analýzy 60

4.2 ČINNOSTI V RÁMCI ANALÝZY 61

4.2.1 Modelování podnikových procesů a simulace 61

5 VÝVOJ 62

5.1 NÁSTROJE REALIZACE FRONT-ENDOVÝCH APLIKACÍ 62
5.1.1 IBM Business Process Manager 63
5.1.2 IBM Blueworks Live 63
5.1.3 Appian BPM Suit 64
5.1.4 Intalio Create 65
5.1.5 Bonita BPM 66
5.1.6 Oracle Application Development Framework 67
5.2 TEST 67
6 MODELOVÉ POUŽITÍ 68
6.1 PŘÍKLAD POUŽITÍ 68
6.2 ZADÁNÍ 69
6.3 PŘÍPADY UŽITÍ 70
6.3.1 Scénář k případu užití Vyhledat klienta 71
6.3.2 Scénář k případu užití Smazat klienta 72
6.3.3 Scénář k případu užití Vytvořit klienta 72
6.3.4 Scénář k případu užití Změnit údaje o klientovi 72
6.3.5 Scénář k případu užití Založit běžný účet 73
6.3.6 Scénář k případu užití Založit spořící účet 73
6.3.7 Scénář k případu užití Založit životního pojištění 73
6.4 DIAGRAMY AKTIVIT 74
6.5 PARAMETRIZACE 76
6.6 NASAZENÍ 76
6.7 TESTY 76
ZÁVĚR 78
7 SEZNAM POUŽITÉ LITERATURY: 80
8 SEZNAM OBRÁZKŮ: 85

Předmluva

Téma „Vývoj front-endových aplikací“ jsem si zvolil, jelikož se prolíná s mou pracovní zkušeností a zároveň do mého pracovního oboru přináší nový pohled na uchopení a zpracování informací z pohledu mého studijního oboru. Nad to jde o téma, které nemá aktuálně ucelené zachycení v žádné literatuře.

V praxi jsem řešil problematiku, jaké zvolit nástroje pro masové přetváření podnikových procesů do automatizované elektronické podoby. Primární téma je výběr nástrojů pro vhodnou formalizaci a zachycení informací, jak dané podnikové procesy aktuálně fungují, a nástroje, metody, techniky, jak zachytit informace navrhující budoucí stav řešení. Sekundární oblast mapuje možné techniky řízení podnikové architektury jakožto cesty k rozhodnutí o implementaci nástroje pro tvorbu front-endu a automatizovaných podnikových procesů. Praxe ukazuje, že na toto téma není zcela jednotný pohled.

Osnova práce vychází z praxe, tak aby zachytila pro podnik všechny důležité kroky od rozhodnutí, že se jim vyplatí robustní řešení na automatizovaný vývoj fronte-endu, po úspěšnou realizaci prvních procesů v novém nástroji. Pro každou svébytnou kapitolu práce jsem analyzoval existující informační zdroje, zejména EBSCO host. Průběžně jsem některé z témat konzultoval s kolegy z podnikové praxe. Výsledky analýzy jako návrhů možných řešení jsem aplikoval do kapitol: Podniková architektura; Metodiky vývoje; Standardy; Nástroje realizace front-endových aplikací.

Jako citační styl jsem zvolil ČSN ISO 690.

1 Úvod

Cílem diplomové práce je popsat jednotlivé části vývoje front-endových aplikací, které představují svébytný typ softwaru se specifickými požadavky na implementaci. Tyto aplikace se vyznačují přímou interakcí s uživatelem a grafickým rozhraním na jejich obsluhu, tzv. GUI (graphical user interface). Pro jejich funkčnost je však nutné propojení s back-endovým systémem jakožto aplikací v pozadí, bez interakce s uživatelem, který představuje v obecném pojetí databázi pro správu a zpracování dat.

Často je front-end implementován jiným kolektivem či dokonce jinou společností nežli back-end. Tato situace vyžaduje kvalitní zpracování zadání, ale nově i důkladnou interakci s okolními systémy (back-endy). To klade velký důraz na standardizaci postupů a pravidel vedoucích k úspěšnému vyvinutí a následnému používání aplikací.

Práce popisuje dnes používané metody vývoje (př. vodopádový vývoj) a nástrojů pro jednotlivé fáze (př. UML), s důrazem na dodržení životního cyklu vývoje softwaru. Zaměřuje se na specifika při vývoji front-endu. Hlavním tématem práce je návrh ucelených postupů pro implementaci tak, aby aplikace byla zdokumentována, splňovala požadavky zadavatele a její části byly případně znovu použitelné.

Pro ucelenost tématu práce obsahuje také základní pohled na podnikovou architekturu, která by měla existovat v každé střední a velké firmě. Díky strategickému pohledu na efektivní koexistenci informačních technologií a obchodních/výkonných částí podniku pak snižuje náklady, zvyšuje šanci na včasnou realizaci IT projektů apod.

2 Zhodnocení stávajících postupů a nástrojů pro analýzu, vývoj, testování a nasazení

Úvodní část práce osvětluje pojem front-endová aplikace v návaznosti na možnosti a důvody využívání těchto aplikací v dnešním světě. Klade velký důraz na klasické požadavky dnešní komerční sféry, tedy na nutnost flexibility, propojitelnost distribuovaných systémů, připojení na IT infrastrukturu (nicméně tato témata jsou pro práci jen sekundární).

2.1 Základní pojmy

Front-endová aplikace je část programu, kterou vidí a nějakým způsobem obsluhuje sám uživatel. V konceptu informační architektury, kde používáme označení front-end, je též nezbytný pojem back-end. To je systém v pozadí, který nemá nativní uživatelské rozhraní, většinou slouží jako úložiště dat, řídících procesů nebo má obě funkce zároveň. Back-end je část aplikace v pozadí, která je uživateli vizuálně skryta a nepotřebuje o ní ke své práci vědět. V praxi se také setkáváme se zastřešujícím pojmem front-end a back-end aplikace. Aby program fungoval, musí být vyřešeny obě části. Jde však o velice volný svazek, jak si představíme později. Části back-end se budeme věnovat jen v nejnutnější míře.

V generalizované podobě jsou tedy pojmy front-end a back-end označením dvou různých částí jednoho celku – softwarové aplikace. Front-end definujeme jako část aplikace, která řeší grafické rozhraní s uživatelem, zastřešuje určitý postup – činnost (například sled obrazovek) a přináší nám díky své velké nezávislosti na back-endu technické výhody.

Pro jiný úhel pohledu na front-end si vypůjčím slova Jiřího Peterky: „Přívlastkem či substantivem "front end" se zde míní něco, co zajišťuje předzpracování či jiné počáteční akce, které lze udělat předem, před vlastním (skutečným, hlavním) zpracováním, co mu může nějakým způsobem odlehčit nebo co vhodným způsobem zpřístupňuje *to, co je vzadu* (a co se pak označuje jako "back end")“ [PETERKA, 1994].

Dalšími pojmy k uvedení do problematiky jsou klient a server. Pojmy chápeme jako místně popisné. Záleží, na které části hardwaru (počítačích) jsou jejich úkoly vykonávány. Pro naše potřeby je to označení dvou částí jedné softwarové aplikace. Zde však slovo klient říká, že takto označená část programu má být vykonána na klientském počítači (rozumějme počítači koncového uživatele). Z druhé strany server označuje nějaký centralizovaný bod (serverový počítač), který poskytuje služby, schraňuje a sdílí data pro více klientských počítačů. Nutno dodat, že ne vždy musíme mít fyzicky alespoň dva počítače, jeden jako klient a druhý jako server. I v rámci jednoho fyzického počítače se může naplnit filozofie klient-server architektury. Nicméně hlavní výhody plynou z logického rozdělení na část klient a server. Pro pochopení v rámci této práce je nejvhodnější představa klienta a serveru jako dvou fyzicky různých počítačů.

Pojem klient má v rámci detailní terminologie ještě dvě samostatná řešení. První varianta, takzvaný tlustý klient, je robustní typ programu (můžeme také slýchat klient s exe souborem apod.), který v sobě má jak grafické rozhraní pro komunikaci s uživatelem a definicí činností, které mu nabídne, ale také obsahuje velkou sadu operací pro komunikaci se serverem. V těchto případech je serverová část z velké části jen úložištěm sdílených dat. Toto řešení má větší nároky na koncový počítač, a pokud chceme přejít na novou verzi, musíme na každý z počítačů, kde je tlustý klient, nahrát novou verzi atd.

Druhá varianta, takzvaný tenký klient, si ponechává na straně klientského počítače velice málo informací. Dnes jsou většinou řešeny přes webové rozhraní. Uživateli tedy postačí internetový prohlížeč, přístup k síti a zadání správné adresy aplikace. Tenký klient pak všechny instrukce (o tom jakou obrazovku zobrazit, v jakém pořadí, s kterými údaji apod.) obdrží od serveru. Toto řešení nepotřebuje silné počítače pro koncové uživatele. Není potřeba distribuovat nové verze programu, protože se po přihlášení přes web načte vždy nejnovější verze ze serveru [PETERKA, 1994].

Tato práce se zabývá tvorbou front-endu s architekturou klient-server a využitím tzv. tenkého klienta.

2.2 Oblast využití

V čase dynamického chování moderních trhů všech odvětví je kladen silný důraz na rychlý vývoj a uvádění nových výrobků na trh. Předmětem tohoto snažení jsou kvalitní obchodní (business) procesy, stejně tak činnosti řídící (management) a kontroly kvality. Výzkumy ukazují, že důležitá je již výchozí fáze, takzvané uvedení produktu na trh. Je známé spíš pod zkratkou NPI, vycházející z anglických slov New Product Introduction, kde se definují základní očekávání od zamýšleného produktu. Z pohledu činnosti na to můžeme taktéž nahlížet jako na obchodní činnost vyvíjející a následně uvádějící nový výrobek na trh.

Praxe ukazuje, že proces uvedení nových výrobků na trh je horší tam, kde není používáno měřitelných kritérii NPI. V podnicích bez jasné metodiky se vývoj produktu odvíjí od ad-hoc činností jednotlivých oddělení, které si mezi sebou předávají zodpovědnost až po samotné uvedení produktu na trh [WILLIAMS, 2007].

V podniku se silnou automatizací vydefinování produktu velmi souvisí s využitím front-endu, kde možnost tvorby jednoduchého a rychlého front-endu přináší šanci pro efektivní implementace nových produktů. Pro měření úspěšnosti lze využít NPI.

Další výhody jsou spíše technologické, avšak z pohledu podniku využívající takovéto řešení jsou nezanedbatelné. Pokud se rozhodneme pro front-end fungující jako samostatný modul, který bude technický nezávislý na ostatních částech IT infrastruktury, dostáváme do rukou řešení, které časem můžeme použít jako front-end i pro aplikace již dříve nasazené do provozu. Můžeme dosáhnout toho, že celá firma bude mít pro mnoho aplikací na pozadí jen jeden front-end. Z hlediska uživatelů tedy bude používán jen jeden program se stejnou logikou uživatelského rozhraní. A jen zasvěcení budou vědět, že na pozadí, na úrovni back-end, je aplikací mnoho. Výše popsané přinese dané firmě úspory při zaškolování nových zaměstnanců, protože všechny aplikace budou mít stále stejnou logiku fungování. Kombinace technologie a profese front-endového specialisty nám též zaručí, že různé firemní činnosti nebudou popsány jen v procesních dokumentech různých oddělení, ale budou popsány i standardizovanou formou na straně front-endových specialistů. Tudíž všechny automatizované činnosti pomocí front-endu budou mít i jednotné úložiště metadat popisující řešené činnosti firmy. Čistě technologickou výhodou odvislou od konkrétního řešení je možnost napojení na stávající IT infrastrukturu nejen pro možnost přenosu dat, ale i pro využití různých služeb jako je správa uživatelských práv, či monitoring chodu aplikací. Tím prohloubíme integritu podnikových systémů a současně ušetříme na nákladech v případě vytváření např. modulu pro správu práv jen pro front-end [LEVITAN, 2008], [SALAHAT, 2009], [Podniková architektura, 2001].

Svébytná aplikace pro jednoduchou tvorbu front-endu nemá jen klady. Popisované řešení je velice nákladné. Na trhu není žádné krabicové řešení, které by neobsahovalo náklady na zapojení do vlastní infrastruktury. Nelze tedy koupit takový software, nainstalovat jej a do pár minut jej začít používat. Druhou možností řešení je zaplatit si vývoj takovéto aplikace, případně najít standardizované řešení a upravit jej na míru. Samotný software pro tvorbu front-endu bude potřebovat zejména lidi, kteří jej budou tvořit. Proto je nutné počítat s náklady na vytvoření typově nové profese v rámci firmy, jejím zaškolení a hlavně začlenění do firemních činností.

Všechna zmíněná negativa poukazují na robustnost řešení, nutnost velkých počátečních investic, ale i na nezanedbatelné provozní náklady. Návratnost a zvýšení efektivity je možná u podniku s mnoha složitými podnikovými procesy, které je nutno automatizovat. Podnik, který si pořídí takovéto řešení, má k dispozici mechanizmus, který mu umožní rychle reagovat na změnu na trhu (rychle a snadno bude moci změnit své firemní procesy a uživatelské rozhraní k nim). Firemní procesy budou jasně popsány, proto i ve znalostní rovině budou změny rychlejší a lépe kontrolovatelné [MUKNŠNÁBL, 2001], [TENNANT, 2001], [LIN, 2009].

2.3 Činnosti a profese vedoucí k funkčnímu front-endu

Za činnosti vedoucí k funkčnímu front-endu považujeme ty úkony, které bude vykonávat front-endový specialista (dále jen FE specialista) a které v souladu s jinými profesemi povedou k provozu front-endové aplikace nebo její části. Nutným předpokladem pro realizaci jednotlivých činností je existence zadání popisujícího řešený požadavek. Na druhém konci řešení počítáme s tím, že o back-endové aplikace se stará jiná profese („klasický“ programátor) a o provoz se stará profese IT provozního specialisty. FE specialista tedy pokrývá činnosti od převzetí zadání po předání hotové front-endové aplikace do provozu.

2.4 Specializovaný tým – FE specialista

Abychom mohli jednotlivé obrazovky a za nimi skryté činnosti jednoduše vytvářet, standardizovat a udržovat, potřebujeme aplikaci, ve které budeme jednotlivé části front-endu vytvářet - aplikaci vytvářející front-endovou aplikaci.

Tento typ aplikace nás ve větším podniku, který má vlastní technická oddělení, dovede až k vytvoření specializovaného týmu. V něm budou specialisté na navrhování technické realizace obchodních procesů, které budou řešeny pomocí obrazovek front-endu. Takoví FE specialisté budou schopni komunikovat s odborníky z různých oblastí, kde se bude vytvářet zadání definující chování budoucích aplikací, tedy i obrazovek a jejich obsahu. Tito specialisté tedy budou zdatnými partnery zadavatelům. Díky samostatnosti front-endu, budou z druhé strany komunikovat i se zástupci koncových aplikací (IT odborníci), ale jasné oddělení obou částí aplikace umožní menší potřebu znalostí z oblasti IT. Po profesní stránce bude mít firma využívající takovéto řešení novou profesi, která

bude víc osvobozena od technických záležitostí, a o to více bude rozumět jednotlivým činnostem dané společnosti, ale stále bude technicky schopná vytvořit obrazovky. Z pohledu personalisty nebudeme pro tvorbu front-endu s výše popsanou vizí potřebovat IT odborníky (programátory), ale budeme vyhledávat lidi se základními znalostmi informačních technologií, kteří budou mít současně zájem o firemní procesy daného oboru (př.: bankovnictví, telekomunikace, automobilky apod.). V konečném důsledku v koloběhu tvorby front-endu jeden typ profese ušetříme, protože čistý programátor by pravděpodobně neměl čas, znalosti ani zájem o častou komunikaci přímo se zadavateli. Museli bychom stejně ještě zavést profesi určitého konzultanta, který by transformoval požadavky zadavatelů do detailu pro samotný vývoj aplikace.

2.5 Funkční front-end

Vývoj front-endu se v zásadě neliší od vývoje softwaru obecně. V pojetí této diplomové práce však počítáme s masovou tvorbou procesů, které pomocí obrazovek vytváří komunikační rozhraní mezi systémem a uživatelem. Taktéž počítáme s existencí aplikace, která výrazně usnadní tvorbu samotného obsahu front-endu. Díky tomu ušetříme část práce vzniklé profese FE specialisty, na druhou stranu nám to umožní přidat další činnost, a to analýzu zadání a účast na vytváření testovacích scénářů společně s testy. Proto ať vybereme kteroukoliv metodiku, i zde neřešenou, musíme se zaměřit na kroky řešící oblasti specifikace, návrhu, implementace a testů. Pro rozhodnutí, kterou ze zde řešených metodik zvolit pro vývoj front-endu, musíme přibrat ještě dvě kritéria. Jak velký projekt řešíme? A týká se jen front-endu nebo i dalších IT oblastí?

Pokud stojíme před menším požadavkem či malým projektem, který se týká z pohledu IT jen front-endu (například tvorba rozhraní pro stávající back-end), pak je nejlepší volbou vodopádový vývoj. Jelikož FE specialista řeší část práce hned tří lidí z pohledu této metodiky (analytik, vývojář a tester) a není přitom závislý na výstupech jiných IT profesí (protože v tomto příkladu se nově tvoří jen front-end), dosáhneme největší efektivity. Nemělo by smysl dávat paralelně více FE specialistů na jednotlivé části (analýza, vývoj, test) tvorby front-endu. V tomto případě je tedy nejvhodnější ponechat řešení na jednom FE specialistovi a jít ve sledu kroků metodiky. Nezanedbatelnou výhodou je zde i menší riziko nesplnění předem odhadnutého času realizace. Jasně daný

sled dokončených kroků metodiky při jasném zadání, řešený jednou osobou, téměř zaručuje přesný odhad pracnosti projektu a jeho trvání.

Jiná situace nastává, stojíme-li před velkým projektem, na kterém se podílí více IT elementů. Na příklad pokud se vytváří nový front-end a zároveň nový back-end, dá se pak předpokládat i častější změna zadání a nutnost většího pochopení pro víc profesí. V takovémto případě se využijí iterace a také to, že na určitých fázích budou současně pracovat různé profese. Proto je doporučeno užití metodiky Unified Software Development Process [ARRLOW, 2008], [Just Another Blog by JohnNy, 2008], [Vodopádový model, 2008], [SALAHAT, 2009].

3 Vývojový cyklus

Kapitola je zaměřena na dvě roviny: na strategické řízení podniku a jeho technický rozvoj pomocí podnikové architektury a na část technik/metodik samotné realizace projektů s velkým obsahem informačních technologií.

3.1 Podniková architektura

V podnikové praxi se setkáme zejména s anglicismem Enterprise Architecture, proto dále v textu naleznete i zkratku anglického originálu EA. Jde o architektonický pohled na obchodně technické fungování firmy se silným důrazem na znalost budoucích obchodně-podnikových potřeb a znalost rozvoje daného subjektu tak, že tyto potřeby zohledňujeme a dáváme do souladu se stávající a budoucí IT architekturou firmy. Enterprise architekt je osoba dívající se komplexně na všechny IT systémy podniku, která zná jejich základní poslání a současně spolupracuje s obchodními složkami podniku, kde jeho hlavním cílem je zastřešit komplexně, rychle a levně obchodní požadavky do rozvoje IT infrastruktury.

Podniková architektura zahrnuje popis cílů organizace, způsobů, jak jsou tyto cíle dosahovány pomocí podnikových procesů a způsobů, jak mohou tyto procesy být podpořeny technologiemi [Podniková architektura, 2008]

Hloubka detailu práce EA je na vyšší až střední úrovni, stejně jako u architektury ze stavebnictví, kde např. detailnější návrhy rozvodu vody v domě dělá jiná profese apod. V každém případě doména EA má své informační potřeby a informace, které získává, spravuje a vytváří, musí nějakými pracovními postupy zpracovat. Zde má každý podnik

dvě základní cesty realizace. Buď si vše vymyslí od základu po svém, nebo použije některou z metodik či jejich kombinaci. Přehled metodik podporujících informační činnosti oboru enterprise architektury s detailním popisem je obsahem následujících kapitol.

3.1.1 Zachman

Jde o první metodiku / pracovní rámec, který se na poli EA objevil. První verze se datuje k roku 1987. Poskytuje vysoce strukturovaný pohled na podnik a jeho definování. Zachman píše o architektuře jako o nástroji pro komplexní a měnící se činnosti, kde stejně jako při stavbě / tvorbě čehokoliv, je potřeba si sepsat od shora dolů kroky a hranice dané činnosti. To platí i pro změny. Informační věk přináší architektonický pohled i do oblastí mimo stavebnictví, vyvolává potřebu vytvářet, udržovat a měnit složité informační systémy. Proto vzniklo odvětví podnikové architektury.

Zachmanův pracovní rámec pro podnikovou architekturu je normalizované schéma, které drží metainformace na jednom místě. Proto je dobrým analytickým podkladem pro inicializační návrh projektů a jejich průběžnou kontrolu, zda jsme se neodchýlili od jednotlivých požadavků. Dle Johna A. Zachmana jím navržený pracovní rámec pomáhá s analýzou čehokoli, zejména pak podniku nebo jeho části. Tvrdí: „Čím šířeji definujeme analytické cíle, tím silnější dopad to bude mít na budoucí integraci, znovupoužitelnost, interoperatibilitu atd. Obráceně, čím užší cíle si vytýčíme, tím méně ovlivníme důležité chování podniku a jeho procesy v důležitých oblastech jako je integrace, znuvupoužitelnost, interoperatibilita atd. Zároveň pokud si širokými cíli přesahujeme hranice své jurisdikce, nemůžeme ami ručit za vytvořené modely, musíme vést jednání s ostatními aktéry v podniku. Když si dáme úzké cíle, máme plnou kontrolu, ale zároveň nepopisujeme podnik v celé jeho integritě“ [ZACHMAN, 2008].

Vhodné je si uvědomit, že pokud jsme již nové systémy pro podnik vyhotovili, máme již aplikace, databáze, jejich návrh a implementaci, jen těžko je zásadně budeme měnit.

Z pohledu Zachmanova pracovního rámce není vždy nutné pro každou oblast tvořit modely. Vždy je dobré zvážit předpoklady, za kterých je můžeme oželet a rizika, která bez nich mohou nastat.

Z toho plyne, že při započetí nějaké aktivity, od které očekáváme zásadní vliv na podnik, musíme jasně definovat její dopady na podnik a rozumně zmapovat, na které aktéry a role ve firmě má tato akce vliv. Taktéž bychom neměli počítat s tím, že zásadní vlastnosti a očekávání zakomponujeme do systémů až někdy později. Vše musí obsahovat již ucelený pohled v návrhu na základě pracovního rámce pro podnikovou architekturu. Na závěr důkladně zvažme náklady na tvorbu jednolitých modelů. Pro jednotlivé oblasti pracovního rámce a za pomocí definice předpokladů, které by měly nastat, a potencionálních rizik, která nám jejich neexistence může přinést, se rozhodneme. Autor doporučuje nevynechat sloupce 1, 3 a 6, tedy oblasti s otázkami co, kde a proč.

Hlavní roli v Zachmanově metodice hraje dvourozměrné pojetí za pomoci matice, kde sloupce nesou otázky: co, jak, kde, kdo, kdy, proč a řádky představují transformace obecných představ do již konkrétní úrovně modelování a definování rozsahu projektu. Zde jsou vedeny řádky: kontextová úroveň, koncepční, logická, fyzická a detailní/implementační. Filozofie schématu upřednostňuje různé perspektivy - přes celý proces návrhu až po tvorbu a uvádění architektury podniku do provozu. Jde o popisnou reprezentaci vyjadřující koncept a omezení relevantní pro různé perspektivy. To znamená popsat jak nezbytné informace pro projektantskou činnost, tak i informace pro ostatní aktéry podnikové architektury s pochopitelností z jejich perspektivy.

Perspektiva vlastníka (řádek 2) – příjemce (zákazník, uživatel, vlastník) koncového produktu (v našem případě podnik nebo jeho části). Popisnou reprezentací reflektujeme užitné vlastnosti produktu, které bude vlastník využívat (příkladem nový front-end).

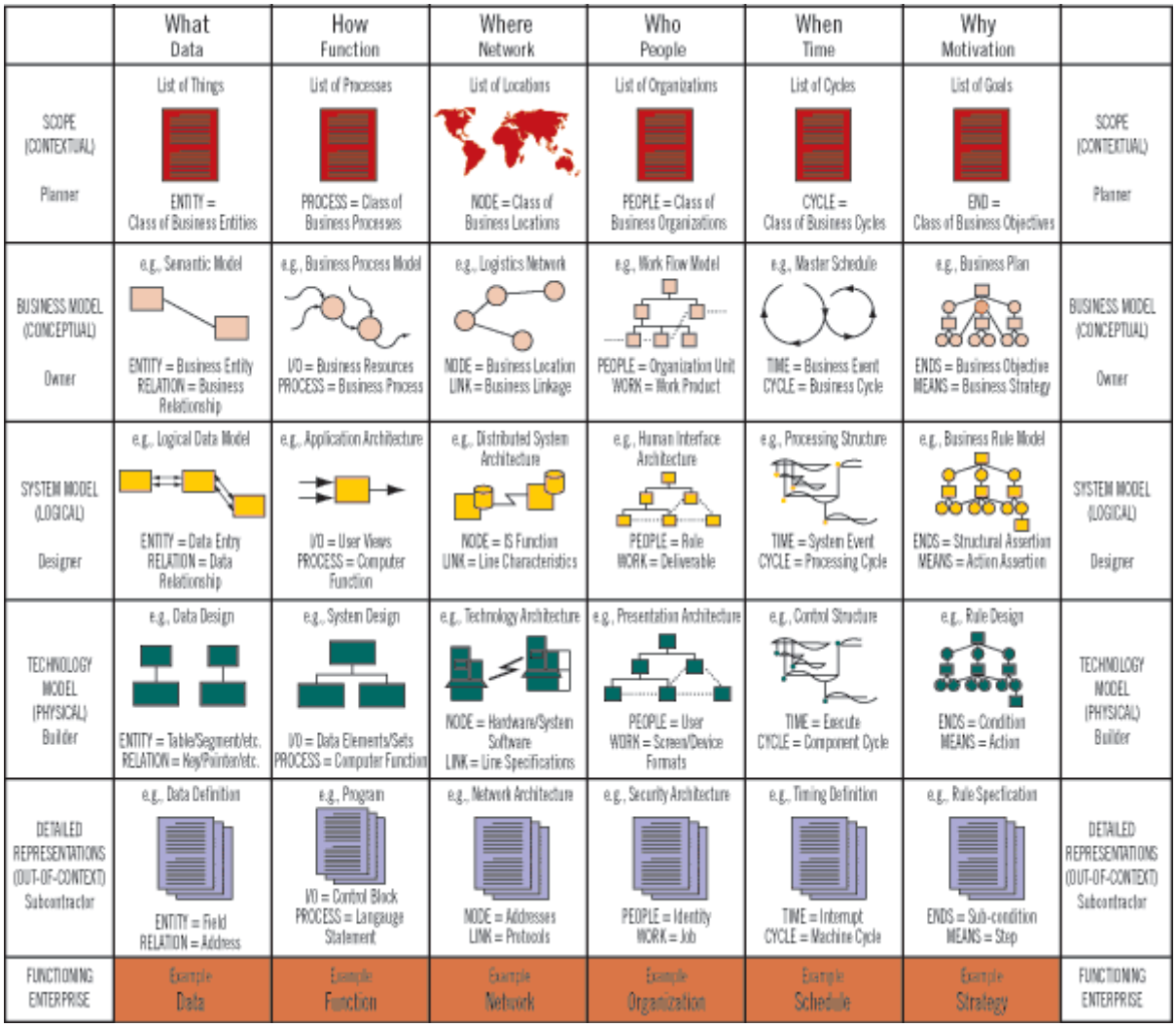
Perspektiva návrháře (řádek 3) – inženýr, architekt, prostředník mezi tím, co je požadováno (řádek 2) a tím, co je fyzicky a technicky možné (řádek 4). Tato část schématu odráží vlastnosti a charakteristiku systému, logických pohled na koncový produkt a jeho logická omezení. Z pohledu podniku jde o jeho základní logickou reprezentaci ve formě popisu systému účetnictví, systému správy dokumentů, systémy manažerů, logistika apod.

