MASARYKOVA UNIVERZITA
FAKULTA INFORMATIKY

Použitie BPMN pre male SW projekty

DIPLOMOVÁ PRÁCA

Bc. Miroslav Ligas

Brno, jaro 2009

Prehlásenie

Prehlasujem, že táto diplomová práca je mojím pôvodným autorským dielom, ktoré som vypracoval samostatne. Všetky zdroje, pramene a literatúru, ktoré som pri vypracovaní používal alebo z nich čerpal, v práci riadne citujem s uvedením úplného odkazu na príslušný zdroj.

Vedúci práce: Mgr. Tomáš Ludík

Poďakovanie

Dakujem za všetko čo pre mňa urobili ...

Zhrnutie

aa

Kľúčové slová

BPMN, UML

Obsah

Kapitola 1

Úvod

S riešením softvérových projektov vznikajú rôzne problémy, ktoré môžu viesť k zlyhaniu projektu. Tieto riziká sa pri vývoji snažíme odstrániť zavedením metodík, ktoré nám napomáhajú uchopiť projekt, rozanalyzovať problematické miesta a čo najlepšie navrhnúť riešenie. Žiadna metodika nám nezaistí splniteľnosť projektu ale jej použitie minimalizuje riziko krachu projektu.

Najpoužívanejšie modelovacie metodiky súčasnej doby sú veľmi rozsiahle a silné nástroje. Definujú veľké množstvo roli a zavádzajú komplexné procesy, čím dokážu zvládať veľké projekty. Vnášajú tým do vývoja veľkú réžiu, ktorá projekt pomáha lepšie zvládať ale ho aj predlžuje. Čím je projekt menší tým je citeľnejšia záťaž komplexnej metodiky. Opomenutie metodík by zbavilo projekty všetkej réžie a ušetrilo by čas aj prostriedky, ale riziko, ktoré by vzniklo by mnohonásobne prevýšilo úsporu.

Pri modernom vývoji nie je preto rozumné postupovať bez metodik pri akomkoľvek vývoji. Napriek tomu sa naskytuje priestor na hľadanie nových ciest pri ich riešení. Namiesto využívania rozsiahlych používaných a overených metodik sa treba zamerať na ich esenciálne časti. Na základe týchto častí sa vybuduje ľahko zvládnuteľná a flexibilnú metóda.

Malé softvérové projekty sú väčšinou spracúvané neveľkým počtom pracovníkov ako na strane vývojára tak na strane klienta. Ukazuje sa tu preto miesto pre rýchlu a flexibilnú metódu, ktorá dokáže rýchlo produkovať funkčné moduly a flexibilne reagovať na požiadavky klienta. Cieľom tejto práce je nájsť stanovenú metódu.

1.1 Štruktúra práce

Druhá kapitola práce sa zaoberá najpoužívanejšími modelovacími nástrojmi, ktoré sa v súčasnosti používajú na zachytenie interakcie a stavu v danom systéme. Podrobne sa tu popisuje rozšírený no možno nie tak notoricky známi nástroj na modelovanie firemných procesov Business Process Modeling Notation (BPMN). Okrajovo sa spomína aj Unified Modeling Language (UML). Popisované sú len niektoré prvky UML, s ktorými sa v práci stretneme.

Tretia kapitola je venovaná metodikám. Popisuje rôzne prístupy ako riešiť budovanie systému. Zaoberá sa agilnými metodikami, ktoré sa vyznačujú rýchlosťou a flexibilitou. V kontraste k nim stoja tradičné štruktúrované metodiky ako Unified Process (UP), ktoré stavajú na definovaných postupoch a roliach. Podrobnejšie sa venujeme najmä tým, z ktorých čerpáme inšpiráciu pre zostavenie vlastnej metódy.

Štvrtá kapitola zachytáva hlavnú časť práce a to definovanie metódy pre malé softvérové projekty s využitím BPMN. V tejto kapitole sa uplatňujú nástroje a postupy definované v predchádzajúcich oddieloch. Metóda je zostavená zo zaužívaných metodík, z ktorých sa

vyberajú relevantné časti. Spája sa v nej agilný prístup a tradičné štruktúrované metodiky. Metoda čerpá inspiráciu z Business Driven Development (BDD) z dielne IBM. Na zachytenie požiadavkov a identifikáciu modulov v projektovanom systém používa hierarchiu BPMN diagramov. Jednotlive moduly sú následne modelované za pomoci tradičných UML diagramov.

Záverečná piata kapitola overuje vhodnosť definovanej metódy. Z jej využitím je vytvorená prípadová štúdia popisujúca správu vedeckého časopisu. Pomocou uvedených nástrojov modeluje hierarchiu procesov prebiehajúcich pri fungovaní správy vedeckého časopisu. Vo vzniknutej procesnej mape sú identifikované procesy, ktoré je možné automatizovať. Následne sú určene komponenty, ktoré sú pomocou UML modelované a na záver je načrtnutá implementácia.

Kapitola 2

Modelovacie nástroje

2.1 Business Process Modeling Notation

V roku 2004 bol Business Process Management Initiative (BPMI) vydaný štandard BPMN 1.0. Cieľom tohoto štandardu je poskytnúť ľahko pochopiteľnú notáciu pre všetkých užívateľov podieľajúcich sa na tvorení, implementácii, spravovaní a monitorovaní firemných procesov. Súčasťou BPMN je aj interný model, ktorý umožňuje prevod na spustiteľný BPEL4WS kód. Vypĺňa sa tým medzera medzi firemným procesným návrhom a implementáciou.