Perspektiva stavitele/tvůrce (řádek 4) – IT inženýr, programátor, IT dodavatel, zaměstnanec s technickými znalostmi umožňující realizaci koncového produktu. Odráží fyzické omezení použitých technologií koncového produktu.

Dále jsou použity dva jiné pohledy:

Perspektiva rozsahu (řádek 1) – kontext, který nám říká rozsah našich aktivit, univerzum, ve kterém bychom se měli pohybovat. Deklaruje hranice, které máme naplnit a nemáme je překročit.

Perspektiva technické reprezentace (řádek 5) – detailní popis implementace jednotlivých částí [ZACHMAN, 2008].



Obrázek 1 Zachmanova matice. Zdroj: [ZACHMAN, 2008]

Jednotlivé řádky jsou primitivy návrhu jakékoliv architektury. Stejné oblasti se ukázaly jako vhodné jak v klasické stavební architektuře, tak i v architektuře podniku. Lze říci, že jsou všechny silně odlišné a na sobě nezávislé, a tak další perspektivy ke kompletnímu a integrálnímu pohledu na podnik nepotřebujeme. Minimální části schématu pro celkový pohled pak jsou řádky rozsah, perspektiva vlastníka a poslední řádek vyjadřující výsledek celého snažení za danou oblast.

3.1.1.1 Abstrakce pracovního rámce

Jiná dimenze klasifikace řešeného schématu je pomocí nezávislých proměnných, které vytvářejí kompletní pohled na jednotlivé popisované objekty:

Co – představuje otázku, co to je? Z čeho to je? Pohled na materiální složení popisovaného objektu. V případě podniku jde o pohled na informace/data, datový model podniku, sloupec 1.

Jak – podnik pracuje/funguje – funkční specifikace, procesy. Popis fungování podniku přes procesní/funkční model, sloupec 2.

Kde – jsou komponenty umístněny, jaké mají mezi sebou vztahy. Z pohledu podniku řešíme logistiku, připojení systémů, sítě, sloupec 3.

Kdo – co a jak dělá– manuály a provozní příručky. Pro podnik popisujeme role lidí, model toků informací, sloupec 4.

Kdy – daná věc nastává a její vztah k ostatním objektům. Zajímá nás životní cyklus a časový harmonogram. U podniku řešíme veličinu času, sloupec 5.

Proč – se věci dějí – ve smyslu motivace proč chceme dělat změny, proč to dnes funguje tak, jak to funguje a jak to má fungovat ve finále, sloupec 6.

Co? Materiální popis	**Jak?** Funkční popis	**Kde?** Prostorový popis	**Kdo?** Provozní popis	**Kdy?** Časování	**Proč?** Motivační popis
Struktura (věci)	Transformace (procesy)	Toky informací (umístnění)	Provoz (lidi)	Dynamika (události)	Motivace (Strategie)

Obrázek 2 Zachman - úrovně abstrakce. Překlad: [ZACHMAN, 2008]

Šest abstraktních otázek rozděluje naši práci na základní a přitom úplné části, které dávají holistický pohled na celý podnik.

3.1.1.2 Výběr ze Zachmanových pravidel ke schématu

Není vhodné přidávat do schématu sloupce ani řádky. Každý sloupec by měl mít svůj jednoduchý model. Pro každou buňku by měl vzniknout specializovaný model od generického modelu sloupce, protože perspektiva řádku vnáší svůj specifický pohled (ať už na použitý slovník, technické znalosti atd.) Detail popisu říká perspektiva řádku ne sloupec. Žádný metakoncept by neměl být ve více než v jedné buňce. Netvořte diagonální vztahy mezi buňkami schématu. Neměňte názvy řádků a sloupců. Schéma je generické, tedy natolik obecné, že ho lze použít na cokoliv [ZACHMAN, 2008].

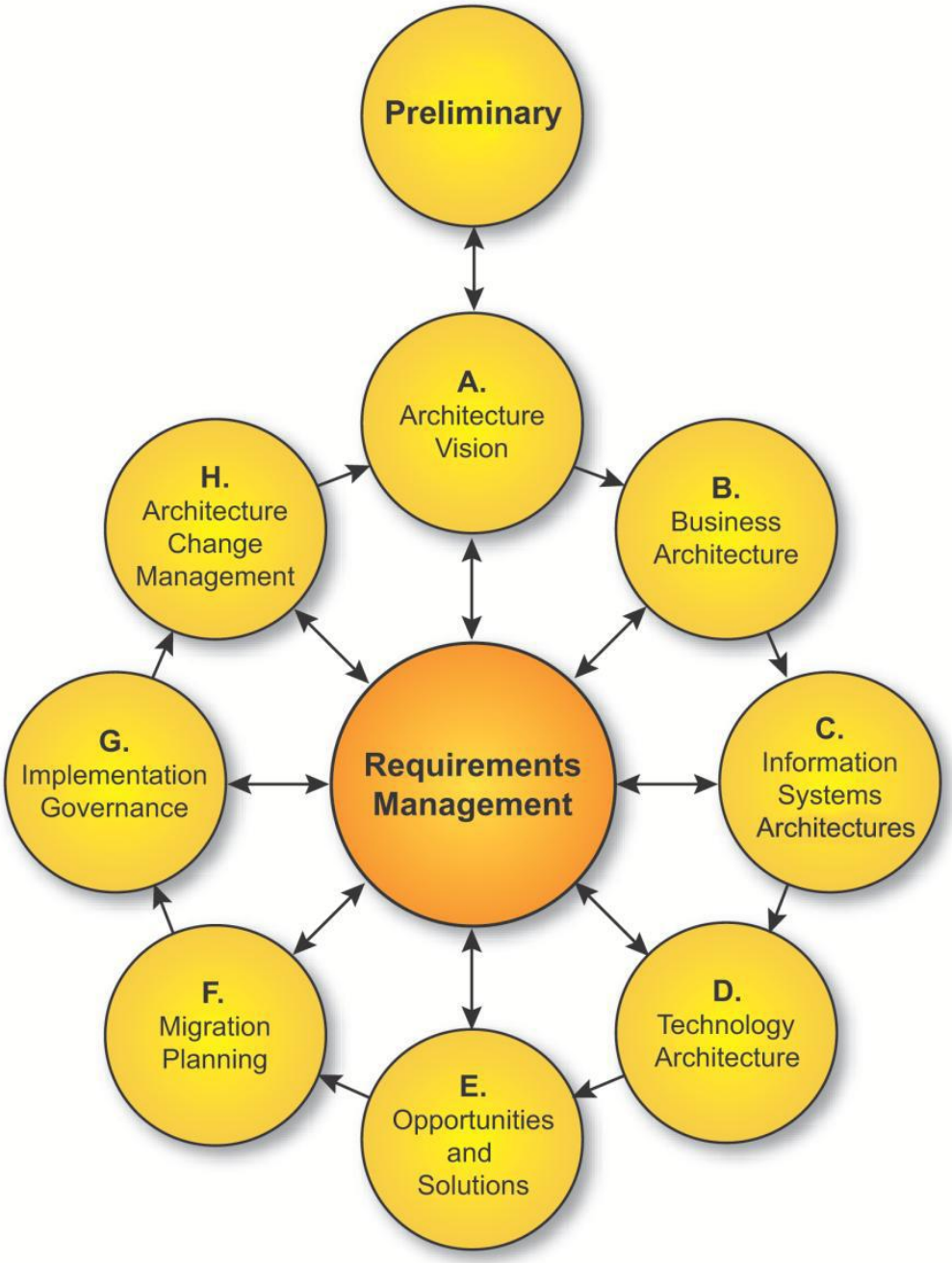
3.1.1.3 Shrnutí Zachmanova rámce

Zachmanův pracovní rámec je vhodný jako reference k naší snaze zmapovat zásadní změny v chování podniku. Dobře pomáhá v uvědomění, zda jsme něco nezapomněli, a ke sledování, zda se držíme na vytyčené cestě. Nicméně tento pracovní rámec vůbec nereflektuje, že již třeba nějaká podniková architektura existuje. Je příliš generický a v dnešní době není vhodné používat pouze tento nástroj sám o sobě.

3.1.2 TOFAG (The Open Group Architecture Framework)

Jde o pracovní rámec s velkou snahou o co nejlepší a včasné zavádění změn v podniku. TOGAF se zaměřuje na velké podniky, metodika byla vyvinuta organizací The Open Group Architecture Forum. Je kontinuálně vyvíjen od druhé poloviny 90. let. Samotná metodika navazuje na Technical Architecture Framework Management (TAFIM), které poskytlo americké ministerstvo obrany jako vstup. Samotný TAFIM byl vyvíjen mnoho let a americká vláda do něj investovala mnoho milionů dolarů. Aktuálně je na trhu dostupný TOGAF ve verzi 9, respektive 9.1.

Vývoj informačních systémů velkých podniků a jejich velkých útvarů se stává stále komplexnější, zároveň však vyžaduje stále větší flexibilitu. Toto vede k vzniku metodik, které pro návrh a řízení počítají se strukturovaným přístupem vycházejícím z každodenní zkušenosti a podpory jejich potřeb. V tomto duchu mění teoretické modely, jak by mělo něco fungovat, do procesů, které fungují v praxi. Jde o procesně zaměřený pracovní rámec, jehož hlavní schéma ukazuje procesy řízení a akce, které by měly pro úspěšný projekt změny v podniku nastat.



Obrázek 3 TOGAF proces řízení změn. Zdroj: [An Overview of TOGAF® Version 9.1, 2011]

Přípravná fáze (Preliminary) – příprava podniku na úspěšný projekt změny podnikové architektury.

Architektonická vize (Architecture Vision) – definování rozsahu projektu, jeho omezení, nastavení očekávání od TOGAFu. Obsahuje vyhodnocení business kontextu (porovnání vůči zákazníkům, uživatelům netechnického charakteru, naopak ti co naplňují

poslání podniku). Měl by být vytvořen tzv. the Statement of Architecture Work, dokument, který oficiálně stanovuje rozsah projektu a říká, jakým přístupem se cíle naplní. Mimo jiné obsahuje požadavky, popis okolí, role, zodpovědnosti, co má být dodáno (až na úroveň dokumentace apod.) a je podepsán hlavními aktéry (tzv. stakeholders) a sponzory (většinou vedení podniku).

Obchodní architektura (Business Architecture) – architektura jednotlivých obchodních funkcionalit. Zahrnuje vytvoření obchodní architektury, popis stávajícího stavu, cílového stavu a definování rozdílů (tzv. gaps).

Architektura informačních systémů (Information Systems Architectures) – vytvoření architektury informačních systémů, popis stávajícího stavu, cílového stavu a definování rozdílů (tzv. gaps).

Architektura technologií (Technology Architecture) - vytvoření architektury informačních systémů, popis stávajícího stavu, cílového stavu a definování rozdílů (tzv. gaps).

Příležitosti a řešení (Opportunities and Solutions) – vytvoření iniciálního plánu implementace projektu. Jde o část identifikující a kompletující podklady z předchozích třech částí (obchodní, informační a technologická architektura). Rozhoduje, jaké projekty, programy, aplikace a produkty podniku budou řešeny. Určuje typ přístupu k projektu, kupříkladu zda se podniková architektura bude řešit přírůstkově, a pokud ano, tak jestli bude muset být řešena dočasná (Transition) architektura.

Plán migrace (Migration Planning) – analýza nákladů, přínosů a rizik (CBA – Cost Benfit Analysis), vytvoření detailního plánu implementace a migračních plánů.

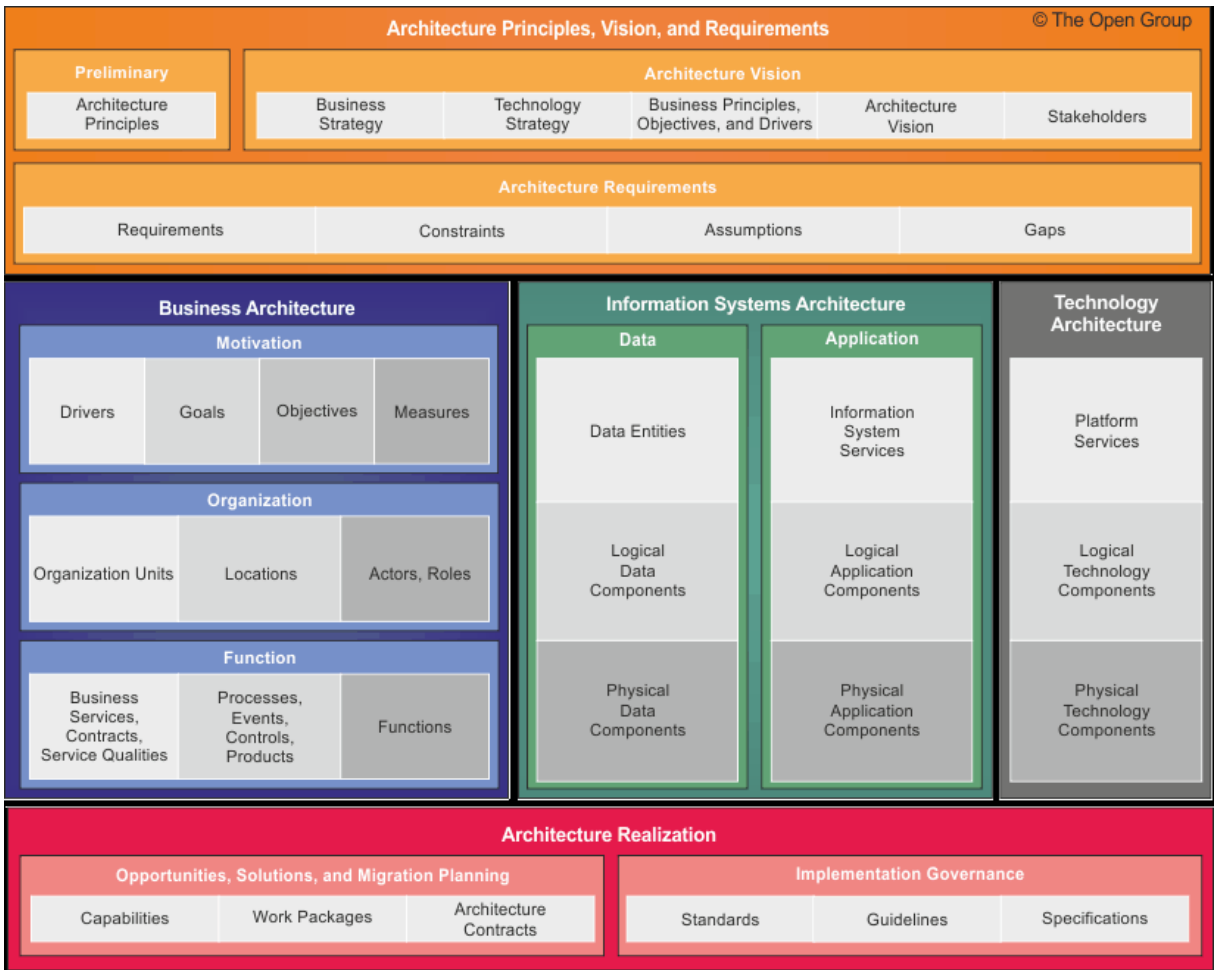
Dohled nad implementací (Implementation Governance) – dohled, že to, co se realizuje, je v souladu s předem deklarovanou architektonickou vizí.

Změnové řízení (Architecture Change Management) – proces managementu změn, zajištění, že jsou změny v souladu s plánovanou/aktuálně realizovanou architekturou nebo, že se změny zapracují tak, aby v souladu byly.

Management požadavků (requirments management) – ujistit se, že v každém kroku metodiky jsme stále v souladu s požadavky podniku.

Metodika má pro každý krok A-H velice podrobně popsaný set dokumentů a aktivit, které mají vzniknout, ale vzhledem k cílům této práce je zde podrobně nerozepisuji. Metodika taktéž pečlivě řeší jednotlivé účastníky zavádění nové podnikové architektury či změny její části. Rozlišuje pět oblastí: korporátní, koncoví uživatelé, projektové řízení, provoz systémů a externí oblast. Do korporátní části spadá vedení firmy (v případě velkých firem tzv. board management, CxO), IT/obecná bezpečnost, projektová kancelář, oddělení vnitřních a korporátních standardů a kontroly, nákup a oddělení lidských zdrojů. V sekci koncových uživatelů rozlišuje liniový management, experty na jednotlivé obchodní domény a vlastníky dat. Projektové řízení obsahuje role: výkonné (projektoví manažeři apod.), opět liniový management, experty na obchodní procesy, produktové specialisty a technické specialisty. Z pohledu provozu systémů jsou středem zájmu oddělení IT provozu, helpdesk (IT podpora uživatelům), aplikační management, oddělení infrastruktury a datové/hlasové komunikace. V neposlední řadě se musíme zajímat o externí role, a to o dodavatele a regulátory [TOGAF, 2013].

TOGAF velice detailně řeší přístup k dokumentaci. Diagram níže popisuje, co má být dodáno, jaké artefakty mají být dodány a v jakých fázích. To vede k velké konzistenci a možnosti plné kontroly nad tím, že bude dodáno vše potřebné. Zajistí se tím lepší integrace budoucích změn do podniku, budou existovat standardy a metamodel podniku nebo jeho řešené části.



Obrázek 4 TOGAF schéma dokumentace pro jednotlivé části projektu. Zdroj: [An Overview of TOGAF® Version 9.1, 2011]

3.1.2.1 Shrnutí TOGAF

TOGAF poskytuje silně procesně a prakticky orientovanou metodiku, které provede podnik podnikovou architekturou od úvodních nápadů, požadavků a vizí až po implementaci a provoz podniku v nové architektuře. Mezi negativa TOGAFu můžeme zařadit fakt, že nepočítá s existující podnikovou architekturou v tom smyslu, že nemá ve svých předpokladech, že již nějaké modely existují a nevnucuje jejich vznik. Tedy řeší jen definovaný rozsah projektu. Nabízené techniky jsou zaměřeny zejména na proces jako takový. Roger Sessions na svém webu píše: „TOGAF nám říká, jak dělat podnikovou architekturu, ale ne, jak dělat dobrou architekturu“ [A Comparison of the Top Four Enterprise-Architecture Methodologies, 2007].

3.1.3 FEA (Federal Enterprise Architecture)

Jde o pracovní rámec podnikové architektury federální vlády Spojených států amerických. Řeší obecné přístupy k integraci strategie, obchodu a technologii managementu jako součásti podniku, který je průběžně zlepšován, jak v rovině návrhu, tak v rovině jeho výkonu.

Hlavním cílem metodiky je propagace aktuální politiky a cílů managementu federální vlády Spojených států tak, aby zajistila hospodárnost, tedy omezila plýtvání, duplikace služeb a zvýšila jejich sdílení. Cílovými čtenáři FEA metodiky jsou zaměstnanci federální vlády, kteří plánují, schvalují a vykonávají agenturní programy spojené s podnikovou architekturou. Uvnitř federální vlády je 300 organizačních entit různých velikostí, zaměření a složitosti, co do komplexnosti struktur oddělení, kanceláří, komisí apod. Na první pohled si možná říkáte, k čemu je metodika federální vlády? Metodiku si představíme detailněji dále. Zjistíme, že díky různým úrovním pohledů se touto metodikou může inspirovat i soukromý sektor velkých či nadnárodních firem s více divizemi.

3.1.3.1 Základní koncept

Federální podniková architektura poskytuje na své nejobecnější úrovni schématický pohled na principy a standardy pro obchod, informace a technologie architektury, které by měly být vyvíjeny a uplatňovány prostřednictvím federální vlády tak, aby byly používány konzistentně na úrovních zadání a detailu jak mezi federálními organizacemi, tak i soukromým sektorem. Zmíněný přístup by se měl použít jako schéma integračních bodů pro oblast strategického, investičního a projektového plánování, personální politiky a informační bezpečnosti.

Základní návrh přístupů tvoří následující předměty: primární cíle, úroveň rozsahu zadání (co vše a v jaké úrovni detailu řešit), základní elementy, jednotlivé architektonické domény, referenční model, současný a budoucí pohled, plány přechodových fází a plán projektu. Když se model implementuje, zmíněná standardizace poskytne porovnatelnou architekturu prostřednictvím federálních institucí. Díky tomu je mnohem jednodušší management změn, snižuje se čas na implementaci, náklady, jsou zřejmé standardy a zodpovědnosti.

3.1.3.2 Primární cíle

Metodika míří na čtyři základní výsledky, které v sobě obsahují z praxe nejdůležitější cíle, které by se měly vždy ve všech federálních projektech dodržovat.

Zajištění služeb

Funkční integrace

Optimalizace zdrojů

Podložené reference

Zajištění služeb

Federální úřady vykonávají různé poslání, projekty a veřejné sužby. Stále víc tyto úkoly jednotlivých úřadů vyžadují spojené úsilí managementu a výkonných částí více federálních úřadů v rámci jednoho projektu a veřejné služby. To vše přes sdílené IT služby, strategie na jejich rozvoj a různě zabudované informačně technologie. Úspěch při plnění poslání agentur a optimalizaci zdrojů vyžaduje koherentní a konzistentní porozumění projektům, agilním procesům, plánování a rozvoji. Pohled z roviny soudržnosti nabývá na významu v prostředí s omezenými zdroji. Podniková architektura zajistí, že IT umožní podniku a jeho poslání dosáhnout optimálních výkonů.

Funkční integrace

Významově jde o interoperabilitu mezi projekty, systémy a službami, které pro úspěšné ukončení vyžadují metakontext a standardy. Podniková architektura poskytuje oboje. Metakontext pro všechny funkční domény – strategie, obchod a technologie, stejně tak standardy pro celý životní cyklus aktivit pro každou doménu. Interoperabilita je základem organizace federálních úřadů, díky tomu jsou schopny být úspěšnými partnery v případě nových sdílených služeb v rámci externalizace s venkovními poskytovateli, konzumenty nebo vývojáři. Podniková architektura dodává kontext a je zdrojem standardů pro všechny úrovně interoperability.

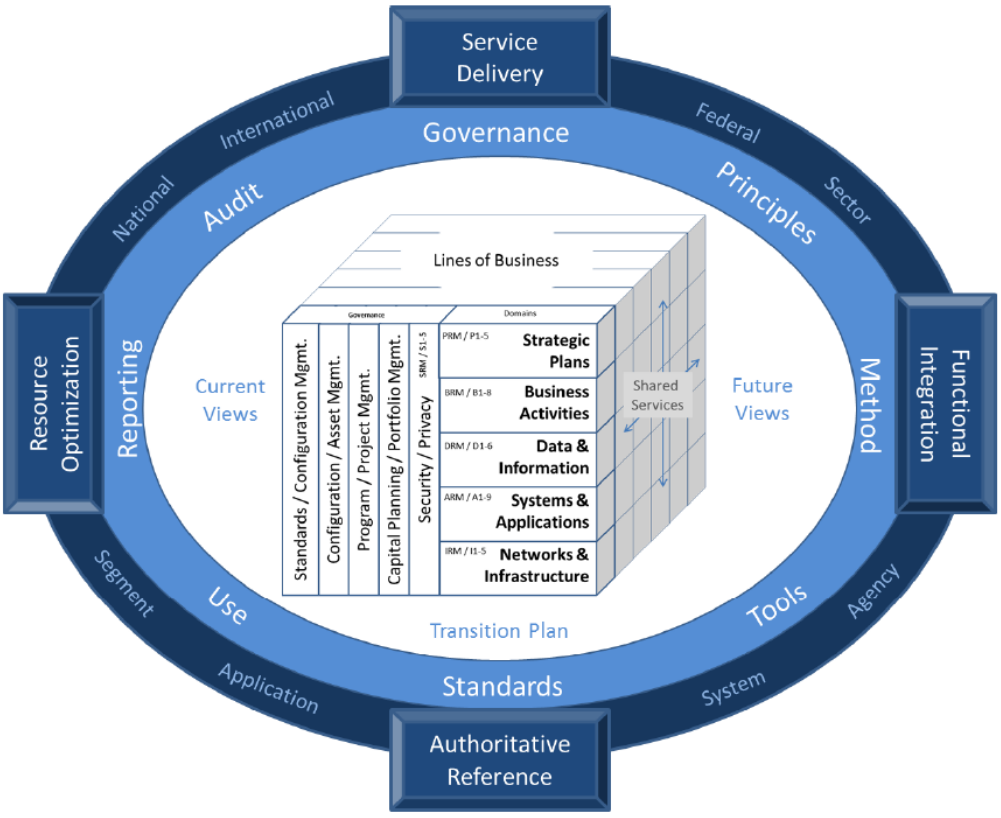
Optimalizace zdrojů

Jakožto správce veřejných fondů má sektor federální vlády speciální zodpovědnost za optimalizaci svých zdrojů. Navíc kvůli mnoha faktorům (nové zákony, politika, regulace, nové technologie, přírodní pohromy atd.) musí často federální úřady dokončit projekty s méně zdroji, než počítali. Průběžně musí být validovány cíle projektů, zda i po přechodu na nižší úroveň detailů sledujeme původní cíle, zdroje a soudržnost s organizací.

Management aktiv (hardware, software) a konfigurační management (dokumentace, kdo a jak co používá, popis procesů apod.) jsou důležité části optimalizace zdrojů. Stejně tak pro tuto sekci patří uplatnění podnikové architektury jako uvaděče nových technologií a provozních paradigmat, která optimalizují zdroje jako „cloudcomputing“, virtualizace, sémantický web, mobilní technologie, datové sklady, sociální sítě apod.

Podložené reference

Projekt pro stavbu domu je zaštítěn návody vytvořenými autoritou z oboru, jak má dům vypadat a fungovat určují plány a návrhy architekta. Podobně podniková architektura definuje v sekci podložené reference integrovaný, konzistentní pohled na strategické cíle, poslání podniku, jaké služby se mají podporovat, jaká data udržovat, které technologie ano a které ne apod. Když se bude moci říci, že podniková architektura je podloženou referencí podniku pro jeho návrh/změny a dokumentaci – systémy, služby, spory o vlastnictví zodpovědnosti v organizaci, zdroje výkonnostní cíle apod. se budou řešit víc konzistentně a efektivně [THE COMMON APPROACH TO FEDERAL ENTERPRISE ARCHITECTURE, 2012].



Obrázek 5 FEA hlavní schéma architektonického přístupu. Zdroj: [The Common Approach to Federal Enterprise Architecture, 2012]

3.1.3.3 Úroveň záběru

FEA zná osm úrovní pro implementaci architektury. Uvedené úrovně záběru poskytují jednotnost v metodikách architektury, umožňují porovnání a kontrolu napříč úrovněmi. Pohled na různé úrovně umožňuje zachytit změny od obecného pohledu až po detailní změny v poslední nejníže položené organizační složce federální vlády, stejně tak jako uvnitř jednotlivých složek nebo mezi několika složkami či vůči jejich okolí. Uvědomme si, že tato metodika je stavěna pro podnikovou architekturu „širokého“ typu, pro celý záběr federálních agentur a podobných složek vlády, pro použití v různých kontextech a detailech. Je tvořena tak, aby postihla strategické cíle, obchodní a technologický pohled.

Mezinárodní úroveň

Tato úroveň je zaměřena na mezinárodní partnerství federální vlády Spojených států amerických a jiných vlád, nadnárodních firem, nevládních organizací a ostatních skupin na mezinárodní úrovni. Často se soustředí na zpřístupnění a sdílení služeb na mezinárodní úrovni, kde je potřeba zanalyzovat obchodní model, definovat role poskytovatelů a konzumentů apod.

Národní úroveň

Obsahuje pohled na federální, státní, místní vlády uvnitř USA a jejich teritorií. Tato vrstva podnikové architektury je velice důležitá pro koordinaci a prověřování proveditelnosti jednotlivých projektů. Například jde o řešení oblastí koordinačních, varování v případě živelných nebezpečí, telekomunikace a dopravní infrastruktury.

Federální úroveň

Tato úroveň se soustřeďuje na služby a asociované systémy, které slouží celé výkonné větvi federální vlády. Jde o úroveň, na které se definuje poslání jednotlivých projektů, zajišťuje jejich podporu skrz kanály řízené dle návrhu kanceláře pro řízení a rozpočet (v originále OMB-designated; OMB – Office of Management and Budget), poskytuje tzv. „obchodní linii“.

Úroveň sektorů

Architektura se zde zaměřuje na jeden konkrétní sektor s jasně daným posláním, jde o jím používané systémy a služby. Často jde o zprovoznění aktivit mezi více částmi federální vlády, aby sdílely své poslání a služby. Tato sekce může obsahovat i napojení na soukromý sektor [Federal Enterprise Architecture Framework, 2013].

Úroveň agentur

Tato úroveň architektury dává celkový přehled o divizích/odděleních a agenturách státní správy, tak aby byl celistvý dle jednotné podnikové architektury. Popis na této úrovni, jde až na základní popis systémů, informačních sítí, popis jejich poslání a služby, které podporují. Hloubka dokumentace je odvozena od toho, jak moc se konkrétní agentura vyskytuje ve strategických plánech federální vlády.

Úroveň segmentu

Jde o úroveň konkrétních služeb nebo obchodních jednotek, které nespadají do záběru federální, sektorové nebo agenturní úrovně. Každý segment je definován organizačně nebo funkcionalitou.

Systémová úroveň

Tato úroveň architektury řeší konkrétní technologie systémů, které podporují služby uvnitř segmentů a agentur. Dokumentace by měla obsahovat důvody existence systému, obchodní požadavky na něj, aplikované standardy, popis infrastruktury, vzdálený přístup, popis procesů, výměnu informací, použitý software a zajištění informační bezpečnosti.

Úroveň aplikací

Tato část popisuje konkrétní aplikaci z pohledu vývoje, aktualizací a integrace v rámci systémů a/nebo služeb. Typickou oblastí jsou webové stránky, databáze, emailové servery [THE COMMON APPROACH TO FEDERAL ENTERPRISE ARCHITECTURE, 2012].

Diagram níže ukazuje vztah mezi úrovní záběru, konceptem, detailem pro plánování a dokumentaci.

Architektonická úroveň	Rozsah/Záběr	Dopad	Detail	Posluchači
Mezinárodní	USA a ostatní vlády	Globální výsledky	Nízký	Všichni „stakeholdři"
Národní	USA a ostatní vlády	Národní výsledky		
Federální	Výkonná část federální vlády	Vládní výsledky		
Sektor	Víc agentur federální správy	Výsledky z pohledu poslání sektoru	Střední	Obchodní vlastníci

Agentura	Jedna agentura/organizace	Výsledky z pohledu poslání agentur		
Segment	Jedno/více obchodních jednotek	Obchodní výsledky		
Systém	Jeden/víc systémů	Funkcionalita	Vysoký	Uživatelé a vývojáři
Aplikace	Jedna/víc aplikací	Funkcionalita		

Obrázek 6 FEA úroveň záběru versus oblasti dopadu. Překlad: [The Common Approach to Federal Enterprise Architecture, 2012]

3.1.3.4 Základní elementy

Součástí každého projektu federální podnikové architektury musí být osm základních elementů, které zajistí, že bude projekt kompletní, efektivní v návrhu a způsobu řešení a stejně tak budou dostatečné podklady pro plánování a rozhodování.

Vláda/řízení

První element odkazuje k plánování, rozhodování a dohledu nad procesem a oblastmi podnikové architektury typu: vývoj, schvalování, verzování a výsledného užívání. V čase projektu probíhají průběžné kontroly zaměřené na sledování kompletnosti, konzistence, souladu a přesnosti z perspektivy všech stakeholderů. V podstatě jde o projektové řízení s reportingem a schvalováním vedením podniku. V případě federální architektury jde i o soulad mezi federálními předpisy, agenturními procesy atd. Stejně tak se i v soukromé sféře se musí dodržovat platná legislativa a požadavky zadavatele.

Principy

Jde o skupinu základních pravidel, podle kterých by měly být zvažovány potencionální investice a architektonická rozhodnutí. Níže si jednotlivá pravidla představíme.

Připravenost na budoucnost – podniková architektura vybírá podle nejlepších zkušeností taková řešení, která v čase odolají změnám a budou naplňovat poslání a schopnosti jednotlivých úřadů.

Investiční podpora v interních i mezinárodních rozhodnutích ohledně architektury řešených projektů, zda do nich opravdu investovat, jak je implementovat a v jakých krocích. Agentury musí prokázat, že postupovaly dle architekturních doporučení, projekt

bude mít patřičné výsledky, využije doporučené technologie, stejně tak efektivně využije zdroje na realizaci atd.

Sdílení služeb – agentury mohou vybírat ze znovupoužitelných a sdílených služeb. K tomu jim pomáhá standardizace a pomoc od orgánů centrální vlády pro podnikovou architekturu.

Provozní standardy – podniková architektura poskytuje standardy strategického řízení a technologických možností, zároveň se snaží, aby bylo pole standardů co nejvíce vyhovující celé plejádě agentur federální vlády.

Svobodný přístup k informacím – zajišťuje se transparentní přístup a zřejmý způsob, jak budou nové služby a řešení uvedeny do provozu pro všechny vrstvy zainteresovaných osob, od občanů, soukromé sféry, vládní orgány apod. Současně se však musí striktně rozlišovat přístup k veřejným informacím a k těm neveřejným.

Zabezpečení a soukromí – zajištění vládních informací proti neautorizovanému přístupu. Podniková architektura federální vlády používá principy dle „Privacy act“ z roku 1974.

Technologické osvojení – vybrat osvědčené technologie se zajištěnou flexibilitou, jasnou a dlouhodobou podporou.

Metody

V nejúspěšnější podobě je podniková architektura používána organizací k umožnění konzistentního plánování a rozhodování. Nejde o jednoduché rozhodování v rámci jedné malé organizace. V dnešní době agentury vyžadují efektivnější použití řešení a služeb. Organizace vyžadují celou škálu postupů a standardů, které pomohou znásobit úsilí na řízení federální, národní a lokální úrovně, tak aby měly požadované výsledky a zároveň byly efektivní.

Role podnikových architektů je pomáhat vést, organizovat a podporovat obecné potřeby podnikové architektury, formulovat doporučení, aby tyto potřeby mohly být naplněny, organizovat tvorbu plánu projektů pro integraci na vyšších úrovních a dávat podporu v celé řadě disciplín - jako je zajištění plánu pro bezpečnost systémů, infrastruktury, výkonnost systémů apod. Také mají důležitou roli ohledně měření aktivit projektu. FEA obsahuje vnitřní metodiku CPM (The Collaborative Planning Methodology), která se dělí na dvě fáze: Organizace a plánování a Implementace a měření.

V té první je architekt spíše jen „facilitátorem“ a podporou projektovému manažerovi a zadavatelům, v druhé fázi má silnější roli při dohledu a hodnocení správnosti.

Nástroje

Je mnoho různých softwarových nástrojů na podporu podnikové architektury a tvorby její dokumentace, FEA má na ně tyto požadavky:

- Sdílená repozitoř a možnost obsah vizuálně reprezentovat
- Dekompozice pohledů celkové a specifické architektury
- Manažerský pohled
- Strategické plánování produktů a výkonnostní měření
- Dokumentace k obchodním procesům, v detailu odpovídajícím otázkám kolem řešení implementačních problémů
- Návrh fyzických a logických datových entit, objektů, aplikací a systémů
- Návrh fyzických a logických modelů sítí a „cloud“ řešení
- Odkazy k analýzám a reportům jednotlivých aplikací a databází
- Odkazy do portfolia investic a aktiv
- Řešení bezpečnosti a rizik pro fyzickou, informační, osobní a provozní rovinu

Nástroj, který si agentura pro své potřeby vybere, by neměl být jen vývojovým nástrojem a úložištěm dokumentace, ale musí být centrálním úložištěm dat pro informační podporu ohledně plánování a rozhodování o podnikové architektuře.

Standardy

Architektonické standardy jsou použity ve všech oblastech praxe podnikové architektury a jsou základem pro dosažení provozu mezi agenturami federální vlády a optimalizací zdrojů. Bez standardů by modely a analýzy vypadaly rozdílně, nebyla by možnost srovnávat různé varianty řešení, stejně tak by nebylo možné identifikovat stejné služby v různých organizacích apod. Vybrané standardy by měly být od amerických nebo celosvětových autorit: National Institute of Science and Technology (NIST), the Institute of Eletrical and Electronics Engineers (IEEE), the International Enterprise for Standardization (ISO) a European Committee on Standardization (CEN).

Důležitou součástí standardizace jsou jednotlivé artefakty podnikové architektury, typicky jde o modely a dokumentaci popisující část nebo celou architekturu. Na vysoké úrovni pohledu jsou často užívány textové dokumenty, nebo diagramy popisující celkovou

strategii projektu a předpokládané výsledky. Střední úroveň tvoří dokumenty, diagramy, grafy, tabulky a prezentace popisující organizaci, procesy, systémy, sítě apod. Nízká úroveň pohledu se věnuje specifickým systémům, aplikačním zdrojům, specifikaci rozhraní, datovým slovníkům, technickým a bezpečnostním standardům. Pokud jsou artefakty sladěny a integrovány za pomoci podnikové taxonomie celého rámce federální podnikové architektury, dostaneme ještě více užitečné pohledy na architekturu. Jedna z velkých výhod je i dokumentační proces jako takový, který jde se standardizací ruku v ruce.

Použití

Podniková architektura přináší hodnoty jak v rovině procesu, tak v rovině výsledného produktu (kvalitní/úspěšný projekt). Architektonický projekt poskytuje zaměření na poslání podniku nebo podporuje organizace a výsledné analytické a návrhářské aktivity, čím zvyšuje jejich efektivitu. Pouze řešená architektura může poskytnout celistvý pohled na strategii, podnik a technologické domény, na všech úrovních podniku, služeb a systémů. To je klíčem k optimalizaci zdrojů za stálého plnění původních cílů a poslání. V současnosti není jiná manažerská praxe nežli podniková architektura, která by dávala stejný kontext pro plánování a řízení změn v podniku.

Reporting

Zajišťuje informovanost o aktuálním stavu projektu a možných scénářích do budoucna. Protože repozitoř architektonických artefaktů má mnoho artefaktů a pohledů, statický pohled nestačí. Je důležité průběžně informovat o stavu projektu, zda jdeme dle plánu či nikoliv, případně jak jde řešit náprava. Je potřeba psát pravidelné reporty o stavu proveditelnosti projektu, o celkovém pokroku a jeho „zdraví“. Primárním standardem na poli reportů jsou: roční plán a sbírka odkazovaných modelů (různé úrovně pohledů na danou oblast, obsahuje taxonomii na FEA standardy pojmenování).

Audit

Kontrola architektury je důležitá k zajištění kvality, celistvosti a zvýšení pravděpodobnosti, že projekt bude úplný a včas. Periodické audity je potřeba vykonávat za účasti interních i externích expertů se zaměřením, zda bylo použito vhodných metod, zda jsou poskytované informace v projektu přesné, zda je projekt opravdu hodnotný a přínosný pro organizaci [Federal Enterprise Architecture Framework, 2013].

3.1.3.5 Dokumentace

Federální podniková architektura rozeznává šest základních domén pro dokumentaci, typově odlišné oblasti s různými analýzami a modely. Níže je tabulka se základní sadou artefaktů pro každou z oblastí.

Architektonická doména	Požadovaný (hlavní) artefakt
Strategie	Konceptuální diagram
Obchodní služby	Procesní diagram
Data a informace	Logický datový model
Aplikace	Interface diagram
Infrastruktura	Síťový diagram
Bezpečnost	Kontrolní list

Obrázek 7 FEA požadované artefakty. Překlad: [The Common Approach to Federal Enterprise Architecture, 2012]

Strategie

Doména identifikující poslání, vizi a cíle existence podniku (nebo jeho části). Primární dokumentací je konceptuální přehled a identifikace, co pohání podnik, co je impulsem pro změny a jaké jsou požadované cíle změn. To vše by mělo být v tzv. Agency`s Strategic Plan. Základní otázkami je: „Za jakým účelem podnik existuje?“ a „Co chce podnik dělat a čím chce být známý?“ Dále je potřeba navázat spuštění nového projektu na cíle a předměty hodnocení. Též se musí vytvořit definice toho, co bude měřítkem úspěchu a dosažení cílů.

ID Dokumentace	Oblast: Strategie
S-1	Konceptuální (přehledový) diagram
S-2	Strategický plán
S-3	Koncept provozních scénářů
S-4	SWOT analýza
S-5	Výkonnost měřící karta

Obrázek 8 FEA doporučené artefakty pro strategii. Překlad: [The Common Approach to Federal Enterprise Architecture, 2012]

Obchodní služby

Otázky pro tuto oblast jsou obvykle: „Jaký je obchodní (provozní) plán? Je to v souladu se strategickými cíli? Pak nás zajímá: Jaké jsou obchodní jednotky? Jaké je

jejich poslání a jaké podporují služby uvnitř a mezi obchodními jednotkami? Jak se měří efektivita obchodních procesů?“ Jaké jsou standardy, bezpečnostní otázky apod.

ID dokumentace	Oblast: Obchodní služby
O-1	Procesní diagram
O-2	Provozní plán
O-3	Katalog obchodních služeb
O-4	Organizační diagram
O-5	Popis a diagram užití
O-6	Obchodní případy / alternativní analýza

Obrázek 9 FEA doporučené artefakty pro obchodní služby. Překlad: [The Common Approach to Federal Enterprise Architecture, 2012]

Data a informace

Oblast se logicky soustředí na dotazy ohledně zpracování informací a dat ve firmě. Mezi důležité dotazy patří: „Které toky informací služby potřebují, aby byly úspěšné? Jak můžeme toky zlepšit, aby byly služby efektivnější, bezchybnější či víc zabezpečené? Jak jsou přenosy a toky řešeny po stránce formátu, generování, ukládání a sdílení dat? Jaké jsou typy pracovní síly, použité standardy a bezpečnost ohledně provozu a správy dat a informací?“

ID dokumentace	Oblast: Data a informace
D-1	Logický datový model
D-2	Znalostně-manažerský plán
D-3	Plán datové kvality
D-4	Plán datových toků
D-5	Fyzický datový model
D-6	CRUD matice *Create, Read, Update, Delete
D-7	Stavový diagram
D-8	Sekvenční diagram
D-9	Datový slovník
D-10	Knihovna objektů

Obrázek 10 FEA doporučené artefakty pro data a informace. Překlad: [The Common Approach to Federal Enterprise Architecture, 2012]

Aplikace

Pro tuto oblast analýzy a následné dokumentace jsou důležité otázky typu: „Jaké systémy a aplikace potřebujete, abyste vytvářeli, sdíleli a ukládali data, informace a znalosti potřebné pro obchodní služby? Kde jsou složité oblasti a kde musí spolupracovat

více systémů, služeb, aplikací, databází apod. dohromady? Jak můžeme nastavit IT provoz, tzv. helpdesk a podobná oddělení, tak aby byly náklady co nejefektivněji využity? Jaké jsou typy pracovní síly, použité standardy a bezpečnost ohledně provozu a správy aplikací?“

ID dokumentace	Oblast: Aplikace
A-1	Interface diagram
A-2	Komunikační diagram
A-3	Matice interfaců
A-4	Matice výměny dat
A-5	Matice služeb
A-6	Výkonnostní matice
A-7	Systém/aplikační vývojový diagram
A-8	Integrační diagram služeb
A-9	Provozní příručky
A-10	Knihovna aplikací
A-11	Knihovna licencí software

Obrázek 11 FEA doporučené artefakty pro aplikace. Překlad: [The Common Approach to Federal Enterprise Architecture, 2012]

Infrastruktura

V rámci této oblasti je záhodno využít otázky: „Které hlasové, datové, mobilní a video sítě vyžadují systémy/aplikace a jaké typy protokolů, dat a konverzí vyžadují? Jaká je nutná fyzická infrastruktura sítí (včetně věcí jako serverovny apod.)? Je nutná vysoká škálovatelnost řešení, potřebujeme používat tzv. cloud, jaký je vztah na poskytovatele a konzumenty služeb? Jaké jsou typy pracovní síly, použité standardy a bezpečnost ohledně infrastruktury?“

ID dokumentace	Oblast: Infrastruktura
I-1	Síťový diagram
I-2	Provozní koncept "hostingu"
I-3	Technické standardy profilů
I-4	Technologická předpověď
I-5	Topologie sítí
I-6	Topologie bezdrátových sítí
I-7	Schéma místností datových center
I-8	Integrační diagram služeb
I-9	Detailní diagramy zapojení sítě
I-10	Diagram připojovacích bodů

I-11	Seznam hardware/IT infrastrukturních prvků
I-12	Příručky

Obrázek 12 FEA doporučené artefakty pro infrastrukturu. Překlad: [The Common Approach to Federal Enterprise Architecture, 2012]

Bezpečnost

Oblast je zaměřena na bezpečnost a zajištění soukromí zpracovávaných informací. S cílem zabudovat tyto prvky ještě s větší efektivitou do informačních toků, aplikací a datových sítí.

ID dokumentace	Oblast: Bezpečnost
B-1	Katalog bezpečnostních kontrol
B-2	Bezpečnostní plány
B-3	Certifikace a akreditační dokumentace
B-4	Dohledové postupy
B-5	Záložní plány pro případy katastrof
B-6	Provozní plány zachování kontinuity

Obrázek 13 FEA doporučené artefakty pro bezpečnost. Překlad: [The Common Approach to Federal Enterprise Architecture, 2012]

3.1.3.6 Referenční modely

FEA obsahuje doporučení vytvářet šest referenčních modelů, a to pro oblasti: výkonnost, obchod, data, aplikace, infrastruktura a bezpečnost. Každý z modelů obsahuje svou vlastní taxonomii, metody tvorby a případy užití.

Výkonnostní referenční model podporuje architektonickou analýzu a reporting ohledně oblasti strategie a celkového pohledu na podnikovou architekturu. Model určuje kategorizaci a standardní metody pro měření výkonu. Umožňuje na úrovni jednotlivých větví výkonných částí federální vlády měřit úspěšnost investic a plnění očekávaných cílů. Hlavním cílem modelu je podpora v plánování rozvoje a být podkladem pro důležitá rozhodnutí. Model zlepšuje informovanost o výkonnostech jednotlivých částí podniku, snižuje entropii pro strategické i každodenní rozhodování. Zároveň umožňuje identifikovat příležitosti pro zlepšení určitých částí organizace.

Obchodní referenční model pomáhá celkovému pohledu na oblast obchodních služeb, zlepšuje analýzu a reporting. Umožňuje navázat obchodní činnosti na vnitřní obchodní služby, stejně tak služby mezi jednotlivými federálními úřady. Jde více o pohled

zaměřený na funkčnost nežli na organizaci federálních jednotek. Obchodní model umožňuje ukázat, který útvar nebo agentura by měl být hlavním pro poskytování konkrétních služeb a který nikoliv.

Datový referenční model se zaměřuje na použití dat a informací. V jednoduchosti jde o dvě základní otázky: „Které informace jsou dostupné pro sdílení a znovupoužití? Jaké informace jsou nedostatečné a potřebují úpravy?“ Datový model je navržen k poskytnutí informací jak efektivně, flexibilně sdílet informace, zvýšit jejich zapojení v rámci odpovídajícího množství federálních úřadů a zároveň dodržovat pravidla bezpečnosti a neporušovat zásady ochrany osobních údajů. Model používání dat umožňuje dát odpovědi na otázky, které by bez něj nemusely být zřejmé. Především jak a v jaké části organizace s daty pracujeme, kdo a jak je používá a mohl by je používat tak, aby se efektivně naplnila poslání jednotlivých částí federální správy.

Aplikační referenční model se řídí jasně definovanou taxonomií pro kategorizaci jednotlivých komponent se zohledněním standardů a technologií, tak aby podpořil celkový pohled na aplikace. Napomáhá rozpoznat, které služby využívají stejné komponenty (části nebo celé aplikace). Díky celkové představě o aplikacích a jejich vlastnostech se pak výrazně lépe vyhodnocují možnosti standardizací technologií, jednotných provozních postupů apod. To umožňuje zaměření na úspory, zvyšování rychlosti a znuvupoužitelnosti aplikací.

Infrastrukturní referenční model se zaměřuje na celkový pohled, kde a v jaké infrastruktuře jsou provozovány jednotlivé aplikace. Stejně jako s ostatními modely je s ním spojena i svébytná kategorizace, třídící sítě a infrastruktury komunikačních platforem do kategorií. Jejich cílem je mít přehled o možnosti regulování propustností sítí, zajistit standardy pro konfigurace sítí a jednotlivých počítačů atd.

Bezpečnostní referenční model popisuje základní metodiky řešení bezpečnostních rizik jako specifické řízení informační bezpečnosti, ochrana osobních údajů, a to v rámci systémů, konkrétních oblastí služeb, agentur a sektorů federální vlády. Pomocí tohoto modelu se definují plány, kdy a kde se mají konkrétní bezpečnostní pravidla/standardy zavést. Poskytuje mechanizmy, jak identifikovat bezpečnostní požadavky. Podporuje zavádění a zlepšování informační bezpečnosti a ochrany osobních údajů [Federal Enterprise Architecture Framework, 2013].

3.1.3.7 Plánování

Federální podniková architektura říká jaké plány pro projekty připravit. Doporučuje několik pohledů, které zachytí jak harmonogram rozvoje podniku (agentura, část federální organizační struktury apod.), tak plán pro přechodové období, současný pohled na podnik a budoucí pohled na podnik. Tyto čtyři pohledy zaručí, že nebude opomenuto žádné hledisko.

3.1.3.8 Harmonogram rozvoje podnikové architektury

Harmonogram dokumentuje a váže strategické cíle podniku na obchodní služby, integrované technologie přes úroveň jednotlivých organizačních jednotek. Jde o zhmotnění strategických cílů na rozvoj organizací federální vlády v horizontu několika let. Harmonogram pojednává o celkové koncepci naplnění strategie s načasováním a jasnými dopady realizace do jednotlivých oblastí federální vlády. Obsahuje a identifikuje cíle a problémy, které chce řešit, požadavky na zdroje (peníze, lidské kapacity a čas pro naplnění strategie), dokumentuje zamýšlená řešení. Harmonogram není statickým dokumentem, musí se počítat s jeho aktualizacemi. V pravidelných intervalech (minimálně na roční bázi) musí být aktualizován, a to na všech úrovních. Řízení požadavků na úpravy musí být zřejmé. Změny musí být verzované a obsah harmonogramu musí být neustále přístupný v rámci repozitoře podnikové architektury.

Tento rozvojový plán má i silně manažerskou úlohu ohledně plánování rozpočtu a jeho schválení. Jde o základní podklad, který říká, kolik bude stát naplnění konkrétní strategie, a management se na základě tohoto plánu na strategické úrovni rozhoduje, zda uvolní peníze na realizaci.

Podniková architektura federální vlády důkladně popisuje i jednotlivé nezbytné doplňky harmonogramu:

Řízení a rozhodování

Jedná se o sekci, kde musí být co nejlépe popsán řídící a rozhodovací proces rozvoje architektury. Udává politiku projektu, tedy pravidla, kdo, kdy, jak má rozhodovat, prezentovat a schvalovat. Tato část harmonogramu řeší provozní detaily typu: investiční nabídky, jejich vyhodnocení, plán pravidelných řídících schůzek projektu, schvalovací pravomoc, komunikační matice a použité standardy.

Role a zodpovědnosti

Metodika též v seznamu cca 15 rolí říká, jaké role by měly být zahrnuty, na jaké jsou pozici v organizaci a jaké mají zodpovědnosti a pravomoci. Seznam začíná ředitelem agentury (ředitel podniku), který má roli sponzora (financuje rozvoj architektury), je autoritou pro rozhodování, schvaluje zdroje a podílí se na řešení architektonických problémů na vysoké úrovni detailu. Na konci seznamu je pozice manažera repozitoře podnikové architektury, který má roli správce repozitoře a jeho zodpovědností je provoz repozitoře a připojeného webu, údržba obsahu a propojení na jiné zdroje na stejné informační úrovni.

Rozpočet

Metodika říká, že by měl být řešen na fiskální rok, na celý životní cyklus a stejně tak z pohledu celkových nákladů budoucího vlastnictví výsledků projektu. Doporučení je počítat celkové náklady v horizontu pěti let. Náklady obecně obsahují: nastartování programu podnikové architektury, provoz, platy, místa a vybavení, kde bude tým pracovat, tvorba úvodní dokumentace podnikové architektury dané oblasti federální správy, zakoupení nástrojů pro podnikovou architekturu, repozitoř a její údržba. Takto stanovené náklady, by měly být základem pro rozpočet. Ten bude současně tvořit základní měřítko finanční efektivity. Vůči němu se bude v budoucnu porovnávat naplňování finančních kritérií pro podnikovou architekturu.

Měření přínosu podnikové architektury

Měří se dvě oblasti výsledky a výstupy. Ohledně výsledků se porovnává zlepšení mezi novým a stávajícím stavem, například větší integrace komponent a aplikací, vyšší spokojenost uživatelů nebo větší efektivita IT investic. Ohledně výstupů měříme, kolik poskytujeme dat a v jakých aktivitách. Měřené oblasti mohou být: počet databází, počet odeslaných e-mailů v rámci jednoho dne nebo vyhodnocování IT projektů, zda splnily odhady ceny, časové plány nebo očekávanou výkonnost. Důležité je měření pokroků projektů a obecné zlepšování IT oblastí. Nejdůležitější je však korelace výsledků měření v souladu s definovanými cíli. To znamená, že zkvalitňujeme měřené oblasti a současně naplňujeme záměry jednotlivých projektů [Federal Enterprise Architecture Framework, 2013].

3.1.3.9 Současný stav

Jedním z úkolů podnikové architektury je zformulování a shrnutí současné architektury podniku do pochopitelné a standardizované podoby. Popis současného stavu ukazuje celkový pohled na propojení mezi službami a zdroji v každé z oblastí podnikové architektury. Je to cesta, jak prezentovat roli IT uvnitř podniku tak, aby byla lépe chápána pro budoucí analýzy z různých perspektiv. Zmíněné informace nejsou duplicitní vůči případným již existujícím informacím v architektonické repozitoři, ale měly by poskytovat již zmíněný ucelený objektivní pohled na současné aktivity a podporovaná technologická řešení v podniku. Dále jsou popsány dokumenty, které popisují současný stav.

Strategické cíle a iniciativy

Tato část identifikuje, jak podniková architektura a specifické zdroje podniku podporují agenturní strategické cíle a iniciativy. Taktéž tvoří popis k strategickým plánům, díky čemuž jasněji ukazují napojení obchodních aktivit na IT zdroje, a je zřejmé, které oblasti jsou novými projekty zasaženy.

Obchodní služby a toky informací

Sekce řeší podporu analýzy obchodních procesů a jejich zlepšování, stejně tak identifikuje optimalizace informačních toků mezi procesy. Výsledkem by měly být dokumenty a modely, které ukazují, které IT zdroje podporují poslání a obchodní služby / procesy podniku. Především můžeme využít modely informačních toků a datových struktur (ale na obecné úrovni).

Aplikace

Jde o oblast identifikující současné artefakty na aplikační úrovni podnikové architektury s napojením na informační toky a požadované projektové aktivity dané agentury. Z dokumentu by mělo být zřejmé, zda jsou vhodné k naplnění poslání dané části podniku a umožní splnit strategické cíle. Rozsah může být pojat v různém měřítku dle potřeby, od popisu velkých komplikovaných systémů na obecné úrovni po popis malých aplikací na míru ve velkém detailu. Slovní popis by měl být zaměřen na možný stupeň integrace, škálovatelnost, naplnění uživatelských očekávání, závislost na proprietárních nebo dodavatelských řešeních.

Infrastruktura

Zde se zaměřujeme na pohled ohledně hlasové, datové, video a mobilní infrastruktury. Z této sekce by mělo být zřejmé, které sítě jsou interní a které externí, zda

není některá infrastruktura duplicitní. Též by mělo být zřejmé napojení na služby typu email, IT podpora apod.

Informační bezpečnost

Oblast se zabývá IT a informační bezpečností, společně s ochranou osobních údajů na všech úrovních pohledu podnikové architektury. Bezpečnost musí být součástí strategie, jelikož naplnění cílů závisí na tématech úplnosti a autentičnosti předávaných informací, stejně tak jako autentifikaci přístupů k datům a další.

Standardy

Jde o oblast dokumentů, které sdružují standardy jak obchodní, tak technologické pro aplikace, infrastrukturu, hlasové, datové, mobilní formáty až po bezpečnostní standardy. Do řešené oblasti patří seznamy doporučených dodavatelů, produktů a standardy, které by se měly upřednostňovat. FEA doporučuje tyto federální a celosvětové organizace: the Institute of Electrical and Electronics Engineers (IEEE), the National Institute of Science and Technology (NIST), the International Enterprise for Standardization (ISO), the European Committee on Standardization (CEN), the Object Management Group (OMG), federální zákony a metodiky federální podnikové architektury [Federal Enterprise Architecture Framework, 2013].

Požadavky na pracovníky

Tato část popisuje požadavky na změny ohledně znalostí a zkušeností pracovníků. Lidé jsou často nejhodnotnější součástí podniku, proto je nutné dopředu naplánovat konkrétní rozvojová školení a správně popsat konkrétní školení, která vyžadují změny v naplnění poslání podniku.

3.1.3.10 Budoucí architektura

Měla by být založena na několika alternativách provozních scénářů a také na uvědomění, že není možné u žádné z federálních agentur přesně předpovědět vývoj okolností na několik let dopředu. Proto bychom měli být připraveni na možné základní scénáře vývoje pro danou oblast.

Budoucí operační scénáře

Scénáře by měly v rámci prezentace reflektovat popis účelu jednotlivých scénářů a celé spektrum provozních prostředků, na kterých budou provozovány. Měly by být připraveny tři scénáře popsány níže:

Scénář 1: Pokračování za nezměněných okolností

Scénář 2: Rozšíření strategie pokud budou dostupné zdroje

Scénář 3: Defenzivní strategie se sníženým rozpočtem

Každý scénář má v hrubých rysech v sobě zahrnuty i předpoklady nutných změn z oblasti procesu, lidí a technologií.

Plánované předpoklady

Řešená oblast identifikuje nové schopnosti a zdroje, které jsou potřeba pro úspěšné dokončení každé varianty scénáře. Tyto podklady dobře poslouží i pro představu, zda je daný scénář v souladu s celkovou strategií dané agentury.

Aktualizace současného a budoucího pohledu

Dokumentace plánovaných změn by měla mít vliv na všechny úrovně architektonického rámce. Do všech úrovní popisu by se měly průběžně přenášet změny. Ať ty realizované nebo nově plánované dle strategických cílů. Impulsem jsou změny ve strategii, která je na periodickém základu nebo důležité změny v interních a externích procesech. Aktualizace je důležitá pro udržení aktuálních informací. Měli bychom u realizovaných projektů mít spolehnutí na informace ohledně možnosti sdílení služeb, infrastruktury apod. Stejnou měrou slouží jako podklad pro validaci vůči aktuálním plánům.

Modernizace

Sekce zastřešuje dokumentaci pro přechodovou fázi mezi současným stavem a tím budoucím. Obsahuje dokumenty: kolovník projektu, hlavní milníky, časový harmonogram pro implementaci nových systémů a služeb. Velké a středně velké agentury často mají mnoho nových vývojových, aktualizačních, migračních a konzervačních (zrušení systémů apod.) projektů. Tato sekce jim dává doporučení, co nezapomenout, co vést v patrnosti. Taktéž dokumentace dává celkovou přidanou informační hodnotu celé federální architektuře, která je díky tomu informačně aktuální.

Konfigurační management

Jde o sekci, která se zabývá již velkým detailem zaměřeným na podprocesy a jejich konfiguraci a standardy. Z pohledu podnikové architektury se zachycují změny typu přidání, zrušení a aktualizace aplikace, systému nebo služeb. Konfigurační management

zajišťuje dodržení standardu pro proces změny, stejně tak technické standardy, verzování a kontroly.

Inventář aktiv informačních technologií

Harmonogram podnikové architektury obsahuje inventář všech agenturních IT aplikací, systémů a služeb [Federal Enterprise Architecture Framework, 2013].

3.1.3.11 Shrnutí FEA

Pracovní rámec federální podnikové architektury je primárně zaměřen na řízení a rozvoj organizací státní správy ve Spojených státech amerických. Může být však ucelenou inspirací pro lehce upravenou metodiku jakéhokoliv středního a velkého podniku. FEA důkladně mapuje pracovní postupy od projektového managementu po samotnou implementaci, stejně tak se věnuje vydefinováním jednotlivých artefaktů, kde je vysvětlen účel i okamžik, kdy by měly vzniknout.

Mezi slabé stránky řadím až přílišnou komplexnost a složitost, které můžou být v podnicích překážkou. Stejně tak udržování předepsaných artefaktů v aktuálním stavu může samotnou podnikovou architekturu výrazně prodražit.

3.2 Metodiky vývoje

Kapitola se zaměřuje na metodiky a techniky, které standardizují vývoj aplikací v rámci rozvoje podniku, ať už za pomocí projektů přinášejících nové produktu, funkcionality nebo změnových požadavků stávajících řešení. Z pohledu softwarového inženýrství se tato část věnuje metodikám, které zahrnují práci všech důležitých rolí na přípravě procesu, od zadání po implementaci, včetně nasazení do provozu.

Alena Buchalcevová definuje metodiku vývoje takto: „Metodika budování IS/ICT definuje principy, procesy, praktiky, role, techniky, nástroje a produkty používané při vývoji, údržbě a provozu informačního systému, a to jak z hlediska softwarově inženýrského, tak z hlediska řízení“ [BUCHALCEVOVÁ, 2005].

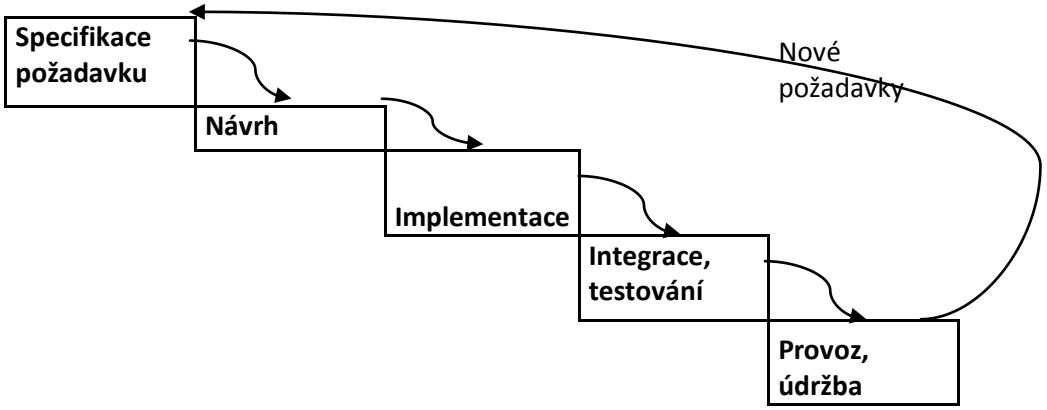
Současně rozděluje metodiky do dvou typů, rigorózní a agilní: Rigorózní charakterizuje jako metodiky, které vycházejí z přesvědčení, že procesy při budování IS/ICT lze popsat, plánovat, řídit a měřit. Snaží se podrobně a přesně definovat procesy, činnosti a vytvářené produkty. Bývají proto často velmi objemné. Rigorózní metodiky jsou zpravidla založeny na sériovém (vodopádovém) vývoji. Existují ale také rigorózní

metodiky založené na iterativním a inkrementálním vývoji. Příkladem těchto metodik jsou OPEN, Rational Unified Process (RUP), Enterprise Unified Process. Oproti tomu k agilním metodikám říká: Změny technologií a ekonomického prostředí, ke kterým v současnosti dochází, a požadavky na rychlé zavedení IS/ICT vyžadují změny v metodikách. Tradiční rigorózní metodiky přestávají v takových podmínkách vyhovovat a začínají se prosazovat metodiky, které umožňují vytvořit řešení velmi rychle a pružně jej přizpůsobovat měnícím se požadavkům. Tyto metodiky jsou označovány jako agilní. Jedná se o různé metodiky, které vznikaly od druhé poloviny 90. let a které prosazují myšlenku, že jedinou cestou, jak prověřit správnost navrženého systému, je vyvinout jej (nebo jeho část) co nejrychleji, předložit zákazníkovi a na základě zpětné vazby jej upravit. Každá z agilních metodik je svým způsobem specifická, ale všechny jsou postaveny na stejných principech a hodnotách. Proto se představitelé těchto přístupů v únoru 2001 sešli a podepsali „Manifest agilního vývoje software“ a také vytvořili „Alianci pro agilní vývoj software“ [BUCHALCEVOVÁ, 2005].

3.2.1 Vodopádový vývoj a rigorózní metodiky

Vodopádový vývoj je spíš popisem sledu hlavních aktivit v rámci životního cyklu vývoje software nežli čistou metodikou. V každém případě je vhodné si tuto historicky nejstarší a stále nejvíce používanou techniku v základu představit.

Vodopádový model byl představen v 70. letech Dr. Winstonem Roycem. Jeho model vychází z posloupnosti jednotlivých fází, od definice požadavku resp. vzniku problému / potřeby po nasazení do provozu a údržbu. Ač je model postaven na přímočarosti sledu jednotlivých aktivit, byl revoluční ve své jednoduchosti a zřejmosti na pochopení. Dává velkou sílu ohledně projektového řízení a možnosti měřitelnosti a jako první přišel s možností se v případě neúspěchu v jedné z aktivit o krok vrátit.



Obrázek 14 Základní schéma vodopádového modelu

Častá argumentace pro vodopádový vývoj vyzdvihuje možnost včasného odhalení chyb, nedomyšleného zadání apod., díky oddělení jednotlivých aktivit do samotných celků. Mezi pozitiva se také uvádí, že je správné investovat ze začátku dostatek času pro zadání a průběžnou tvorbu ucelené dokumentace pro danou aktivit vodopádu, jelikož úplnost zadání se vždy vrátí v pozdějších fázích implementace, integrace, testování a předávání uživatelům do provozu. Nedochází k nepochopení či dokonce dodávkám něčeho jiného nežli bylo požadováno. Ti, kteří preferují vodopádový model, na něm oceňují jednoduchost jeho přístupu. Připadá jim také disciplinovanější. Vodopádový model by měl poskytovat strukturovaný přístup; model postupuje lineárně, diskrétními, jednoduše pochopitelnými a vysvětlitelnými fázemi, a tak není složité mu porozumět. Poskytuje také snadno určitelné milníky ve vývojovém procesu. Možná to je tím důvodem, proč je vodopádový model uváděn mezi prvními ukázkovými příklady vývojového modelu v mnohých učebnicích a kurzech o softwarovém inženýrství [Vodopádový model, 2013].

Vodopádový model je mnohými považován za nevhodný pro praxi. Kritici jsou především přesvědčeni, že u jakéhokoli netriviálního projektu je nemožné dovést jednu fázi životního cyklu softwarového produktu k dokonalosti předtím, než se přejde k fázi následující. Klientům nemusí být například úplně jasné, jaké jsou jejich požadavky, dokud neuvidí fungující prototyp, ke kterému se pak mohou vyjádřit; své požadavky můžou neustále měnit a návrháři a implementátoři pak nad tím ztrácejí kontrolu. Studie z posledních let ukazují, že v oblasti vývoje webových aplikací, což je blízká množina našeho tématu vývoje front-endu, na první pohled upadá obliba vodopádových metodik. Monica Lam ve svém výzkumu o používání metodik při vývoji webových aplikací píše:

vodopádový životní cyklus byl použit v případě 3,5% respondentů, metodika Rational Unified Process s podobnými základy 2,6%. Nicméně když největší procento respondentů používá metodiky spadající do oblasti agilních technik 12,8%, je nutné podotknout, že kromě této hodnoty žádná metodika v daném průzkumu nepřesáhla dvouciferné číslo použití. Většina respondentů má vlastní metodiky, kde nelze z výzkumu rozpoznat, z jakých filosofií vychází. Přes menší procenta používání vodopádových technik, výzkum současně říká, že v případě velkých a prioritních projektů se týmy vyvíjející weby vracejí k metodikám se základy z vodopádu [Vodopádový model, 2013], [LAM, 2012].

3.2.2 Unified Proces

Můžeme se též setkat s názvem Unified Software Development Proces, dále v textu jen UP. Metodika filozoficky vychází z aktivit vodopádového vývoje, oproti nim však zná pojem iterací, který si dále vysvětlíme. Metodika svou povahou patří do kategorie rigorózních technik.

UP pochází od autorů UML, kde je tato metodika procesní částí a UML jazykovou. Na rozdíl od UML však není standardem podporovaný organizací OMG. Vychází z metod Ericsson (Ericsson approach, 1967), Rational (rational Objectory Process, 1996-1997). UP je otevřený standard, ke které existuje i komerční verze Rational Unified Process, kterou v roce 2003 převzala IBM. Hlavní rozdíly spočívají v otázkách úplnosti a detailu, spíš než v sémantice a ideovém obsahu.

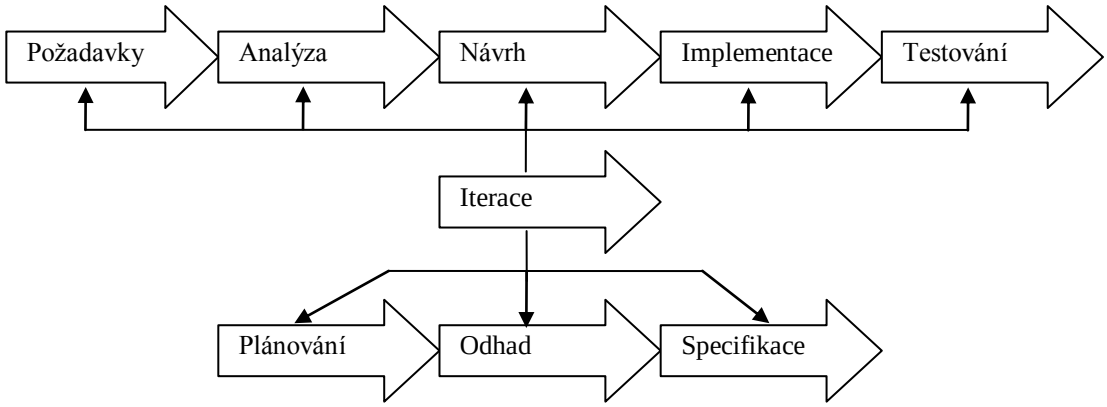
Hlavní myšlenka UP vychází z Jacobsonova názoru, že žádný vývoj software není stejný. Proto se zaměřil na čtyři plus jeden architektonický pohled, který by měl vyhovovat všem. Tento model počítá s logickým, procesním, fyzickým a vývojovým modelem, které spojuje případ užití. Společně se svým týmem přichází s iterativním vývojem. To znamená, že výsledky jednotlivých aktivit, jak je známe z vodopádu, budou dodávaný po jednotlivých částech. Iterativní vývoj definuje jako fáze: zahájení, rozpracování, konstrukce a zavedení [ARLOW, 2003].

Metodika řeší základní otázky: kdo, co a jak? Roli, kterou v projektu hraje osoba nebo tým, představuje takzvaný dělník (worker), který dává odpověď na otázku kdo. Výsledky dělníka jsou buď aktivity, nebo artefakty, které odpovídají na otázku co. Pracovní postup (workflows) je entitou a odpovědí na otázku jak.

Unified Proces má tři zásady:

- Řídit se případem užití a rizikem
- Soustředit se na architekturu
- Používat iterace a přírůstky

Požadavky uživatelů jsou přetvářeny do případů užití, které jsou pak impulsem pro jednotlivé části specifikací. V rámci návrhů se ukazují rizika spojená s realizací a ta musí validovat projektový management. Jelikož většina projektů postihuje více systémů s mnoha komponentami, je důležité zaměření na celkovou architekturu. Důkladně zanalyzovat jednotlivé komponenty, možnost samotného vývoje (a ostatních aktivit životního cyklu), tak aby na konci projektu zapadly funkčně a splněním požadavků do celku, a to i na úrovni hardware. Metodika netypicky pro vodopád používá iterace a přírůstky. Pod iteracemi si představme rozpad do menších pod-projektů, které dodávají výstupy zastřešujícímu projektu v dávkách. Současně dodávají své výstupy na základě toho, jak se upřesňují jednotlivé požadavky [ARLOW, 2003], [JACOBSON, 1999].



Obrázek 15 Pracovní postupy v iteraci. Zdroj: [ARLOW, 2003]

3.2.2.1 UP životní cyklus projektu

Každá iterace vytváří tzv. základní linii (baseline), k níž přibývá množina předem domluvených artefaktů za danou iteraci. Tato množina tvoří schválený základ pro další analýzu, vývoj nebo testy. Může být měněna jen na základě změnového managementu.

Životní cyklus projektu je rozdělen na čtyři fáze: zahájení (inception), rozpracování (elaboration), konstrukce (construction) a zavedení (transition). Fáze projektu by měla být ukončena předem definovaným hlavním milníkem. Současně v každé fázi může být libovolný počet iterací, odvislých od velikosti projektu.

Zahájení

Cílem je "zahájení projektu", fáze obsahuje dokumenty: Podmínky proveditelnosti – jde buď o psaný analytický dokument shrnující podmínky a předpoklady, za kterých lze projekt realizovat, nebo tzv. Proof of Concept, modelové části určitých technologií na vyzkoušení, že jsou dané předpoklady v podmínkách podniku proveditelné. Návrhy obchodních případů – slouží pro vyhodnocení, zda je reálná přínosnost projektu. Používá se například analýza nákladů a výnosů (Cost Benefit Analysis). Zachycení podstatných požadavků – umožňuje definovat rozsah vznikajícího projektu. Používá se například dopadová analýza (Impact Analysis). Rizika – definice kritických rizik projektu.

Hlavní role této fáze jsou projektoví manažeři, podnikoví architekti, business analytici a samozřejmě zadavatelé a vedení podniku.

Hlavní pracovní postupy se zaměřují na specifikaci zadání a jejich analýzu. Případně na vytváření modelů a funkčních konceptů.

Rozpracování

Hlavní cíle fáze rozpracování jsou: vytvoření spustitelných základů aplikací, vylepšené odhady rizik, definice měření kvality, případy užití pro 80% funkčních požadavků, vytvoření plánu pro fázi konstrukce a formulace poptávek na realizaci projektu a jeho částí.

V této fázi se zabýváme požadavky, analýzou a návrhem. Na konci fáze nabývá na důležitosti implementace funkčního základu navrhovaných aplikací. Důležité jsou tyto aktivity: Požadavky – upřesnění rozsahu požadavků na aplikace. Analýza – definice dodávaných artefaktů. Návrh – vytvoření stabilní architektury. Implementace – vybudování základních funkčních komponent nových nebo upravovaných aplikací. Testování – testování dodané implementace.

Konstrukce

Cílem fáze je splnit všechny požadavky analýzy a ze spustitelného základu předchozí fáze vyvinout konečnou verzi aplikací. Klíčovým zadáním je zachování integrity architektury vytvářených aplikací. Důležité jsou tyto aktivity: Požadavky – odhalit a popsat zbylé požadavky, které mohly být v předchozích fázích přehlédnuty. Analýza – dokončit analytický model. Návrh – dokončit model návrhu. Implementace –zajistit počáteční provozní způsobilost (Initial Operation Capability). Testování – otestovat počáteční funkční variantu. Milníkem fáze jsou připravené aplikace na počítačích uživatelů.

Zavedení

Začíná okamžikem dokončení testování a nasazením aplikací. V této fázi se opravují všechny nalezené chyby. Zavádí se nové procesní manuály pro uživatele. Aplikace se upravují na základě kritických nepředpokládaných chyb. Vytváří se vyhodnocení projektu [ARLOW, 2003].

3.2.3 Agilní metodiky a techniky

Rigorózní metodiky čelí během posledních desetiletí obviňování ze strnulosti a neschopnosti čelit změnám. Jedním z pokusů, jak se těmto tématům postavit, byl příchod agilního (angl. agile = hbitý, bystrý) přístupu, který už svou povahou předpokládá práci a reakci na změny v turbulentním obchodním prostředí tak, aby výsledky přinášeli profit. V případě agilních metodik je kladen důraz na kvality vedoucího týmu i každého jedince v něm. Vedoucí musí pružně reagovat na aktuální potřeby zadavatelů, musí znát schopnosti svých podřízených (vývojářů, analytiků a testerů) a správně jim přidělovat práci s vysokým stupněm osobní svobody a zodpovědnosti za vykonání jednotlivých úkolů. Nutno dodat, že agilní přístup se může používat ve všech útvarech podniku, jen z důvodu zaměření diplomové práce míří příklady týmů do IT. Zároveň pokud nemá IT spojence v obchodních týmech, v řadách zadavatelů požadavků, kteří používají stejný agilní přístup, je výsledek celé snahy o pružný přístup k projektům a požadavkům poloviční až nulový.

Agilní metodika je způsob rozvržení a ověřování práce. Cílem agilní metodiky bývá lepší organizace práce. Odlišné agilní metodiky vedou k odlišným produktům, protože považují jiné aspekty produktu za klíčové. Liší se i přístup ke změně zadání.

Vedoucí týmu přináší úkoly, zodpovědnost za realizaci předává týmu a na konci iterace společně vyhodnocuje. Díky tomu může řídit více lidí nebo se zaměřit na kvalitu. Historické iterace dávají odhady o schopnostech týmu pro typové úkoly. Díky tomu je v horizontu třech měsíců zřejmé, kolik úkolů a jaké složitosti tým zvládne v rámci jedné iterace. Díky rozdělení úkolů je možné měnit zadání za běhu. Zrychlí se doručení produktu od vývojáře k zákazníkovi (klientovi produkt doručujete po každé iteraci – zákazník se často iterace účastní). Zapojení zákazníků do vývoje sníží třecí plochy mezi produktem a dodavatelským týmem. V rámci agilních přístupů jsou v absolutní většině používány automatické testy, to zaručí dostatečnou kvalitu i přes přírůstky po menších funkcionalitách. Omezí se rigidní procesy ve prospěch komunikace (nejdou proti dobře

nastaveným procesům – pouze omezují ty procesy, které brání efektivitě, komunikaci a agilnímu přístupu k práci) [KNESL, 2009].

Fáze celého projektu

Nultá iterace, krátká analýza a naprogramování ukázky nebo prototypu, aby bylo co ukázat zákazníkovi. Následuje analýza změny, jde o výběr toho, co se bude implementovat a detailně analyzovat. Poté přichází implementace požadovaných vlastností. Není-li projekt hotov, vrací se do bodu analýza změny. Je-li projekt hotov, přechází se do údržby a rozvoje, který je řešen také iteracemi. Kromě nulté iterace a údržby v praxi všechny zmíněné aktivity vedeme jako jednu iteraci, většinou v trvání od čtrnácti dnů po měsíc a půl.

Analýza změny

V této fázi začíná analýza. Analytik nebo celý tým dostanou priority k jednotlivým požadavkům. Stanoví se, jaké činnosti budou v dané iteraci. Analytici (a vývojáři) upravují modely, aby vyhovovaly změně. Připravují akceptační testy, mění datové modely, mohou probíhat menší změny v implementaci (úpravy pomocných knihoven atd.)

Implementace změny

V této části se plně uplatňuje samořízení týmu. Vedoucí týmu pravidelně kontroluje plnění úkolů (například jen 10 minutové tzv. stand-up schůzky), komunikuje změny do další iterace, odstraňuje problémy bránící vývojářům v práci. Agilní přístup roli vedoucího zásadně omezuje na rovnocenné postavení s týmem, kde všichni společně pracují na jedné věci, a to na dodržení obsahu v daném čase. Jde o nejdelší fázi.

Ukázka zákazníkovi

V průběhu iterací by se měly průběžně ukazovat výsledky zákazníkovi, tak aby mohl upřesnit své představy. Vždy bychom si měli najít čas na přípravu ukázky, tak aby byly změny zřejmé a bylo s ním co diskutovat [KNESL, 2009].

3.2.4 SCRUM

Je jednou z metodik agilního přístupu, zaměřenou na vývoj aplikací. Jde o metodiku na rozvoj a údržbu komplexních produktů. Autoři jej popisují jako: metodiku / pracovní rámec, který umožňuje řešit adresně komplexní problémy, a současně efektivně a tvořivě dodávat produkty s velkou přidanou hodnotou. Metodika se dle tvůrců vyznačuje:

lehkostí (co do rozsahu dokumentace), jednoduchostí (co do pochopení), extrémní složitostí na řízení a vůbec udržení efektivně funkční v praxi.

Metodika se používá pro komplexní vývoj produktů od počátku devadesátých let minulého století. Není určena pro výrobní procesy (výroba automobilů, nábytku apod.), ale jde o rámec pro začlenění různých technik a procesů v rámci vývoje. Používá se i v již existujících a funkčních procesech, které umí ještě zefektivnit [SCHWABER, 2011].

Opěrný bod metodiky tvoří SCRUM tým, k němu se asociují role, události, artefakty a pravidla. Každá komponenta v metodice má svůj účel a je základem pro úspěch nebo neúspěch řešeného projektu. Metodika vychází z empirické teorie procesu řízení. Předpokládá, že poznání, jak správně procesně realizovat projekty, pochází ze zkušeností a rozhodování o tom, co známe. Proto používá iterativní a inkrementální přístup k předvídání a omezení rizik. Filozofie stojí na třech pilířích: transparentnosti, revizích a adaptaci.

Transparentnost

Důležité aspekty procesu (projekt) musí být zřejmé těm, kdo jsou zodpovědní za výsledky. Transparentnost předpokládá definování požadovaných aspektů výsledků a v pochopitelné podobě sdílení v rámci projektového týmu.

Revize

Účastníci SCRUMu musí často revidovat, zda se neodchýlili od požadovaného směru. Revize by však neměly být tak časté, aby překážely v práci.

Adaptace

Pokud inspekce zjistí v jednom či více aspektech odchylky od akceptovatelného výsledného produktu, musí být všechny procesy, inkrementy a materiály upraveny do požadované podoby v co nejkratším čase, aby nezpůsobily ještě větší odchylky.

SCRUM popisuje čtyři formální události ke kontrole a adaptaci: plánování sprintu (Sprint Planning Meeting), každodenní SCRUM (Daily Scrum), revize sprintu (Sprint Review), retrospektiva sprintu (Sprint Retrospective).

3.2.4.1 SCRUM tým

Tým se skládá z vlastníka produktu (Product Owner), vývojového týmu (Developer Team), SCRUM vedoucího (SCRUM Master) a zainteresovaných stran (Stakeholders). SCRUM tým se organizuje sám a zastane všechny role, které známe z klasického vývoje (vodopád). Samo-organizující se tým znamená, že si tým sám volí, jak a kdy dokončí

konkrétní úkoly, raději než aby jej úkoloval někdo z venku. Týmový model je navržen tak, aby byl co nejvíce produktivní, efektivní a tvořivý. Tým pracuje v iteracích, které zaručují neustálou zpětnou vazbu od zainteresovaných osob, a dodávají hotové části v inktementech, čímž je vždy zaručena dostupnost poslední funkční verze.

Produktový vlastník

Tato role je zodpovědná za maximalizaci hodnoty spravovaného produktu a práce vývojového týmu. Reprezentuje zájmy zákazníka a spravuje tzv. backlog (seznam úkolů na vylepšení produktu). Určuje priority jednotlivých vlastností produktu, pravidelně je upravuje, přijímá a odmítá výsledky práce. Zajišťuje, že backlog je pro tým pochopitelný, sdílený a transparentní. Volnost metodiky je taková, že vše výše zmíněné může nechat na vývojovém týmu, ale zodpovědnost zůstává na něm. Produktový vlastník musí být vždy jen jeden člověk, ne komise nebo podobně. Aby byl produktový vlastník úspěšný, musí organizace respektovat jeho rozhodnutí. Jeho rozhodnutí musí být viditelná a v souladu s obsahem a prioritami backlogu. Vývojový tým nesmí nikdo jiný úkolovat a nesmí naopak přijímat úkoly od nikoho jiného.

Vývojový tým

Tým je složený z profesionálů, kteří každou iteraci dodávají inkrementy, a lze je nasadit na některém z IT prostředí firmy. Dodávku tvoří pouze vývojový tým. Je strukturovaný tak, aby se mohl vnitřně sám organizovat a rozdělovat práci. Výsledné synergie optimalizuje vývojový tým a dodává mu účinnosti a efektivitě. Jednotliví členové vývojového týmu mohou mít specializované dovednosti a oblasti zaměření, ale odpovědnost patří do vývojového týmu jako celku. Nicméně SCRUM zná jen pojem vývojář (developer), ať programuje, analyzuje nebo testuje. Metodika nezná role jako analytik, programátor a tester.

Velikost vývojového týmu

Měl by být tak malý, aby zůstal svižný a natolik velký, aby dokončil významné úkoly. Tým o třech členech není produktivní. Čelí problémům během sprintu, nedokáže dodat inkrementy. Tým s více než devíti členy zase dosahuje takové komplexity, že nejde empirickými metodami SCRUMu řídit. Produktový vlastník a SCRUM vedoucí se do členů týmů nepočítají.

SCRUM vedoucí

Je odpovědný za zajištění pochopitelnosti a fungování týmu dle SCRUMu. Tento vedoucí zajišťuje, že tým dodržuje metodické teorie, praktiky a pravidla. Je jakýmsi kustodem týmu, který pomáhá se změnami, tak aby byly co nejefektivnější ve své přidané hodnotě k výsledkům týmu.

SCRUM vedoucí a produktový vlastník

SCRUM vedoucí pomáhá produktovému vlastníku najít techniky na práci s produktovým backlogem, pomáhá jasně komunikovat vize, cíle a jednotlivé části produktového backlogu na vývojový tým. Zná a chápe dlouhodobé produktové cíle, praktikuje agilní přístup, provádí SCRUM událostmi.

SCRUM vedoucí a vývojový tým

Předává týmu znalosti, jak se má sám řídit a jak si správně rozdělit specializace tak, aby zvládl všechny klasické role vývojového týmu. Odstraňuje překážky pro rozvoj týmu. Koučuje vývojový týmu v prostředí podniku, kde SCRUM ještě není plně přijatý a pochopený.

SCRUM vedoucí a podnik

Vede a koučuje samotnou adaptaci podniku na SCRUM. Plánuje zavedení SCRUMu v jednotlivých částech podniku. Pomáhá zaměstnancům a zainteresovaným osobám pochopit fungující SCRUM a empirický produktový vývoj. Je hybatelem změn, které zvyšují produktivitu vývojových týmů. Pracuje s ostatními SCRUM vedoucími na celkovém zefektivnění této metodiky v daném podniku [ŠVENDOVÁ, 2011], [SCHWABER, 2011].

3.2.4.2 SCRUM události

Předepsané události existují z důvodů zavedení pravidelnosti a potlačení schůzek, které metodika nedefinuje. Pro každý typ události je vyhrazen časový blok, tak aby mohla být vykonána a aby se zároveň nemrhalo časem na plánování apod.

Sprint

Ken Schwaber nazývá sprint srdcem SCRUMu. Časový blok by měl být měsíc nebo méně, na konci by měl být použitelný a provozu schopný inkrement. Každý sprint musí mít pevnou délku, po jeho dokončení bezprostředně začíná další. Přirovnáno k jiným metodikám jde o typ iterace. Obsahuje plánování sprintu, každodenní SCRUM, vývojovou práci, revize sprintu, retrospektivu sprintu.

Během sprintu se nedělají žádné změny, které by ovlivnily cíl sprintu, kompozice vývojového týmu se nemění, požadavky na kvalitu se nesnižují, rozsah však může být projednán podle aktuálních zjištění s vlastníkem produktu.

Sprint lze považovat za projekt s jednoměsíčním horizontem. Vždy je jasná definice, co má být dodáno. Neměl by přesáhnout jeden kalendářní měsíc; tím se na tuto dobu snižuje i riziko nadměrných nákladů v případě nepochopení ze strany zadavatele (vlastník produktu).

Zrušení sprintu

Předčasné ukončení sprintu může provést jen produktový vlastník, ačkoliv to může nastat pod vlivem zainteresovaných osob, SCRUM vedoucího a vývojového týmu. Ke zrušení sprintu dochází, když podnik mění strategii, nebo když se změní podmínky na trhu nebo v technologiích. Většinou dává smysl sprint zrušit, pokud již cíl nedává za aktuálních okolností smysl. Když je sprint zrušen, většinou jsou již hotové položky backlogu přijaté a nadcházející, nehotové se musí znovu odhadnout. Zrušení sprintu plýtvá zdroji a narušuje vývojový tým - je to poměrně neobvyklý krok.

Plánování sprintu

Plánování práce, která by měla být vykonána během sprintu, by mělo být uvnitř vývojového týmu společné. Časový blok na plánování sprintu je osm hodin pro měsíční sprint, pro dvoutýdenní blok by to měla být zhruba polovina. Schůzka by měla mít dvě rovnoměrné části. V první bychom měli získat odpovědi na to, co budeme dodávat za výsledné inkrementy v následujícím sprintu. Sejde se vývojový tým, vlastník produktu představuje jednotlivé položky, tým se snaží předpovědět, jaké vlastnosti se tím budou upravovat / rozšiřovat. Snaží se od vlastníka produktu pochopit, jaká je jeho představa jednotlivých položek. Vstupem je backlog a historické údaje o efektivitě týmu. Nicméně, kolik položek z backlogu si tým vezme do nadcházejícího sprintu, je čistě na něm. Výsledkem této schůzky by tedy měl být konkrétní časově ohodnocený plán, který se tým zaváže splnit. Když jsou jasné jednotlivé položky, připraví se cíle (goals) sprintu. Jde o cíle, které by měly v průběhu sprintu týmu napomáhat uvědomit si, proč dané položky do inkrementu připravují. V druhé části se řeší, co tým potřebuje k dodání a jak dané části dodá. Vývojový tým obvykle začne s přetvářením položek backlogu do podoby inkrementů, tým si taktéž rozplánuje práci na jednotlivé činnosti pro nejbližších několik dní.

Cíle sprintu

Cíl sprintu dává vývojovému týmu určitou flexibilitu, pokud jde o funkčnost realizovanou v rámci sprintu. Jak vývojový tým pracuje, udržuje tento cíl na mysli. Aby bylo možné naplnit daný cíl sprintu, zavádí nové funkce a technologie. V případě, že se práce ukáže být složitější, než vývojový tým očekával, spolupracuje a vyjednává s vlastníkem produktu tak, aby naplnili dané cíle i po případných úpravách.

Denní SCRUM

Je mu určen časový úsek patnácti minut, slouží k synchronizaci v rámci týmu a vytváří plán pro následujících 24 hodin. Měl by se konat vždy ve stejnou denní dobu. Každý člen týmu by měl vysvětlit, co dokončil od poslední schůzky, co dodělá do příští schůzky a jaké má překážky v práci. Denní SCRUM týmu říká jeho posun v naplnění cílů sprintu a v přípravě inkrementů jednotlivých položek z backlogu. Vývojový tým může často po denním SCRUMu přeplánovat svou zbylou práci ve sprintu. SCRUM vedoucí zajišťuje uspořádání schůzky, ale za obsah schůzky je zodpovědný vývojový tým. SCRUM vedoucí tým učí, aby se vešel do patnácti minut, zajišťuje, že jde opravdu jen o schůzku vývojového týmu a nestává se z ní status zainteresovaných osob. Cílem těchto denních schůzek je zvýšení komunikace, odbourání jiných schůzek během dne, odhalení slepých vývojových větví inkrementů, rychlé řešení a rozhodování o aktuálních problémech.

Revize sprintu

Tato událost se koná na konci sprintu jako revize inkrementů a validace naplnění realizace jednotlivých částí backlogu. V rámci revize spolupracuje vývojový tým se zainteresovanými osobami a představuje jim, co bylo během sprintu vytvořeno. Na základě diskuze se připravují další úkoly do backlogu. Metodika této události přiděluje pro měsíční sprint čtyři hodiny.

Revize by měla obsahovat stanovisko produktového vlastníka, co je hotové a co ne. Vývojový tým by si měl prodiskutovat, co se povedlo, na jaké problémy narazil a jaké vyřešil. Vývojový tým by měl předvést ukázky hotových inkrementů a vysvětlit je. Produktový vlastník by měl říci, co z backlogu je hotové a co se plánuje. Všichni diskutují, jaké budou další aktivity, což je hodnotný vstup pro plánování sprintu.

Retrospektiva sprintu

Je to příležitost, aby se vývojový tým sám zhodnotil a případně si navrhl, jak pracovat jinak a co zlepšit pro další sprint. Pro měsíční sprinty metodika doporučuje na

tuto událost vyhradit tři hodiny. Mělo by se řešit, jak byl poslední sprint úspěšný s ohledem na vztahy v týmu a mimo něj, procesy a nástroje. Identifikovat, co se za minulý sprint zlepšilo. Vytvořit plán, jak zakomponovat do týmu různé zlepšováky [ŠVENDOVÁ, 2011], [SCHWABER, 2011].

3.2.4.3 Artefakty SCRUMu

Artefakty této metodiky jsou tvořeny tak, aby pomáhaly. S důrazem na srozumitelnost práce, která se má vykonat s propojením na hodnotu řešených produktů.

Produktový backlog

Jde o tříděný seznam všeho, co by mohlo přispět k rozvoji daného produktu. Je jediným seznamem požadavků na změny daného produktu. Produktový vlastník je zodpovědný za aktuálnost, dostupnost, třídění a priority tohoto artefaktu. Produktový backlog nebude nikdy kompletní, vývojem se vždy odhalí další možná zlepšení a ze strany produktového vlastníka taktéž. Bude potřeba tak dlouho, jak bude existovat produkt sám.

Jde o seznam všech vlastností, požadavků, vylepšení a oprav. Vždy by měl mít prioritu a odhad na řešení.

Sledování průběhů sprintů a naplňování cílů

V každém časovém bodu bychom měli být schopni sumarizovat, kolik práce bylo vykonáno na daném cíli. Produktový vlastník by měl mít informace, kolik času zbývá pro naplnění aktivit v daném sprintu apod. Je na něm, jakou si zvolí metodiku pro měření. Doporučovány jsou tzv. burndown a burnup či jiné projektové metriky pro předpovědi a měření pokroku projektů.

Sprint backlog

Jde o podmnožinu produktového backlogu, kde je však jen množina vybraných částí pro daný sprint obohacena o plán jednotlivých inkrementů pro daný sprint. Jde o plán, který je diskutován na denních SCRUMech. Plán je měněn podle naléhavosti jednotlivých aktivit a dle zjištění stavu jednotlivých aktivit po detailnějším rozpracování. Sprint backlog může měnit jen vývojový tým, přesto musí být transparentní (dostupný a pochopitelný).

Monitorování pokroku sprintu

Vývojový tým by si měl sám monitorovat na denních SCRUMech, kolik práce již odvedl na jednotlivých inkrementech, kolik zbývá, co má hotové.

Inkrementy

Jde o sumu všech hotových aktivit z produktového backlogu za jeden konkrétní sprint.

Definice slova „hotovo“

Všichni účastníci SCRUMu by si měli vydefinovat, co pro jejich konkrétní implementaci metodiky znamená, že je hotový inkrement. Předejde se tak nedorozuměním. Př.: zda byla dodávka otestována všemi úrovněmi tesů [ŠVENDOVÁ, 2011], [SCHWABER, 2011].

3.2.5 Zhodnocení vývojových metodik

Oba typy metodik (rigorózní a agilní) mají aktuálně své silné zastánce i odpůrce. Ani jedno z řešení není samospasitelné. Rigorózní metodiky ve valné většině bývají na první pohled strnulé, vyžadují projít si všemi aktivitami postupně. Z řad kritiků zaznívá mnoho hlasů o špatné reakci na změny v zadání apod. Mezi klady můžeme počítat psanou formou zadání, specifikace a dokumentaci. Proces a činnosti lze přesně popsat, projektové řízení patří k těm jednodušším. Doporučuje se použít pro velké a střední projekty. Taktéž je vhodné připomenout, že v této diplomové práci má popisovaná metodika UP v sobě zakomponované iterace, které z velké části reagují na potřebu změn v projektech. Agilní metodiky jsou postaveny na určitém typu volnosti v rozhodování řešitelských týmu, jak a kdy (v rámci iterace) budou řešit dané požadavky z tzv. backlogu. Metodiky stojí na snížení papírování a více jde o komunikaci tváří v tvář a dodávání menších výsledků za kratší dobu, aby byl prostor pro diskuzi mezi zadavatelem a řešitelem. Agilní metodika se doporučuje na menší projekty, drobný rozvoj nebo tam, kde je kladen důraz na krátkou reakční dobu a rychlé doručení nových funkcionalit. Mezi negativa patří malý podíl klasické dokumentace, jak jí známe z rigorózních metodik, a složitost na uplatnění v praxi. Přes svou základní jednoduchost se složitě zavádí v praxi a je závislá na ochotě lidí kvalitně spolupracovat a nést týmovou zodpovědnost i za méně zkušené členy. Občasným negativním argumentem je i velké přenesení kompetencí a zodpovědnosti na vývojový tým [ŠVENDOVÁ, 2011].

Z pohledu vývoje fron-endové aplikace se přikláním k použití některé z agilních technik. Automatizace firemních procesů je dobrou ukázkou oblasti, kde se dá použít přírůstkový (inkrementální) vývoj a v krátkých cyklech rozšiřovat funkcionality aplikací. Alternativou je možnost na míru si vybrat z obou směrů to nejlepší pro podnik, jak o tom pojednávám v další kapitole.

3.2.5.1 Spojení výhod rigorózních a agilních metodik v praxi

Tato stať vychází ze závěrů článku Cliffa Sarana „How to get the best from agile and waterfall development approaches“ [SARAN, 2013]. Autor řeší problematiku, kdy mnoho podniků není ochotno nebo nemůže efektivně přijmout agilní metodiky. Přesto v těchto podnicích přetrvává nesouhlas s tím, aby se vedly dvanáctiměsíční a delší projekty, kde není obchodní část podniku schopná průběžně měnit směry a reagovat na změny okolností. Agilní přístup řeší problémy tam, kde obchodní části dlouho neviděly výsledky a neměly na ně velký vliv (pomineme-li úvodní zadání). Nový přístup rozděluje velké projekty na malé a dodává menší části rychleji. Zadavatel vidí nějaký výsledek dříve, a tím se omezují situace typu: „To není to, co jsem chtěl“. Na druhou stranu se některé podniky od agilních technik po jejich vyzkoušení pomalu odvrací. Je to proto, že jim přinášely velké obtíže ohledně projektového managementu, správného časování dodávek velkých projektů, udržení a plánování rozpočtů a v neposlední řadě chyběla psaná detailních zadání a dokumentace. Ukázalo se, že 80% úkolů z celkového penza bylo vždy hotovo včas. Nicméně vývojové týmy si 20% toho nejsložitějšího nechávaly na konec a neplatila na ně historická pravidla pro odhady. Když se podnik Majestic Wine rozhodoval, jak řešit implementaci svého nového webu, systémový integrátor přizpůsobil metodiku Rational Unified Process tak, aby měla osm iterativních kroků, každý o délce tří týdnů. Podnik Yorkshire Water musel překonat vnitřní odpor ve firmě a místo čistě agilních technik zavést kombinaci. Jejich cílem bylo napojit agilní vývojový cyklus na ostatní neagilní části podniku. Za účasti konzultačních firem a úprav celkové podnikové architektury se jim to povedlo. Jednou z cest jak snížit reakční dobu IT je zavést prioritizace. Jinou ukázkou přístupu je tzv. MoSCoW (M - MUST, S – SHOULD, C – COULD, W – WOULD), kde zadavatelé, a pak v reakci vývoj řeší, zda jsou požadavky potřeba a kolik budou stát. Zmíněný přístup použila Cardiff University při spolupráci na projektech s Napp IT. Nakonec se ukázalo, že nejtěžší na tomto projektu byla organizace setkání tak, aby lidé dodrželi svou účast na týdenních a měsíčních schůzkách [SARAN, 2013].

4 Analýza

Úvodem si vydefinujme, jak vnímat pojem analýza v rovině této práce zaměřené na vývoj front-endových aplikací. Analýzou je míněn sled pracovních procesů a standardů analyzujících nutné informace k vydefinování a správnému pochopení požadavků na

budoucí vlastnosti aplikací, s následným vytvořením návrhů samotného technického zadání pro implementaci, kde se v průběhu a na konci realizace aplikací vytváří dokumentace popisující realizovaný stav front-endových aplikací. Definované úkony řeší role analytika (jedna z rolí specializované profese front-endový specialista), tak aby informace byly pochopitelné a snadno sdělitelné všem profesím v rámci podniku, které je potřebují. Za tímto účelem je nutné používat pro všechny zúčastněné srozumitelné standardy záznamu, které mají jasnou sémantiku a syntaxi.

Kapitola se zaměřuje na standardy použitelné pro tvorbu zadání a dokumentací podnikových procesů, řešených ve front-endových aplikacích. Taktéž řeší obsah práce front-endového speciality v rovině role analytika.

4.1 Standardy

Situace na poli analytických nástrojů je poměrně jednoduchá a přehledná. Celoevropsky se nejvíce používá standard UML a pro automatizaci se poslední desetiletí dostává do popředí BPMN, které umožňuje následný převod do reprezentace spustitelného kódu programu.

4.1.1 UML

Jde o zkratku prvních písmen anglických slov Unified Modeling Language, tedy jednotný modelovací jazyk. Počátky metodiky sahají do roku 1994, kdy Grady Booch a Jim Rumbaugh spolupracovali pod hlavičkou firmy Rational Software na spojování svých vlastních metodik (Booch a Object Modeling Technique). O dva roky později se k nim přidal Ivar Jacobson s jeho Object-Managed Software Engineering. Tím započalo UML ve verzi 0,9 společně s procesní metodikou Rational Unified Process (jejíž otevřenou verzi UP jsme si představili v 3. kapitole). Následně bylo v roce 1997 UML verze 1.1 uznáno standardizační komisí OMG. Dnes se můžeme setkávat s nejnovější verzí UML 2.4.1, jež OMG uznalo v roce 2011.

Jazyk UML byl navržen, aby spojil nejlepší modelovací postupy a techniky softwarového inženýrství. Návrh počítá se snadnou implementací v rámci CASE (computer-aided software engineering) nástrojů. Je koncipován tak, aby byl snadno pochopitelný pro lidi a zmíněné CASE nástroje.

Jazyk UML není vázán na žádnou konkrétní metodiku ani životní cyklus tvorby softwaru, kdy je preferován metodika UP. Jednotlivé diagramy lze používat dle zvyklostí daného podniku nebo vývojového oddělení.

Tvrzení o jednotnosti jazyka vychází z několika pohledů: UML nabízí vlastní syntaxi použitelnou během celého projektu vývoje softwaru. Jazyk lze využít pro návrh všech aplikačních domén, od systémů fungujících v reálném čase až po tzv. offline rozhodovací aplikace. Jakožto modelovací standard je nezávislý na programovacím jazyku, který bude finálně pro tvorbu software použit (nicméně UML má doplňky, které v určité fázi návrhu podporují konkrétní programovací jazyky). Zavádí vlastní interní pojmy, které se snaží prosazovat v rámci celého světa vývoje softwaru.

UML vidí tvorbu softwareu a systémů jako kolekci spolupracujících objektů. V rámci statických struktur popisuje, jaké typy objektů jsou pro modelování důležité, a jak spolu souvisí. Popisuje též dynamické chování aplikací a systémů, kde zachycuje jejich životní cyklus a jak spolu systémy spolupracují s cílem dosáhnout požadované funkcionality. Základní kapitoly standardu tvoří tzv. stavební bloky, společné mechanizmy a architektura [ARLOW, 2011].

4.1.1.1 Stavební bloky

Jde o tři základní entity, které si dále představíme: předměty (things), relace (relationships) a diagramy (diagrams).

Věci jsou jednotlivé prvky modelu. Buď popisují abstrakce struktur a chování jako třídy, rozhraní, případy užití, komponenty, interakce a stavy, nebo vyjadřují seskupení a poznámky, typicky balíčky souvisejících prvků a anotace, které lze k modelu připojit.

Relace umožňují vyjádřit, jaký vztah je mezi dvěma předměty. Velice důležitou roli v UML hraje porozumění sémantice vztahů, které umí UML vyjádřit. Jde o vztahy typu: závislost, asociace, zobecnění a další. (Vysvětlením užití bychom šli za rámec této práce, ale přesto je vhodné náznak škály možností zmínit).

Model můžeme chápat jako jakýsi repozitář sdružující všechny předměty a relace na dané téma. Diagram je pak pohledem na danou problematiku z konkrétní úrovně nahlížení. UML verze 2.0 zná 13 typů diagramů. Pro účely vývoje front-endu si představíme jen dva. Oba jsou ze seznamu diagramů chování, a to diagram případů užití a diagram aktivit [ARLOW, 2011].

4.1.1.2 Společné mechanizmy

UML zná čtyři obecné mechanizmy, jež jsou strategiemi, jak dojít k modelování objektů, které jsou opakovaně používány v různých kontextech. Standard rozlišuje specifikace, ornamenty, podskupiny a mechanizmy rozšiřitelnosti.

Modely mají dva rozměry, vizuální, prostřednictvím diagramů, symbolů, a textový, jenž se skládá ze specifikací různých prvků. Množina specifikací je hlavní významovou složkou modelu, texty mu dávají smysl.

V rámci UML má každý prvek svůj symbol, každý symbol pak lze v případě potřeby obohatit ornamentem, který graficky a informačně rozšiřuje atributy dané entity. Ornament si představme jako vyzdobený prvek. Máme-li entitu auto, tak v základní podobě jí zobrazíme jako obdélník obsahující popisek auto. Pokud jej budu časem rozšiřovat, obdélník rozšířím o další část, kam napíšu vlastnosti typu: typ motoru, barva apod. a daná entita bude vizuálně i informačně širší. Pamatujme vždy na to, že diagramy by měly být srozumitelné, a tedy nepřidávejme nepotřebné informace.

Podskupiny v UML vyjadřují různé vidění světa. Můžeme se setkat s dvěma skupinami klasifikátor, instance a rozhraní, implementace. Klasifikátor je abstraktním vyjádřením nějakého předmětu, př.: telefonní číslo, a instance je moje vlastní konkrétní telefonní číslo. Oproti tomu rozhraní je způsob, jak s něčím komunikovat, př.: klávesnice od počítače, implementace je však způsob jak někdo uvnitř klávesnice navrhl, že bude fungovat.

Standard dále rozlišuje tzv. mechanizmy rozšiřitelnosti, čímž univerzální nástroj přizpůsobuje ke specifickému použití. Rozlišuje omezení (constraints), stereotypy (stereotypes) a označené hodnoty (tagged values). Omezující prvky rozšiřují sémantiku prvku tak, že k němu umožňují přidávat nová pravidla. Ve valné většině omezují pravidla chování daného prvku. Stereotyp umožňuje založit nový prvek, který vychází z vlastností již stávajícího prvku. Př.: budu vytvářet služby a všechny aplikační služby budou označeny stereotypem <<aplikační služba>>. Naproti tomu označené hodnoty jsou jakékoliv hodnoty, které chceme k prvku přidat. Typicky to může být nějaký unikátní identifikátor pořadí [ARLOW, 2011].

4.1.1.3 Architektura

IEEE 1471:2000 definuje architekturu takto: „Je to základní organizace systému vtěleného do jeho složek, se vzájemnými vztahy a vztahy k prostředí, kde existuje, s principy pro jeho návrh a rozvoj“ [ISO/IEC/IEEE 42010]. Když se podíváme na definici dle UML, zjistíme, že je velice podobná. Vidí architekturu systémů jako organizační strukturu systému, včetně jeho rozkladu na součásti, jeho propojitelnosti, mechanizmů a směrných zásad, které promítá do návrhu systému. Jazyk UML se na architekturu dívá tzv. pohledem 4+1, kde řeší logický, procesní, implementační a nasazovací pohled plus případy užití.

Logický pohled dává důraz na popis hlavních objektů a tříd, jak spolu souvisí, aby tvořili funkční základ daného systému.

Procesní pohled je vlastně procesně orientovanou variantou logického pohledu s důrazem na spustitelná vlákna a procesy.

Implementační pohled popisuje soubory a komponenty, které vytvářejí samotné kódy aplikací. Pomáhá znázornit vztah mezi komponentami a zároveň umožňuje držet informace o možných konfiguracích a verzích.

Pohled nasazení říká, jaké komponenty byly nasazeny, na které fyzické počítače a servery v rámci konkrétního prostředí.

Všechny ostatní pohledy jsou odvozeny od případů užití, které jsou základními požadavky na chování aplikací a systémů [ARLOW, 2011].

4.1.2 Případy užití

Referenční příručka jazyka UML definuje případy užití jako specifikaci posloupnosti činností, včetně proměnných posloupností a chybových posloupností. Jaké systémy, podsystémy nebo třída může vykonávat prostřednictvím interakce s vnějšími aktéry. Případ užití systému je specifické užití systému konkrétním aktérem. Je doporučeno vždy začít případ užití aktérem a psát případy z pohledu aktéra. Z výše popsaného vyplývá, že případy užití jsou prakticky sepsané jednotlivé konkrétní požadavky na nové funkčnosti nebo jejich změny.

Nejlepší způsob, jak začít s definicí případů užití, je připravit si seznam všech aktérů a vydefinovat, jak budou systém využívat. V publikaci „UML 2 a unifikovaný proces vývoje aplikací“ jsou doporučeny následující pomocné otázky, které napomohou

pochopit filozofii této techniky i v rámci této práce: Jaké funkce od systému jednotliví aktéři očekávají? Bude systém uchovávat a poskytovat informace? Pokud ano, jací aktéři budou tuto činnost aktivovat? Jací aktéři budou upozorňováni na změnu stavu systému? Existují nějaké vnější události, které ovlivňují systémy? Co upozorní systém na tyto události? Reaguje systém na vnější systémy? Generuje systém zprávy?
Standard doporučuje k případům užití vypracovat slovníček pojmů, položkám v tomto seznamu by měli rozumět všichni účastníci projektu. Jde o seznam klíčových termínů daných oborů daného projektu a vyřešení otázek kolem případných synonym a homonym [ARLOW, 2011].

4.1.2.1 Specifikace případu užití

Z vlastní praxe si troufám tvrdit, že jde o nejdůležitější část případů užití. V rámci UML sice neexistuje žádný standard, ale obecně se evidují následující textové informace: název případu užití, jedinečný identifikátor případu, stručný popis – odstavec, zachycující stručný popis, aktéry zapojené do případu užití, vstupní a výstupní podmínky, toky událostí, hlavní a alternativní scénáře. Dále si vysvětlíme některé z pojmů, které nemusí být samy o sobě zřejmé. Vstupní podmínky jsou takové okolnosti a pravidla, která musí nastat před tím, než se může případ užití aktivovat. Výstupní podmínky naopak specifikují pravidla a okolnosti pro dokončení případu. Toky událostí vnímejme jako jednotlivé aktivity vedoucí k naplnění případu užití. Případ užití se nemusí naplnit jen jedním způsobem. Proto scénáře jakožto sledy toků událostí dané sady popisují jednotlivé možnosti používání. Jeden ze scénářů musí být vždy hlavní, ten požadovaný, nejvíce používaný apod., ostatní jsou alternativní. Narazíme-li při definicích případů na neurčitosti, přílišná zobecnění apod. zkusme si odpovědět na jednoduché otázky: Kdo přesně? Jak přesně? Kdy přesně? Kde přesně? [ARLOW, 2011]

4.1.3 Diagramy aktivit

Jde o objektově orientované vývojové diagramy umožňující modelovat procesy jako aktivity, které se skládají z uzlů a hran. Diagram je doporučený pro použití během analýzy jakožto grafický doplněk k případům užití, taktéž jako model propojení jednotlivých případů užití. Během návrhu ho lze použít k modelování jednotlivých operací.

Taktéž lze použít modelování obchodních procesů podniku (v podobném duchu bude případ této práce).

Diagram aktivit je velmi srozumitelný a dobrý komunikační prostředek, nesmíme však jít do přílišného detailu a vytvořit jej příliš složitý. Diagram aktivit lze připojit k jakémukoliv prvku standardu UML, v našem pojetí jej budeme používat s případy užití. Aktivitou je sám modelovaný objekt diagramu. Může jít o obchodní činnost, sled pracovních kroků nebo procedurální logiku. Uvnitř aktivity jsou objekty typu akční uzly, které zastupují samotné jednotky vykonávající nějakou akci. Dále potřebujeme znát řídící uzly a spojnice, jež propojují jednotlivé objekty aktivity diagramu a říkají, kudy a jak se větví nebo slučuje daný proces. Jako poslední potřebujeme znát objektové uzly, jež představují důležité entity v aktivitě vystupující nebo vytvářené. Pro přehlednost lze diagram aktivit zpřehlednit rozdělením do svislých nebo vodorovných oddílů [ARLOW, 2011].

4.1.3.1 Akční uzly

UML zná v české terminologii tyto akční uzly: volání, odeslání signálu, přijmutí události a přijmutí časové události (CASE nástroje mají tuto množinu ještě širší). Volání iniciuje aktivitu nebo operaci. Odeslání signálu odešle asynchronně (nečeká na potvrzení o přijetí) signál. Objekt s názvem přijmutí události, pokud je propojen s jinými objektem, čeká na signál o přechozích aktivitách. Pokud mu nic nepředchází, spouští se naplněním předchozí aktivity nebo konkrétních podmínek. Přijmout časovou událost představuje akční uzel reagující na čas (př.: definovaný čas pro spuštění celé aktivity).

4.1.3.2 Řídící uzly

Tyto uzly představují symboly pro zobrazení začátku, konce aktivit, jejich větvení v rámci procesu apod. UML zná tyto objekty: počáteční uzel, konečný uzel aktivity (celého diagramu), konečný uzel cesty, uzel rozhodnutí, sloučit uzel, rozvětvit uzel a spojit uzel. Uzel rozhodnutí slouží pro možnost definování podmínek, za kterých bude proces diagramu pokračovat různými cestami. Slučovací a spojovací uzel bez další podmínek či pravidel sloučí více předchozích cest do jedné. Rozvětvit uzel umožňuje rozdělit jednu cestu do více, tak aby šlo dvě a více cest paralelně.

4.1.3.3 Objektové uzly

Jde o zvláštní uzly signalizující dostupnost instancí daného objektu v rámci dané části aktivity.

4.1.4 BPMN

BPMN (The Business Process Modeling Notation) představuje standardizovaný proces a techniky pro tvorbu podnikových procesů. Samotný proces je ve slovníku definován jako sled akcí, změn nebo funkcí přinášejících nějaký výsledek. Bill Curtis definuje proces jako částečně uspořádanou množinu kroků podniknutou za určitým cílem. Historie BPMN sahá do roku 2004, kdy konsorcium BPMI (Business Process Management Initiative) představilo první verzi BPMN 1.0. Tento standard se dle Curtisovy klasifikace řadí do rodiny dynamických procesních jazyků a vykazuje tyto znaky: zaměřuje se na dynamický pohled na podnikové procesy, které používají lidští aktéři, a nějak do nich zasahují. Současně podporují formát XML a představují standardizační úsilí velkých hráčů na trhu. Standard pokrývá funkční pohled, informační pohled a stejně tak organizační pohled na podnikové procesy. Přestože primární účel BPMN je pochopitelnost pro lidského čtenáře, v mnoha implementacích a navazujících standardech (př.: BPEL) je počítáno se strojem čitelnou aplikací na konkrétní programovací jazyky.
Použití procesních modelů přináší snížení pravděpodobnosti, že si s novými obchodními procesy zavedeme zásadní chyby. BPMN nabízí bohatou sbírku notačních prvků pro definování pracovních postupů v podobě diagramu a přísné metody k zajištění správnost modelu. Napomáhá identifikovat chyby a odstranit je z návrhu podnikových procesů [MILI, 2010], [CURTIS, 1992], [LAM, 2010].

4.1.5 Symboly a sémantika

BPMN rozlišuje tři kategorie objektů: tokové objekty (flow objects), spojovací objekty (connecting objects) a plavecké dráhy (swimlanes). Chování procesu je specifikováno pomocí objektů toků a spojovací, notační prvky jsou řešeny pomocí plaveckých drah. Metodika obsahuje i datové objekty, notační a skupinové, ale ty nemají zásadní vliv na stav a fungování procesních modelů. Stejně tak se můžeme v rámci BPMN setkat s diagramy konverzace (Conversation Diagram) a choreografie (Choreography diagram), které mají své specifické prvky. My se však zaměříme na prvky

nejpoužívanějšího diagramu, vhodného pro záměry této práce, na diagram spolupráce (Collaboration Diagram).

4.1.5.1 Tokové objekty

BPMN zná tři typy události (events), aktivity (activities) a brány (gateway). Procesní tok diagramu začíná a končí událostmi počáteční událostí respektive konečnou událostí. Pro události v průběhu procesu se používá tzv. průběžná událost. Tento typ události může být klasifikován různými typy jako je zpráva, časovač, chyba nebo vazba (na jiný proces nebo paralelní průběh). Aktivity jsou buď úkoly, nebo podprocesy. Úkol je taková činnost, která nemůže být rozložena. Naproti tomu podproces lze rozložit na soubor dalších aktivit. Dalšími objekty jsou brány, které umožňují větvení a slučování toků nebo procesů v závislosti na zadaných podmínkách. BPMN zná tzv. exkluzivní brány, kde proces může proběhnout pouze jednou z cest. Též rozlišuje brány s opačným chováním tzv. inkluzivní, které se používají pro procesy, kde se dá branou projít více způsoby. Poté se většinou sjednotí do jedné cesty [LAM, 2010].

4.1.5.2 Spojovací objekty

Standard zná sekvenční tok (Sequence Flow), pomocí kterého se zachycuje posloupnost procesu. Zdrojem a cílem je vždy událost, aktivita nebo brána. Je zachycen plnou čarou. Tok zpráv (Message Flow) popisuje komunikaci v rámci dvou a více bazénů, zobrazuje se přerušovanou čarou na začátku s kruhem a na konci šipkou ukazující směr procesu. Asociace (Association) je typ spojení, kdy chceme k jinému objektu diagramu připojit specifický artefakt nebo nějaký komentář [LAM, 2010].

4.1.5.3 Plavecké dráhy

V BPMN se používá pojmů bazén (pool) a dráha (lane). Bazén je hlavní prvek procesu. Jde o hlavní procesní prvky organizace a v rámci většího detailu či pro správné zachycení spolupráce jednotlivých aktérů může obsahovat více drah vymezujících jednotlivé role. Každý bazén má minimálně jednu dráhu [LAM, 2010].
Základní popis jednotlivých objektů a ucelená symbolika je dostupná z http://bpmb.de/poster.

4.1.6 Zhodnocení standardů procesní analýzy

Podíváme-li se na dva nabízené standardy (UML diagram aktivit a BPMN) optikou analytiků, kteří budou analyzovat podnikové procesy a následně navrhovat a dokumentovat jejich automatizaci pomocí front-end aplikací, zjistíme, že obě řešení jsou vhodná. UML nabízí solidní zázemí potencionální i projektové metodiky a plné sady nejen dynamických modelů a diagramů. Pokud bychom chtěli UML používat ve standardní šíři, nabylo by toto řešení na složitosti v rámci zaškolení lidí a přípravy podniku na hladké přijetí. Vzhledem k většímu stáří a zatím větší známosti přináší UML v České republice výhodu znalostí na straně informačních specialistů. Oproti tomu BPMN nabízí v některých ohledech záznamu/modelování jednodušší vyjádření reality a je nativně lépe připravené na automatický převod procesů do strojového kódu. Mezi nevýhody patří nemožnost sdílet výsledné diagramy bez osvěty se všemi členy projektu (čitelnost diagramů potřebuje alespoň základní školení).

Každý podnik, který se rozhodne zavést nástroj na tvorbu front-endových aplikací, si musí na základě své konkrétní situace zvážit klady a zápory obou standardů dle svých aktuálních potřeb a zvyklostí. Tam, kde se celé vývojové týmy orientují v UML, bude jasnou volbou tento standard. V případě, kdy začínáme tzv. na zelné louce nebo máme nástroje na převod např.: z BPMN do BPEL, by měl být volbou standard BPMN.

Pro větší objektivnost shrnu v další části této kapitoly závěry Cristine V. Geambasu z práce BPMN vs. UML Activity Diagram for Business proces Modeling, kde na základě tří hlavních kritérií porovnávala vhodnost obou standardů.

První kritérium je rozsah pochopitelnosti standardu. Dle autorky je důležitá pochopitelnost standardu přes všechny role účastníků projektu, což následně silně podporuje pochopitelnost záměrů a úspěch projektu. Oba standardy hodnotí jako stejně pochopitelné a dokládá tento fakt experimentem Daniela C.C. Peixoto, který představil oba standardy studentům informatiky, kteří jej blíže neznali. Výsledkem experimentu bylo zjištění, že obtížnost obou standardů je na stejné úrovni.

Druhým kritériem je přiměřenost grafické reprezentace elementů BPMN a UML vůči reálným obchodním procesům. Pro porovnání zvolila autorka případovou studii teoretického procesu autoservisu a jeho zákazníka. Namodelovala dvě varianty - jednu pomocí BPMN 2.0 a druhou v UML 2.0 standardu. Dospěla k závěru, že grafická

symbolika je ve většině částí modelu shodná. Nicméně pro část reprezentující vykonání opravy vozu, BPMN postačí jeden symbol a UML používá skupinu symbolů.
Třetím kritériem je mapování obchodního procesu na programovací jazyk. Autorka píše: Dalším krokem po vytvoření vizuální reprezentace obchodního procesu, v modelovacích jazycích jako je BPMN a UML, je realizace. Cílem je porovnat vhodnost na převod do programovacího jazyka BPEL (Business Process Executing Language). Dále konstatuje, BPMN 2. obsahuje mapování, které lze použít vůči verzi BPEL orientované na webové služby zvané WSBEL. Přímou cestou nejde do podoby výkonného kódu převést všechny objekty, ale podpora tam je. Oproti tomu UML v rámci svých norem s žádným podobným mapováním nepočítá. I když posledních několik let probíhaly výzkumy na téma automatického převodu do spustitelného kódu, zatím neexistuje celistvý výsledek aplikovatelný v praxi [GEAMBASU, 2012], [Peixoto, 2008].

4.2 Činnosti v rámci analýzy

Kapitola shrnuje prvky klasické analýzy a specifik návrhů a dokumentace pro vývoj front-endových aplikací.

V prvním části řešení se musí specialista seznámit s obsahem požadavku. Komunikací se zadavatelem a přečtením zadání musí nabýt jasné představy o cíli požadavku. Rozdělí si zadání do funkčních a nefunkčních celků. Nefunkční požadavky popíše jen základním popisem, protože se o ně stará jiná vrstva IT architektury.

Připraví si návrhy daných procesů, které má front-end pokrývat, a do něj zasadí jednotlivé obrazovky pro komunikaci systému s uživatelem. V rámci této činnosti připraví ve spolupráci se zadavatelem testovací scénáře pro ověření funkčnosti zadání. Aby byl krok analýzy ukončen, proběhne odsouhlasení výstupů se zadavatelem.

Pokud bude činnost realizována v rámci projektu, měly by veškeré aktivity řízení projektu vycházet ze souladu s podnikovou architekturou. Informačně technologická část projektu by měla ctít některou z metodik vývoje a samotná analýza by měla využívat některý ze standardů.

4.2.1 Modelování podnikových procesů a simulace

Níže popisované téma nemusí být nutnou součástí analýzy, ale některé praktiky se dají použít i pro část návrhu. Samotná simulace již nemusí být součástí. Pracovní techniky v kapitole popsané pocházejí z práce Manufacturing process analysis with support of

workflow modelling and simulation. První krok je převedení podnikové strategie do objektů procesních cílů. Vytvoření modelu, na kterém lze ukázat, které procesy vedou k naplnění firemní strategie a následně nechat na manažerech podniku rozhodnout o realizaci. V druhém kroku se mají definovat měřitelné cíle pro jednotlivé objekty procesních cílů (zlepšení). V hrubých rysech by se měla vybírat kritéria typu: celkový čas procesu, náklady, potřebné kapacity, utilizace zdrojů, produktivita a kvalita daných procesů. Ve třetím kroku bychom měli přiřadit měřitelným kritériím cíle, jakých chceme změnami docílit. Čtvrtým krokem je vybrání způsobu jak zvýšit efektivitu procesu. Jak autoři konstatují, zvyšování propustnosti procesů a maximalizace přínosů je nejpopulárnější manažerská oblast. Existují dvě základní cesty. První znamená počítat se stávajícími kapacitami a snažit se úpravou kapacit nebo procesu dosáhnout větší efektivity. Druhá znamená počítat s dlouhodobou strategií a rozdělením do kategorií: úroveň procesů, aktivit, zdrojů a řízení. Průběžně se zabývají daty jednotlivých kategorií a výsledky, zároveň se pracuje na simulaci v rámci modelu. Výsledky simulací určí, jakým směrem by se měly upírat změny daných podnikových procesů [LIN, 2009].

5 Vývoj

V rámci vývoje dochází k samotnému neparametrizování jednotlivých procesů, k tvorbě obrazovek a napojení vstupních a výstupních dat z obrazovek na databáze uvnitř back-endů. Po ukončení vývoje FE specialista ověří funkčnost takzvanými unit testy. Ověření, zda základní operace fungují, tak jak bylo požadováno.

V klasickém vývoji bychom v této části potřebovali vývojáře, který napíše samotný kód. V rámci cílů této práce však předpokládáme existenci nástroje, kde front-endový specialista definuje procesy, parametrizuje je (jak z pohledu procesů, tak na funkce back-end systémů) a tvoří obrazovky. Pro účely práce v jedné variantě předpokládejme, že takovýto nástroj si podnik na objednávku interně či externě vyvine, anebo použije jedno z řešení na trhu, které si dále představíme.

5.1 Nástroje realizace front-endových aplikací

V následujících kapitolách naleznete základní popis a zhodnocení aplikací, které lze zakoupit na trhu vývojových nástrojů pro tvorbu podnikových procesů. Všechny uvedené

aplikace mají společný znak. Lze je zakoupit jako standardní softwarový produkt. Obsahují ucelenou paletu funkcionalit umožňujících z návrhu vytvořit koncovou front-end aplikaci.

5.1.1 IBM Business Process Manager

Skupina nástrojů na automatizaci podnikových procesů navazující na populární software Lombardi, který v rámci akvizice získalo IBM v roce 2010. Historie původního Lombardi sahá až do roku 2000. Aktuálně je na trhu dostupná verze 8, nicméně v následující kapitole si popíšeme vlastnosti verze 7.5, které se může lehce lišit od aktuálně dostupné varianty.

IBM Bussines Process manager (dále BPM) se skládá ze dvou hlavních aplikací: IBM Integration Designer a IBM Process Designer, které zastřešuje IBM Process Centrum. IBM Integration Designer umožňuje vytvořit novou službu, tedy technologickou naprogramovanou komponentu, která vykonává nějaké konkrétní funkčnosti a komunikuje s konkrétním systémem. Tímto nástrojem je možné na míru nebo pomocí předvyvinutých konektorů zajistit integraci nově vyvíjených procesů na okolní nové či stávající systémy.

IBM Process Designer nabízí možnost importovat nebo přímo vytvářet nativní diagramy BPD (Business Process Diagram), které jsou kompatibilní se standardem BPMN. Zde můžeme vytvořit návrh procesu, připravit obrazovky budoucí aplikace a nastavit jednotlivé parametry procesu a v neposlední řadě nastavit služby vytvořené v rámci IBM Integration Designeru. Celý proces lze zkompilovat a spustit.

Zmíněné aplikace jsou zastřešeny v IBM Process Centru, které je knihovnou všech součástek procesů a služeb. Veškeré zmíněné aplikace jsou kompatibilní s ostatními aplikacemi IBM pro integraci a podnikové procesy (Př.: IBM Websphere Message Broker, IBM Blueworks Live) [IBM, 2007].

5.1.2 IBM Blueworks Live

Jedná se o odlehčenou verzi výše popsaného nástroje IBM Business Process Manager. Nástroj je zaměřen zejména na automatizaci a digitalizaci kancelářských procesů. Předpokládá automatizaci schvalovacích a jiných procesů s možností automatizace prací spojených s emaily a digitálními dokumenty. Řešení je postavené na tzv. cloudu, tedy všechna data a definice procesů jsou na vzdáleném serveru, všechny procesy jde tedy zpřístupnit i pro mobilní zařízení a tablety. IBM proklamuje možnost zrychlení podnikových procesů až 12krát. Software obsahuje sofistikovaný způsob návrhu

procesních diagramů opět s využitím BPMN standardu, s velmi intuitivním návrhem a jednoduchým zprovozněním procesů i bez podpory klasických programátorů. Mezi nedostatky patří slabá možnost integrace na jiné aplikace v podniku [IBM, 2013].

5.1.3 Appian BPM Suit

Appian představuje nástroj pro návrh podnikových procesů, napojení na služby jiných systémů, vygenerování samotných programových kódů a nasazení na požadované prostředí. V nejnovější verzi přidává široké spektrum tzv. sociální spolupráce nesené na vlně popularity sociálních sítí a stejně tak umožňuje práci z mobilních zařízení.

Pro modelování podnikových procesů je použit standard BPMN, který umožňuje propojení pomocí různých typů služeb vycházejících z idejí SOA (Service Oriented Architecture). Samotná aplikace pro návrh procesů je řešena jako webová, tedy není příliš závislá na počítači uživatele a jde ji spustit z podporovaných prohlížečů. Návrhářská část obsahuje repozitoř, kde se dle potřeby dají sdílet jednotlivé diagramy podnikových procesů s možnosti řízení práv dle rolí uživatelů apod. Appian BPM Suit obsahuje aplikaci Appian Smart Services, které umožňuje poměrně jednoduše se silnou grafickou podporou vytvářet tzv. podnikové služby (enterprise services), na které se pak velice jednoduše dají napojit podnikové procesy. Tyto služby se dají následně znovu použít a můžeme zde držet galerii všech služeb s možným využitím k integraci podnikových procesů na nové či stávající back-endové aplikace. Samotné vytvoření procesů jako funkčního programu by mělo být dle Appianu velice jednoduché, nástroj je konstruován s možností jednoduchého a rychlého nasazení výsledného funkčního programu. Před samotným zprovozněním uvítá každý návrhář možnost validace procesů, ať už po stránce formální, tak logicko-funkční. Nástroj nabízí též část pro sledování nasazených a běžících procesů. Monitorovací část průběžně upozorňuje na úzká hrdla jednotlivých procesů, čímž napomáhá neustálému zlepšování.

Nástroj též umožňuje návrh složitějších formulářů. Díky tomu může front-endový specialista navržený podnikový proces jednoduše doplnit obrazovkami (formuláře) dle vlastního návrhu co do funkčnosti a vzhledu. Appian na svých stránkách uvádí silnou grafickou podporu návrhu, intuitivní webové rozhraní, multi-jazykové verze, mapování polí na externí data, dynamická pole, formátování a validace polí.

Z hlediska integrace Appian BPM Suit podporuje SOAP a REST webové služby, databázovou integraci (RDBS) a v rámci aplikace Smart Service umožňuje konfiguraci a vytvoření služeb zcela nových. Mezi další komunikační možnosti patří samozřejmá podpora e-mailů a jistě velmi užitečná podpora SFTP přímo z návrhu podnikového procesu. Autentifikaci, práva a role lze řešit pomocí LDAPu, jež též podporován.

Z ostatních funkcí je vhodné zmínit podporu pro unifikaci podnikových dat v rámci podnikových procesů procházejících Appian procesy silnou podporu analytických pohledů na podniková data a využívání standardních datových formátů typu XML. Aktuální verze nabízí možnost tvorby podnikových procesů spustitelných z webových prohlížečů či mobilních zařízení.

Velkou devizu představuje nepotřebnost klasického vývojáře při návrhu i realizaci podnikových procesů [Appian, 2013].

5.1.4 Intalio Create

Společnost Intalio již více jak třináct let vyvíjí nástroje pro automatizaci podnikových procesů a tvorbu front-endových aplikací. Opět jde o nástroj, který umožňuje pomocí grafického rozhraní navrhnout jak podnikový proces, tak obrazovky a zároveň se napojit pro integraci na nové nebo stávající back-endové aplikace podniku. Mezi hlavní rysy nástroje patří používání digramů dle standardu BPMN, kde pro samotný návrh jsou určité objekty graficky jinak řešeny, ale vše funguje dle standardu a dají se naimportovat již hotové modely. Navíc Intalio nabízí volně dostupný a interně upravitelný systém pro management podnikových diagramů Open source Business Process management systém, ve kterém si podnik může vytvářet a udržovat diagramy podnikových procesů dle standardu BPMN.

Vrátíme-li se k samotnému Intalio Create, jeho základní část tvoří návrhář podnikových procesů, ze kterého se dá napojit na jednotlivé podnikové služby a neparametrizovat jej. Část pro návrh je vytvořena jako plně grafické prostředí s možností přetahovat si již vytvořené objekty ze seznamu, a na specialistovi je definování datových typů, rozhodovací logiky a mapování parametrů jednotlivých kroků. Nástroj též umožňuje individualizovat jednotlivé obrazovky, přestože mají jednotný vzhled, samotná pole pro hodnoty lze libovolně upravovat a mapovat dle potřeby. Taktéž je zde možnost zpřístupnit jednotlivé aplikace mobilním aplikacím. Zajímavou variantou tvorby návrhu je použití

generátoru objektů do procesních map z tabulkových procesorů, př.: MS Excel. Postačí naimportovat požadovaný list s údaji požadovaných objektů, které chceme nově zpracovávat automaticky v nějakém procesu pomocí migrace v nástroji Intalio Creator. Nástroj sám navrhne jednotlivá datové pole a okamžitě je můžeme zakomponovat do podnikových procesů. V části Intalio Pipes lze za pomocí grafického rozhraní využít velice širokou paletu standardů připojovacích konektorů na okolní systémy podniku a zapojit tak téměř jakýkoliv back-end do podnikových procesů s jednotným front-endem.
Opět je zde velká výhoda nepotřebnosti klasického vývojáře, návrh i samotnou implementaci zvládne front-endový specialista, který může zkompilované výsledné aplikace přímo nasadit na požadované prostředí [Intalio, 2013].

5.1.5 Bonita BPM

Bonitasoft vytváří popisovaný nástroj již od roku 2001. Nástroj je určen k návrhu a implementaci podnikových procesů a jejich manažerské i technické správě. Jako jediný z představovaných nástrojů je v základní verzi zdarma, navíc řešen politikou open-source.

Bonita BPM se skládá z tří základních komponent: Bonita Studio, Bonita BPM Engine a Bonita Portal. Bonita Studio je určen k návrhu obrazovek a připojení na okolní back-endové aplikace podniku. Samotné obrazovky mají v základní verzi jednotný vzhled, ale uživatel si je může libovolně upravovat, co do šíře a typu polí. Sám nástroj pak pro konektivitu na okolní systémy přináší širokou paletu komponent, které lze dále rozšířit buď přímo ze strany Bonitasoft nebo interním vývojem daného podniku. Bonita BPM Engine je samotný poháněcí mechanizmus, jehož specifikace a parametry jsou určené pro IT architekty a provozní oddělení podniku. Bonita Portal míří na manažerskou správu podnikových procesů. Umožňuje vedoucím, pod které spadají konkrétní již zautomatizované procesy, aby manuálně zasahovali do procesů, předávali je na jiné pracovní apod. Taktéž jim umožňuje tvořit statistiky a řídit uživatelská práva v rámci jejich jurisdikce.

Bonit BPM má i placenou verzi, které přidává možnost kooperace nad stejnými procesy více lidem, znovu používání obrazovek, jejich větší upravitelnost i co se vzhledu týče a v neposlední řadě podporu ze strany Bonitasoft v případě zaškolení, řešení problémů a chyb.

Celkově jde o porovnatelný nástroj jako výše popsané systémy. Podnik dostane nástroj, který umožní front-end specialistům vytvářet automatizované procesy a taktéž je nasazovat, cest od podnikových idejí k realizaci může být velmi rychlá. Klasičtí vývojáři budou potřeba pro implementaci nových podnikových služeb v rámci integrace a to jen tam, kde půjde o nestandardní služby na specifické podnikové back-endy [Bonitasoft, 2013].

5.1.6 Oracle Application Development Framework

Oracle Application Development Framework (dále jen Oracle ADF) představuje konglomerát aplikací umožňující mimo jiné návrh podnikových procesů a následnou snadnou implementaci. Od jiných v diplomové práci popisovaných nástrojů se liší tím, že se orientuje na obecný vývoj aplikací. Má opravdu širokou paletu aplikací, které podporují vývoj zejména na JAVA platformě. Z toho plyne silná orientace spíše na klasické vývojáře nežli front-endové speciality, u kterých se nepočítá se znalostmi tvorby programového kódu.

Oracle ADF obsahuje sekci pro návrh BPMN procesů, které se dále dají parametrizovat. Díky komplexnímu záběru vývojářských potřeb obsahuje i modul pro správu, a pak zejména vývoj zcela nových podnikových služeb. Ty lze jednoduše navázat na podnikové procesy a zajistit integraci na podnikové back-endy. Oracle ADF též má svébytný návrhář obrazovek, který na rozdíl od výše popisovaných nástrojů umožňuje neomezeně navrhnout obrazovky ať už po stránce funkční, tak i vizuální.

Celkově tedy Oracle ADF nabízí vše, co výše recenzované systémy. Výhodou a nevýhodou zároveň může být podstatně složitější zaučení v práci s tímto nástrojem. Některé podmoduly jsou orientovány přímo na klasické vývojáře a slouží k tvorbě čistého kódu. Je tedy na potencionálním podniku, aby si rozmyslel, jak bude složen tým, který bude vytvářet fron-end, a zda se mu toto řešení vyplatí [Oracle, 2013].

5.2 Test

Podniky se stále více spoléhají na své aplikace. Slouží jim nejen k efektivnímu řízení vlastních činností, ale i ke komunikaci se zákazníky. Námi zamýšlený front-end bude pravděpodobně aplikace založená na webovém rozhraní. Pro její uvedení do provozu a stabilní chod budou zapotřebí testy, které ověří, zda splňuje funkční, ale i výkonové parametry, které se od ní očekávají.

Pro přípravu zátěžových testů (ověřují výkon a stabilitu) je nutná znalost jak firemních procesů, tak i chování uživatelů. Na příkladu z praxe si můžeme demonstrovat, že chování uživatelů je důležité pro odladění front-endové aplikce. Analýza na příkladu využívání knihovnicko/aukční aplikace ukazuje, že vice než 70% dotazů bylo na prohlížení knih. Ale jen 1,19% dotazů směřovalo k nákupu. Autoři též našli významné procento dotazů vygenerovaných z automatických robotů, kteří donekonečna dotazovali aplikaci a shromažďovaly informace za různými účely (příkladem indexování cen oproti jiným aplikacím) [KRISHNAMURTHY, 2010], [Just Another Blog by Johnny, 2008].

Testy se rozdělují do dvou oblastí: prvním typem jsou integrační testy. V těch FE specialista spolupracuje s back-endovými aplikacemi na ověření bezchybné funkčnosti přes celou infrastrukturu firmy. Musí otestovat, zda front-edová aplikace správně komunikuje přes všechny použité služby. Po úspěšném ověření funkčnosti jako celku následují tzv. akceptační testy, kde zadavatel akceptuje aplikaci vůči původnímu zadání. Pro oba typy testů se využívají testovací scénáře vytvořené ve fázi analýzy.

Druhým typem rozdělení je pohled na způsob vykonání testů, zda jde o testy manuální nebo automatické. Manuální testy jsou standardem, tester vykonávající test funkčnosti aplikaci testuje přes front-end a případně kontroluje data v back-endech, zda se data správně propsala či jinak byla naplněna požadovaná funkčnost. Automatické testy jsou prováděny speciálními programy. V těch specialista na automatické testy nastaví požadované testy a očekávané výsledky, tak aby se testy daly spouštět vícekrát a nevyžadovaly na samotné vykonání lidskou sílu. Tento typ řešení je z pohledu prvotních investic podstatně dražší, ale u tzv. regresních testů se u opakovaných testů v horizontu pěti a více spuštění vstupní investice vrací [BARBOSA, 2007], [KRISHNAMURTHY, 2010], [Just Another Blog by Johnny, 2008].

6 Modelové použití

6.1 Příklad použití

Pro lepší pochopení a omezení generalizace, bude v následujících odstavcích popsán příklad fiktivní banky, která se chystá na implementaci aplikace pro tvorbu front-endu a následně musí zavést i pracovní postup na tvorbu jednotlivých obrazovek a činností v rámci front-endu jako takového. Příklad vychází z předchozích teoretických kapitol

ohledně metodologie a životního cyklu, tak aby představoval možné použití v praxi na bázi příkladu.

Fiktivní banka bude v textu označovaná jako Banka ABC. Tato společnost má tři základní produkty: běžný účet, spořící účet a životní pojištění, které nabízí za tzv. třetí stranu (Pojišťovnu XY). Banka ABC má vybudovanou IT infrastrukturu, kde na úrovni významných koncových aplikací má na core bankovní aplikaci pro provádění transakcí, jejich správu, vytváření a rušení běžných účtů apod. Též má ve svém portfoliu CRM (Customer Relationship Management) systém na správu a evidenci zákazníků a jejich produktů. Dále má samostatnou aplikaci pro zakládání a správu spořících účtů a samostatnou aplikaci, přes kterou se zakládají a spravují životní pojištění, tato aplikace vzdáleně komunikuje s centrální databází pojišťovny XY. Banka navíc vlastní podpůrné menší univerzální aplikace pro jednotnou správu uživatelských práv a monitoring funkčností jednotlivých koncových aplikací. Banka taktéž v rámci své architektury používá tzv. integrační platformu, místo, přes které mohou všechny aplikace komunikovat navzájem. Pro náš účel místo, přes které může cílově komunikovat front-end s jednotlivými aplikacemi. Banka již vlastní aplikaci na vytváření univerzálního front-endu. Tuto aplikaci používají interní bankovní zaměstnanci (FE specialisti) a vytváří samotný front-end pro všechny výše popsané stávající back-endové aplikace.

6.2 Zadání

Produktový tým banky ABC požaduje realizaci fornt-endové aplikace pro oddělení call-centra banky ABC. Hlavním cíli je zvýšit prodeje bankovních produktů, snížit náklady na provoz kamenných poboček, dát zaměstnancům jednodušší a přehlednou aplikaci pro práci.

Aplikace bude schopná komunikovat s integrační platformou a přes tu s koncovými aplikacemi. Pro správu uživatelských přístupů bude použita stávající aplikace banky. Cílově chce banka použít front-end na zastřešení nabídky a vytvoření běžného účtu, spořícího účtu a životního pojištění. Aplikace bude primárně pro call-centra, ale do budoucna bude využita v kamenných pobočkách banky

Požadované funkčnosti:

- Vyhledání, založení a úprava klientských dat
- Založení běžného účtu

- Založení spořícího účtu
- Založení životního pojištění

Nefunkční část řešení:

- Aplikace pro tvorbu front-endu musí být dostatečně jednoduchá, aby k jejímu vyvíjení nebyl potřeba programátor
- Výstupy z aplikace musí být pochopitelné zástupcům z obchodních oddělení firmy

Technické oblasti:

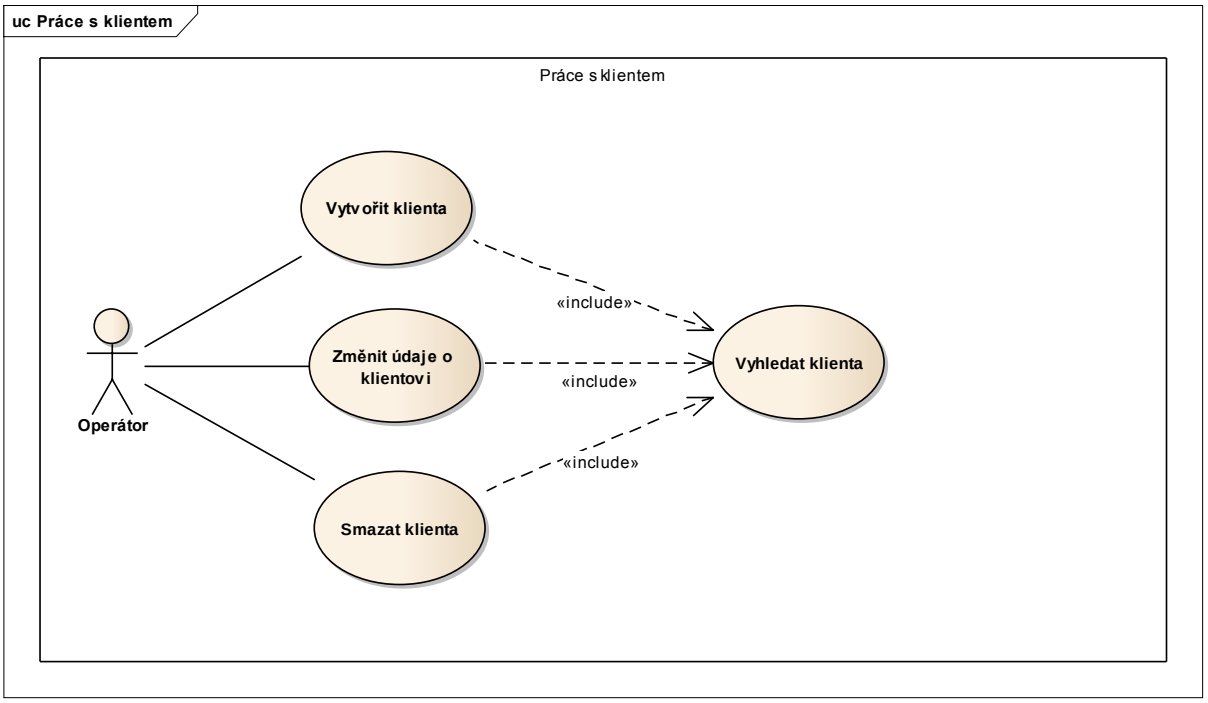
- Pro koncové uživatele musí aplikace fungovat a podporovat webové prohlížeče Internet Explorer, Mozzila Firefox a Google Chrome
- Technicky musí umět engine front-endové aplikace komunikovat s cílovou infrastrukturou (zajištění přenosu dat mezi aplikací pro koncové uživatele a databázemi)
- Propojitelnost a využití již existujících modulů pro správu uživatelských práv a automatický dohled nad provozem aplikace

6.3 Případy užití

Zde naleznete rozpad požadovaných funkčností na případy užití z pohledu jednotlivých aktérů.

Případy užití ukazují podrobnější pohled na relativně obecné zadání. Z praktického úhlu pohledu popisuji budoucí práci uživatelů se zamýšlenou navrhovanou front-end aplikací.

Případ užití práce s klientem je jednou ze dvou skupin případů užití, které navrhuji k pokrytí požadavků na funkčnost aplikace. Je rozpadem požadavku: vyhledání, založení a úprava klientských dat.



Obrázek 16 Případy užití – práce s klientem

6.3.1 Scénář k případu užití Vyhledat klienta

Hlavní cesta: Hledání dle rodného čísla

1. Na vyhledávácí obrazovce vybrat hledání dle rodného čísla.

Alternativní cesta: 1a. Hledání dle příjmení a adresy.

2. Zadat rodné číslo klienta do pole "rodné číslo" a stisknout "Vyhledat".

3. V případě nalezení klienta, načíst informace o něm a přesměrovat uživatele na obrazovku s detailem. V případě nenalezení klienta oznámit uživateli "Klient nenalezen",

Alternativní cesta: Hledání dle příjmení a adresy.

1. Na vyhledávácí obrazovce vybrat hledání dle příjmení a adresy.

2. Zadat údaje do pole "Příjmení" a polí s adresou "Ulice", "Číslo popisné", "Město" a stisknout "Vyhledat".

3. V případě nalezení klienta, načíst informace o něm a přesměrovat uživatele na obrazovku s detailem. V případě nenalezení klienta oznámit uživateli "Klient nenalezen". V případě nalezení více klientů se stejným jménem a adresou je zobrazit v seznamu. Operátor následně vybere dle doplňujících údajů správného klienta a je přesměrován na detail klienta.

6.3.2 Scénář k případu užití Smazat klienta

1. Na obrazovce detailu daného klienta vybere operátor "Smazat".
2. Objeví se obrazovka k potvrzení "Opravdu smazat klienta?" s možnostmi "Ano" a "Ne".
3. Pokud operátor vybere "Ano" opravdu smazat klienta, systém vyhodnotí, zda nemá nějaký aktivní produkt. Pokud jsou splněny všechny podmínky pro zrušení, systém klienta učiní nepřístupným, aby se na něj zaměstnanci již nemohli dostat.

6.3.3 Scénář k případu užití Vytvořit klienta

Hlavní cesta: Vytvořit klienta

1. Na úvodní obrazovce stisknout tlačítko "Založit nového klienta".
2. Vyplnit údaje o klientovi. Obrazovka bude obsahovat údaje: jméno, příjmení, adresu a rodné číslo klienta.
3. Stisknout tlačítko "Uložit klienta"

Alternativní cesta: 3a. Přerušit vytváření klienta.

4. Přejde na obrazovku s detailem klienta.

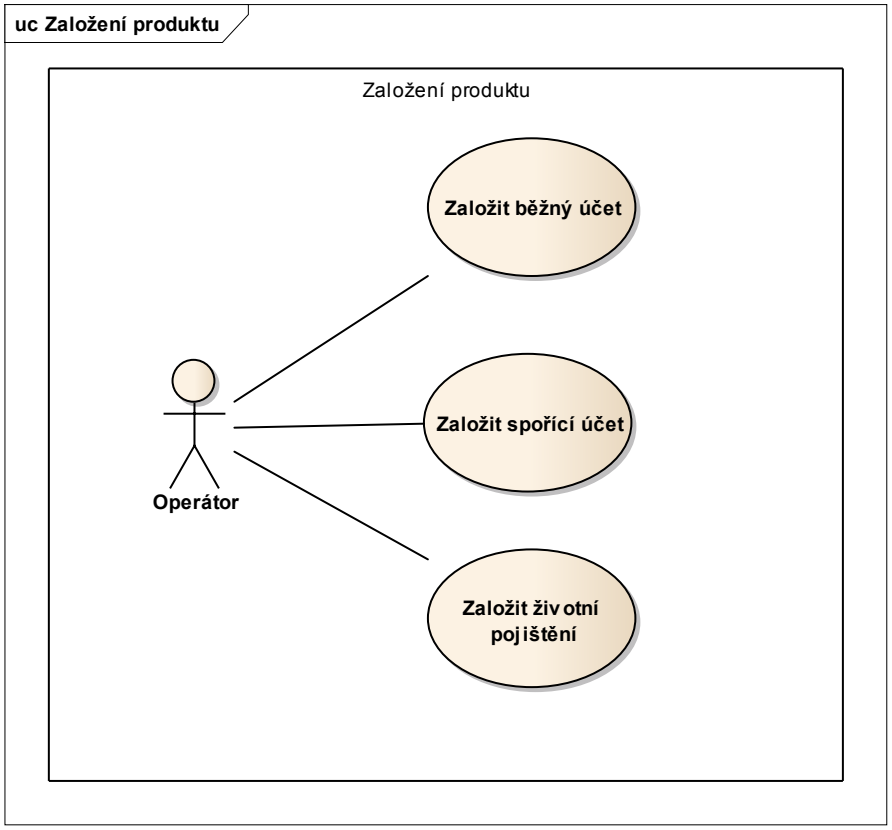
Alternativní cesta: Přerušit vytváření klienta.

Stisknout "Zrušit" uživatel ukončí zakládání nového klienta a vrátí se na hlavní obrazovku.

6.3.4 Scénář k případu užití Změnit údaje o klientovi

1. Operátor vybere na obrazovce klienta volbu "Změnit údaje".
2. Následně změní údaje a dá volbu "Uložit".

Případ užití Založení produktu je druhou skupinou případů užití, které navrhuji k pokrytí požadavků na funkčnost aplikace. Je rozpadem požadavků: založení životního pojištění, běžného a spořícího účtu.



Obrázek 17 Případy užití – založení produktu

6.3.5 Scénář k případu užití Založit běžný účet

1. Operátor na obrazovce detailu klienta vybírá založení běžného účtu.

2. Systém vygeneruje číslo účtu z volných pozic banky ABC. Současně založí všechny ostatní identifikátory nutné pro funkci běžného účtu a vygeneruje dopis na adresu klienta s údaji o běžném účtu.

6.3.6 Scénář k případu užití Založit spořící účet

1. Operátor na obrazovce detailu klienta vybírá založení spořícího účtu.

2. Systém vygeneruje číslo účtu z volných pozic banky ABC. Současně založí všechny ostatní identifikátory nutné pro funkci spořícího účtu a vygeneruje dopis na adresu klienta s údaji o spořícím účtu.

6.3.7 Scénář k případu užití Založit životního pojištění

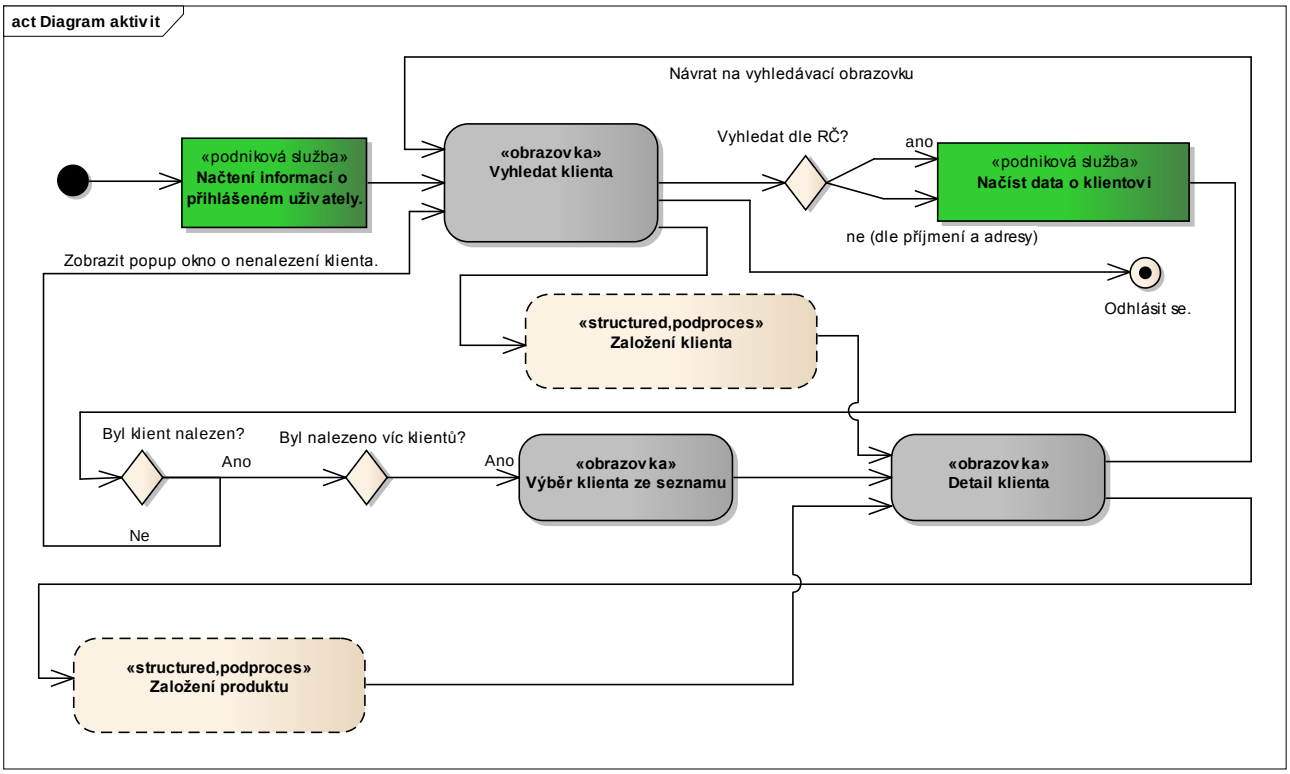
1. Operátor vybere možnost "Založit životní pojištění".

2. Otevře se nová obrazovka, při které operátor přečte klientovi podmínky smlouvy a sdílení informací o klientovi dané pojišťovně. Vyplní potřebná pole pro uzavření smlouvy k pojištění.
3. Následně systém zavolá podnikové službě pojišťovny XY. Pokud vše proběhne v pořádku, operátorovi se zobrazí obrazovka "Úspěšně založené životní pojištění". Pokud je nějaká chyba, zobrazí se důvod chyby.

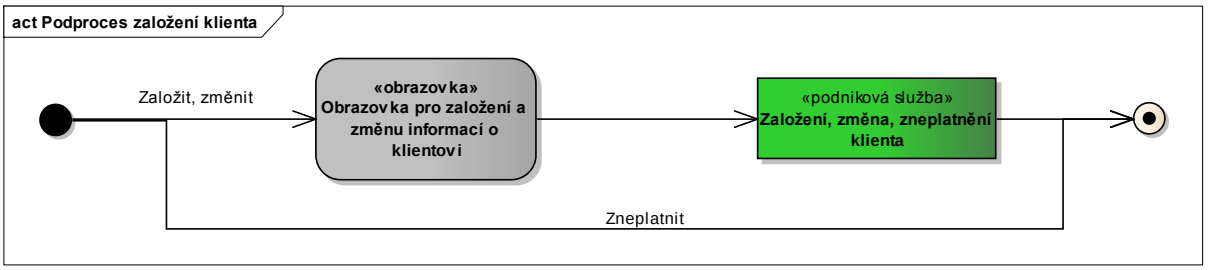
6.4 Diagramy aktivit

Diagramy aktivit dále rozpracovávají návrh budoucích podnikových procesů. Předpokládá se vlastnění nástroje, který umožní přímou implementaci z diagramů aktivit do samotných spustitelných procesů. Pro modelový příklad byl objekt aktivita doplněn stereotypem <<obrazovka>> s šedivou výplní a stereotyp <<podniková služba>> se zelenou výplní. Pro zanořování spustitelných uzavřených procesů byl upraven objekt strukturované aktivity a v modelu vystupuje jako stereotyp <<structured, podproces>>. V ostatních ohledech je použit standardní diagram aktivit dle UML 2.0 notace.

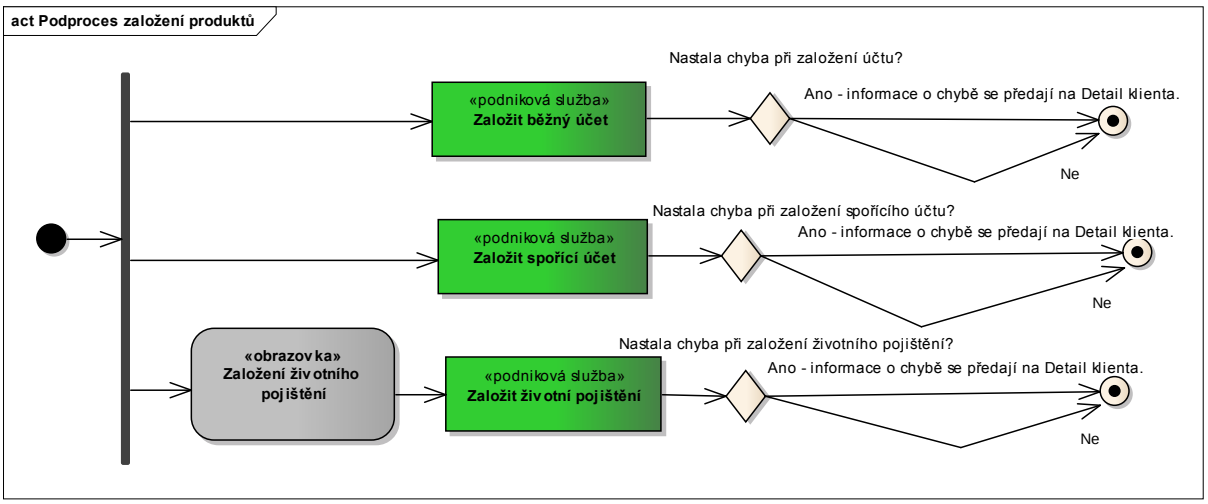
Návrh procesní části aplikace tvoří tři diagramy. Prvním je hlavní proces nazvaný jen Diagram aktivit, který obsahuje dva podprocesy Založení klienta a Založení produktů. Většina propojení mezi objekty tzv. information flow je oproti zvyklostem ve většině případů popsáno, aby bylo pro tento modelový případ zřejmější fungování navrhované aplikace.



Obrázek 18 Diagram aktivit



Obrázek 19 Založení klienta



Obrázek 20 Založení produktů

6.5 Parametrizace

Modelový příklad předpokládá, že podnik vlastní nástroj na implementaci aktivity diagramů jako spustitelných procesů. Předpokládejme, že máme nástroj velmi podobný těm popisovaným v kapitole Nástroje realizace front-end aplikací, s tím rozdílem, že využívají diagramy aktivit a ne BPMN. Taktéž mají moduly pro parametrizaci podnikových služeb a tvorbu obrazovek.

Vyjdeme-li z těchto předpokladů, dalšími pracovními úkony FE specialisty by byly návrhy obrazovek. Všude tam, kde je v části diagramů aktivit použit objekt <<obrazovka>>, musí specialista udělat grafický návrh obrazovek. Na každou obrazovku musí umístit potřebná pole pro čtení nebo zápis. Dalším krokem je samotná parametrizace. FE specialista nejprve nadefinuje parametry nutné pro zavolání objektů <<podniková služba>>.

Příkladem podniková služba Načtení dat o klientovi by měla na vstupu pole: rodné číslo, příjmení klienta, ulice, číslo popisné, město. Služba by vracela údaje pouze tehdy, když by bylo vyplněno rodné číslo nebo všechny zbylé položky vyjma rodného čísla. Výstupními údaji by byly pole: jméno, příjmení, rodné číslo, ulice, číslo popisné, město, číslo běžného účtu, číslo spořícího účtu, příznak zda má klient životní pojištění, aktuální balance na běžném účtu. FE specialista pak udělá parametrizaci a mapování polí z obrazovky Vyhledat klienta, tak aby dle typu hledání byly na vstupu podnikové služby zadané hodnoty z obrazovky. Taktéž nemapuje z výstupu služby hodnoty na obrazovky Výběr klienta ze seznamu a na Detail klienta.

6.6 Nasazení

Předpokládejme, že podnikový nástroj na vývoj front-end aplikace umožňuje i samotné nasazení (deployment) aplikace na zvolené prostředí. FE specialista by po validaci své práce ohledně návrhu obrazovek, namapování vstupních a výstupních polí a parametrizaci mohl přistoupit k nasazení na zvolené prostředí.

6.7 Testy

Jsme ve fázi, kdy máme funkční front-end aplikaci. V tomto kroku by měl FE specialista vyzkoušet základní případy užití vycházející ze scénářů a vymyslet potencionální chybové situace. Pokud aplikaci rozsahu našeho modelového příkladu

otestuje, předá jí zadavatelům nebo pověřeným uživatelům k otestování do akceptačních testů tzv. UAT (user acceptance test).

Následně FE specialista pomáhá s odstraněním chyb, dle závažnosti a rozsahu případné chyby mění buď návrh podnikových procesů (diagramy aktivit) nebo parametrizaci či obrazovky. Pokud je vše v pořádku (splňuje kvóty projektu na dosažení kvality), končí FE specialista svou práci na vývoji modelového příkladu front-endové aplikace.

Závěr

V rámci diplomové práce na téma vývoj front-endových aplikací jsem popsal celistvě veškeré kroky, které dovedou podnik k efektivnímu a úspěšnému provozu aplikací automatizujících podnikové procesy.

Třetí kapitolu jsem věnoval vývojovému cyklu, kde zejména kapitola o podnikové architektuře je vhodnou inspirací pro střední a velké podniky, jak nalézt odpověď na otázku, zda se jim vyplatí nakoupit nebo si interně vyvinout nástroj pro tvorbu automatizovaných podnikových procesů a samotných front-endových aplikací. Záměrně jsem zpracoval tuto architektonickou část, která podniku na jedné straně nabízí metodiky pro zavedení strategie ohledně obchodně technologického rozvoje, a jednou z odpovědí může být zavedení nástroje na snadnou automatizaci různorodých procesů. Na druhé straně zavedení podnikové architektury využije podnik podruhé při samotné realizaci jednotlivých front-end aplikací. Subjektivně doporučuji pro praktické použití adaptaci hlavních rysů z podnikové architektury FEA. Přes silné rysy architektury vládních organizací je velice prakticky a pragmaticky zaměřena a poměrně jednoduše ji lze upravit na míru podniku, který si výrazně zlepší zpracování informací o stávajícím stavu svých technologií a budoucích strategiích na rozvoj.

Další kapitoly představuje zhodnocení dvou aktuálních vývojových stylů uchopení životního cyklu softwaru, a to rigorózní a agilní metodiky. Přestože se celou prací orientuju na prokázání možnost efektivního vývoje front-end aplikací bez potřeby klasických vývojářů a tím být možná blíž zadavatelům, stále musíme počítat s tím, že potencionální oddělení na vývoj front-end aplikací bude součástí nějakého většího IT oddělení, které jistě bude vyznávat jeden z hodnocených vývojových cyklů. Taktéž pro samotný vývoj jednotlivých front-end aplikací bude nutné zvolit jednu z metodik.

V rámci celé práce popisuji doporučenou roli front-end specialisty. Jde o roli sdružující profesi analytika a částečně vývojáře s testerem. Standardy a nástroje, jaké by měla tato role používat, jsou uvedeny v kapitole analýza, respektive vývoj. Analytickými standardy ukazuju cestu, že není potřeba používat všemožné diagramy, které nabízí například UML. Na dostatečné zpracování informací zachycujících požadavky postačí jejich kvalitní sepsání a následný návrh případů užití. Pro samotný návrh a i předpokládané generování samotné aplikace potřebujeme nějaký dostatečně pružný standard pro

modelování dynamických situací. Pro tento účel vysvětluji základní vlastnosti diagramů aktivit a BPMN. Na prokázání možnosti generování z dynamických diagramů přímo funkční aplikace jsem zařadil kapitolu Nástroje realizace front-endových aplikací. V kapitole o nástrojích představuji šest aplikací, které si může podnik pořídit na automatizaci podnikových procesů a generování front-end aplikace. Taktéž však stručně konstatuju, že je možná varianta nechat si takovýto nástroj vyvinout, což používám jako předpoklad modelového příkladu.

Kapitola Modelové použití slouží jako důkaz o proveditelnosti teoreticky popisovaných technik za pomocí analytických standardů. Modelové použití vychází z fiktivního zadání, nicméně samotný způsob jeho řešení se blíží co nejvíce realitě. V kapitole prokazuju návaznost jednotlivých kroků. Všechny důležité kroky může provést front-end specialista. Samotný návrh procesů stejného rozsahu jako u modelového příkladu zabere čas v rámci maximálně jednoho dne a má vlastní praxe ukazuje, že následná parametrizace a návrh obrazovek je proveditelný v horizontu dalších dvou dnů.

Diplomová práce je inovativní ve skloubení technik na zpracování podnikových informací z roviny strategie technologického rozvoje až po techniky realizace front-end aplikací v rámci jednoho dokumentu. Zejména pak ve zcela novém návrhu vytvoření specializované role front-end specialisty, který komunikuje jak se zadavateli (zná jejich potřeby) a zároveň umí vytvářet front-endové aplikace.

Diplomovou prací jsem dosáhl cíle pokrýt informační potřeby podniku na zavedení a průběžný rozvoj front-endových aplikací. V jednotlivých kapitolách popisuji dle své praxe a aktuálních informačních zdrojů známé a aktuální techniky vývoje front-end aplikací. Modelovým použitím prokazuji praktikou realizaci, která ukazuje, že přes jistě větší vstupní investice existuje alternativa ke klasickému vývoji front-endových aplikací. Popisovaná řešení přinesou podniku možnost rychlé reakce na změny na trhu, dlouhodobě pomohou zlepšovat podnikové procesy a po vstupních investicích ušetří i na provozních a personálních nákladech.

7 Seznam použité literatury:

A Comparison of the Top Four Enterprise-Architecture Methodologies. *Developer Network* [online]. 2007, č. 1 [cit. 2013-08-01]. Dostupné z: http://msdn.microsoft.com/en-us/library/bb466232.aspx

Appian BPM Suite: One Complete Intelligent Business Process Management Suite. APPIAN. *Business Process Management (BPM) | Appian* [online]. 2013 [cit. 2013-07-30]. Dostupné z: http://www.appian.com/bpm-software/bpm-suite.jsp

ARLOW, Jim. UML a unifikovaný proces vývoje aplikací : průvodce analýzou a návrhem objektově orientovaného softwaru. Brno : Computer Press, 2003. 387 s. ISBN 80-7226-947-X

Automate manual processes with IBM BlueWorks Live. IBM. *IBM: Developer Works* [online]. 2011 [cit. 2013-07-30]. Dostupné z: http://www.ibm.com/developerworks/websphere/bpmjournal/1106_chang/1106_chang.html

BARBOSA, Daniel. Automating Functional Testing of Components From UML Specifications. *International Journal of Software Engineering & Knowledge Engineering*. 2007, roč. 17, č. 3, s. 339-358.

Bonita BPM, the Open Source BPM Suite | Bonitasoft. BONITASOFT. *Bonitasoft | Open Source Workflow & BPM software* [online]. 2013 [cit. 2013-07-30]. Dostupné z: http://www.bonitasoft.com/products/bonita-bpm-open-source-bpm-suite

BOTHA, G. J., P. S. KRUGER a M. DE VRIES. Enhancing Customer Experience Through Business Process Improvement: An Application of the Enhanced Customer Experience Framework (ECEF). *South African Journal of Industrial Engineering*. 2012, roč. 23, č. 1, s. 39-56.

BÖRGER, Egon. Approaches to modeling business processes: a critical analysis of BPMN, workflow patterns and YAWL. *Software & Systems Modeling*. 2012, roč. 11, č. 3, s. 305-318. DOI: 10.1007/s10270-011-0214-z.

BUCHALCEVOVÁ, Alena. *Metodiky vývoje a údržby informačních systémů: kategorizace, agilní metodiky, vzory pro návrh metodiky*. 1. vyd. Praha: Grada, 2005, 163 s. ISBN 80-247-1075-7.

BUNYAKIATI, P. a A. FINKELSTEIN. Standards compliance testing for unified modelling language tools. *IET Software*. 2011, roč. 5, č. 2, s. 120-131. DOI: 10.1049/iet-sen.2010.0032.

CORALLO, Angelo. Guidelines of a Unified Approach for Product and Business Process Modeling in Complex Enterprise. *Knowledge & Process Management*. 2011, roč. 18, č. 3, s. 194-206. DOI: 10.1002/kpm.381.

CUELLAR, Roland. Agile Software Development and Test Driven Development. *Journal of the Quality Assurance Institute*. 2006, roč. 20, č. 3, s. 33-38.

CURTIS, Bill, KELLNER, Marc, OVER Jim. Process Modeling. *Communications of the ACM*. 1992, roč. 35, č. 9, s. 75-90.

DIJKMAN, Remco M., Marlon DUMAS a Chun OUYANG. Semantics and analysis of business process models in BPMN. *Information & Software Technology*. 2008, roč. 50, č. 12, s. 1281-1294. DOI: 10.1016/j.infsof.2008.02.006.

EBSCO INDUSTRIES, Inc. *EBSCOhost: Základní vyhledávání* [online]. 2013 [cit. 2013-08-01]. Dostupné z: http://web.ebscohost.com.ezproxy.is.cuni.cz/

Federal Enterprise Architecture Framework [online]. 2. vyd. http://www.whitehouse.gov/, 2013 [cit. 2013-08-01]. ISBN N/A. Dostupné z: http://69.89.31.228/~mkerncom/wp-content/uploads/2013/02/Federal-Enterprise-Architecture-Framework-v2-as-of-Jan-29-2013.pdf

GAROUSI, Vahid. Classification and trend analysis of UML books (1997-2009). *Software & Systems Modeling*. 2012, roč. 11, č. 2, s. 273-285. DOI: 10.1007/s10270-011-0189-9.

GEAMBASU, Cristina Venera. BPMN vs. UML Activity Diagram for Business proces Modeling. *Accounting & Management Information Systems / Contabilitate si Informatica de Gestiune*. 2012, roč. 11, č. 4, s. 637-651.

GLASER, Armin. Safe Automation. *Control Engineering*. 2009, roč. 56, č. 8, A1.

GRÖNER, Gerd, BOŠKOVIĆ, Marko, SILVA PARREIRAS, Fernando. Modeling and validation of business process families. *Information Systems*. 2013, roč. 38, č. 5, s. 709-726. DOI: 10.1016/j.is.2012.11.010.

HAMDAN MOHAMMAD, Adel, ALWADA'N, Tariq, ABABNEH, Jafar "M.Ali". Agile Software Methodologies: Strength and Weakness. *International Journal of Engineering Science & Technology*. 2013, roč. 5, č. 3, s. 455-459.

International Journal of Production Research. 2009, roč. 47, č. 7. ISSN 00207543.

International Journal of Software Engineering & Knowledge Engineering. 2010, roč. 20, č. 7. ISSN 02181940.

ISO/IEC/IEEE 42010. Defining architecture. internet: DSCI, 2013. Dostupné z: http://www.iso-architecture.org/ieee-1471/index.html

JACOBSON, Ivar, Grady BOOCH, James RUMBAUGH. *The Unified Software Development Process*. Addison-Wesley Professional, February 14, 1999. 1 edition. ISBN 0201571692.

JACOBSON, Ivar, Grady BOOCH, James RUMBAUGH. *Brief Introduction to Unified Process* [online]. 1999. vyd. [cit. 2013-07-16]. Dostupné z: http://ai.ee.ccu.edu.tw/oose/Fall05/notes/briefRUP.pdf

JUAN, Y.-C. a C. OU-YANG. Systematic approach for the gap analysis of business processes. *International Journal of Production Research*. 2004, roč. 42, č. 7, s. 1325-1364. DOI: 10.1080/00207540310001631223.

Just Another Blog by Johnny [online]. 12. listopadu 2008, 12. listopadu 2008 [cit. 2010-11-30]. Klasické a agilní metodiky vývoje software. Dostupné z WWW: <http://johnny.bloguje.cz/740763-klasicke-a-agilni-metodiky-vyvoje-software.php>.

KNESL, Jiří. Agilní vývoj: Úvod. *Zdrojak.cz* [online]. 11.12.2009, NA [cit. 2013-07-17]. Dostupné z: http://www.zdrojak.cz/clanky/agilni-vyvoj-uvod/

KRISHNAMURTHY, Diwakar; SHAMS, Mahnaz; FAR, Behrouz H. A Model-Based Performance Testing Toolset for Web Applications. *Engineering Letters*. 2010, 18, 2, s. 92-106. ISSN 1816-093X.

LAM, Monica. Methodologies, tools, and techniques in practice for Web application development. *Journal of Technology Research*. 2012, roč. 3, s. 1-20.

LAM, Vitus S. W. Formal Analysis of BPMN Models:: A NuSMV-BASED APPROACH. *International Journal of Software Engineering & Knowledge Engineering*. 2010, roč. 20, č. 7, s. 987-1023.

LEVITAN, Alan S., Jian GUAN a Andrew T. COBB. Modeling an Object-Oriented Accounting System with Computer-Aided Software Engineering. *JOURNAL OF INFORMATION SYSTEMS*. 2008, roč. 22, č. 2, 123–139.

LIN, Huiping, FAN Yushun, NEWMAN Stephen. Manufacturing process analysis with support of workflow modelling and simulation. *International Journal of Production Research*. 2009, roč. 47, č. 7, s. 1773-1790. DOI: 10.1080/00207540701644151.

MILI, Hafedh, TREMBLAY Guy, JAOUDE Guitta Bou. Business Process Modeling Languages: Sorting Through the Alphabet Soup. *ACM Computing Surveys*. 2010, roč. 43, č. 1, s. 4-59. DOI: 10.1145/1824795.1824799.

MUKNŠNÁBL, Josef. *Reboot.cz* [online]. 6. Únor 2001 [cit. 2011-01-09]. Objektově orientované metody analýzy. Dostupné z WWW: <http://reboot.cz/howto/programovani/objektove-orientovane-metody-analyzy/articles.html?id=102>.

Oracle Application Development Framework - Oracle ADF. ORACLE. *Oracle | Hardware and Software, Engineered to Work Together* [online]. 2013 [cit. 2013-07-30]. Dostupné z: http://www.oracle.com/technetwork/developer-tools/adf/overview/index.html

Peixoto, D.C.C., Batista, V.A., Atayde, A.P. A Comparison of BPMN and UML 2.0 Activity Diagrams", VII Simpósio Brasileiro de Qualidade de Software: s.1-12.

PETERKA, Jiří. Front end. *EArchiv.cz: archiv článků a přednášek Jiřího Peterky* [online]. 1999, Roky 1991 až 1999, N/A, 2011 [cit. 2011-01-04]. Dostupné z: http://www.earchiv.cz/a94/a408c120.php3

Rapid App Development - See it Now|Create Enterprise Web and Mobile Apps with Intalio|Create. INTALIO. *Intalio | The modern way to build business applications* [online]. 2013 [cit. 2013-07-30]. Dostupné z: http://www.intalio.com/products/create/product-tour/

SALAHAT, Mohammed; WADE, Steve; UL-HAQ, Izhar . Applicationof a Systemic Soft Domain-Driven Design Framework. *ProceedingsofWorldAcademyof Science: Engineering& Technology*. Sep2009, Vol. 57, s. p476-486. Dostupný také z WWW: <http://www.waset.org/journals/waset/v57/v57-86.pdf>. ISSN 13076884

SARAN, Cliff. How to get the best from agile and waterfall development approaches. *Computer Weekly*. 2013, s. 4-5.

SCHWABER, Ken. The Scrum Guide: The Definitive Guide to Scrum: The Rules of the Game. *Scrum.org: Improving the Profession of Software Development* [online]. 2011, NA [cit. 2013-07-17]. Dostupné z: http://www.scrum.org/portals/0/documents/scrum%20guides/scrum_guide.pdf

Supported web service standards: IBM Business Process Management Reference Guide. IBM. *Help - Rational Requirements Composer* [online]. 2007 [cit. 2013-07-30]. Dostupné z: http://pic.dhe.ibm.com/infocenter/dmndhelp/v7r5m1/index.jsp?topic=%2Fcom.ibm.wbpm.ref.doc%2Ftopics%2Fradm_webservice_specs.html

ŠVENDOVÁ, V. Metodika agilního vývoje softwaru na OVSS ÚVT. *Zpravodaj ÚVT MU*. 2011, XXI, č. 4, s. 1-6. Dostupné z: http://www.ics.muni.cz/bulletin/articles/668.html

TENNANT, C. a P. ROBERTS. A faster way to create better quality products. *International Journal of Project Management*. 2001, Volume 19, Number 6, pp. 353-362(10). Dostupné z: http://www.ingentaconnect.com/content/els/02637863/2001/00000019/00000006/art00010

THE COMMON APPROACH TO FEDERAL ENTERPRISE ARCHITECTURE [online]. http://www.whitehouse.gov/, 2012 [cit. 2013-08-01]. ISBN N/A. Dostupné z: http://www.whitehouse.gov/sites/default/files/omb/assets/egov_docs/common_approach_to_federal_ea.pdf

TOGAF™ 9 and ITIL® V3: Two Frameworks Whitepaper. *Http://www.best-management-practice.com* [online]. 2013, č. 3 [cit. 2013-08-01]. Dostupné z: http://www.best-management-practice.com/gempdf/white_paper_togaf_9_itil_v3_sept09.pdf

Vodopádový model. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2013-07-16]. Dostupné z: http://cs.wikipedia.org/wiki/Vodop%C3%A1dov%C3%BD_model

Vytvoření definice obchodního procesu (BPD): IBM Business Process Manager verze 7.5.1, všechny platformy. IBM. *Help - Rational Requirements Composer* [online]. 2007 [cit. 2013-07-30]. Dostupné z:

http://pic.dhe.ibm.com/infocenter/dmndhelp/v7r5m1/index.jsp?topic=%2Fcom.ibm.wbpm.wle.editor.doc%2Fmodeling%2Ftopic%2Fcreating_bpd.html

What banking processes can benefit from BPM?. IBM. *Blueworks Live* [online]. 2013 [cit. 2013-07-30]. Dostupné z: https://www.blueworkslive.com/#!posts:33c0e3cfc

WILLIAMS, M. A., A. K. KOCHHAR a C. TENNANT. An object-oriented reference model of the fuzzy front end of the new product introduction process. *The International Journal of Advanced Manufacturing Technology* [online]. 2007, Volume 34, 826-841, Numbers 7-8 [cit. 2011-01-04]. Dostupné z: http://www.springerlink.com/content/x344742474717205/

WONG, W. Eric. TESTING ASPECT-ORIENTED PROGRAMS WITH UML DESIGN MODELS. *International Journal of Software Engineering & Knowledge Engineering*. 2008, roč. 18, č. 3, s. 413-437.

ZACHMAN, John A., 1987. A Framework for Information Systems Architecture. *IBM System Journal*. Roč. 26, č. 3, s. 276-292. ISSN 18-870.

ZACHMAN, John A., 2008. John Zachman`s ConciseDefinitionofTheZachman Framework™. Zachman International, Inc. Dostupné z: http://www.zachman.com/about-the-zachman-framework

ZACHMAN, John A., 2009. Yes, "Enterprise Architecture Is Relative" BUT It Is Not Arbitrary. Zachman International, Inc. Dostupné z: http://www.zachman.com/ea-articles-reference/57-eanotarbitrary

8 Seznam obrázků:

Obrázek 1 Zachmanova matice 10
Obrázek 2 Zachman - úrovně abstrakce 11
Obrázek 3 TOGAF proces řízení změn 13
Obrázek 4 TOGAF schéma dokumentace pro jednotlivé části projektu 16
Obrázek 5 FEA hlavní schéma architektonického přístupu 19
Obrázek 6 FEA úroveň záběru versus oblasti dopadu 22
Obrázek 7 FEA požadované artefakty 26
Obrázek 8 FEA doporučené artefakty pro strategii 26
Obrázek 9 FEA doporučené artefakty pro obchodní služby 27
Obrázek 10 FEA doporučené artefakty pro data a informace 27
Obrázek 11 FEA doporučené artefakty pro aplikace 28
Obrázek 12 FEA doporučené artefakty pro infrastrukturu 29
Obrázek 13 FEA doporučené artefakty pro bezpečnost 29
Obrázek 14 Základní schéma vodopádového modelu 38
Obrázek 15 Pracovní postupy v iteraci 40
Obrázek 16 Případy užití – práce s klientem 71
Obrázek 17 Případy užití – založení produktu 73
Obrázek 18 Diagram aktivit 75
Obrázek 19 Založení klienta 75
Obrázek 20 Založení produktů 75