BPMN definuje Business Process Diagram (BPD), ktorý graficky znázorňuje postupnosti firemných procesov. Objekty zachytené v grafe reprezentujú aktivity a orientované hrany naznačujú poradie ich vykonania.

2.1.1 Rozdelenie objektov v BPMN

BPD diagramy sú tvorené z jednoduchých elementov, ktoré umožňujú ľahké tvorenie diagramov, ktoré sú intuitívne pochopiteľné väčšine podnikových analytikov. Tvary elementov boli navrhnuté s ohľadom na už používané nástroje v procesnom modelovaní. Napríklad aktivity sa znázorňujú pomocou štvoruholníka a rozhodnutia sú značené diamantom. Pri vývoji BPMN bol kladený doraz aby bolo možné pomocou neho zachytiť aj komplexné firemné procesy. Pre lepšie zvládnutie týchto protichodných požiadavkov bolo navrhnutých malé množstvo kategórii, ktoré napomáhajú k ľahkej orientácii v základných typoch. V rámci každej zo základných kategórii je možné modifikovať definované elementy rozširujúcimi informáciami. Rozšírenia však nesmú narušovať základné charakteristiky elementov, čím by znižovali ich zrozumiteľnosť.

Základne kategórie elementov sú:

- Tokové objekty (Flow Objects)
- Spojovací objekty (Connecting Objects)
- Plavecké dráhy (Swimlanes)
- Artefakty (Artifacts)

2.1.2 Tokové objekty

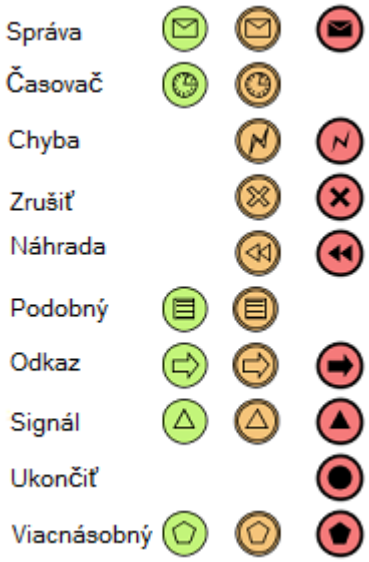
Tokové objekty sú základné objekty BPD udávajú správanie firemného procesu. Definované sú tri Flow Objects:

Udalosť (Event)

Udalosť je reprezentovaná krúžkom a vyjadruje niečo čo sa stalo počas chodu firemného procesu. Tieto udalosti ovplyvňujú tok procesu a obyčajne majú príčinu (spúšťač) alebo dôsledok (výsledok). Definované sú tri typy udalostí. Na začiatku toku sa umiestňujú štartovacie (Start) v priebehu používame medziľahlé (Intermediate) a tok ukončujeme koncovými (End) udalosťami.

Pre každú udalosť môžeme do krúžku umiestniť symbol upresňujúceho spúšťača alebo výsledku. V BPMN je definovaných desať podtypov udalostí. Nie všetky podtypy sa však môžu používa s každým typom udalostí. Na obrázku [Obr. 2.1] sú znázornené všetky podtypy.

Medziľahlé udalosti môžeme naviazať na aktivity čím určíme ich alternatívne ukončenia. Napríklad vypršanie času pre ukončenie aktivity, výskyt chyby počas behu a iné.



Obrázok 2.1: Zoznam podtypov udalostí.

Aktivita (Activity)

Aktivita je spoločný pojem pre činnosť ktorá prebieha vo firme. Aktivita môže byť atomická alebo nie je atomická. Typy aktivít sú proces (Process), podproces (Sub-Process) a úloha

(Task). Podproces a úloha sa značia štvorcom so zaoblenými rohmi, pričom podproces je odlíšený malým plusovým znamienkom v spodnej časti. Proces je obsiahnutý vo vnútri poolu.

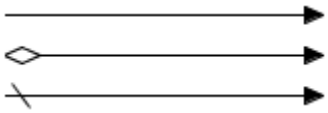
Brána (Gateway)

Bránou sa zabezpečuje rozdelenie a zlúčenie sekvenčného toku. Reprezentuje ho diamant, v ktorého strede sú zobrazene spresňujúce symboly. Tie špecifikujú o aký typ vetvenia sa jedna. Na výber mame z rozhodovacieho alebo paralelného delenia. K týmto deleniam sú definovane odpovedajúce zlučovania.

2.1.3 Spojovací objekty (Connecting Objects)

Tokové objekty sa spájajú a vytvárajú základnú kostru firemného procesu. Spojovací objekty zabezpečujú toto prepojenie a taktiež umožňujú pripájanie artefaktov.

Sekvenční tok (Sequence Flow)



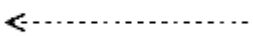
Sekvenční tok reprezentuje neprerušovaná čiara s vyplnenou šípkou. Vyznačuje poradie v akom sú aktivity vykonávané v procese.

Tok sprav (Message Flow)

Tok sprav reprezentuje prerušovaná čiara s prázdnou šípkou. Používajú sa na znázornenie interakcie medzi dvomi separátnymi účastníkmi procesu, ktorý sú schopný prímať a odosielať správy.

Tok správ môže byť podmienený. Graficky sa podmienená správa označí umiestnením diamantu na začiatok čiary a podmienka je zaznamenaná v názve toku. Pri používaní podmienených správ je dôležité aby aspoň jedna podmienka bola splnená aby proces mohol pokračovať. Pre zaistenie pokračovania toku sa môže použiť implicitný tok, ktorý sa značí prešKrtnutím na začiatku čiary. Uplatňuje sa ak všetky ostatne podmienené spravy sa vyhodnotia záporne.

Asociácia (Association)



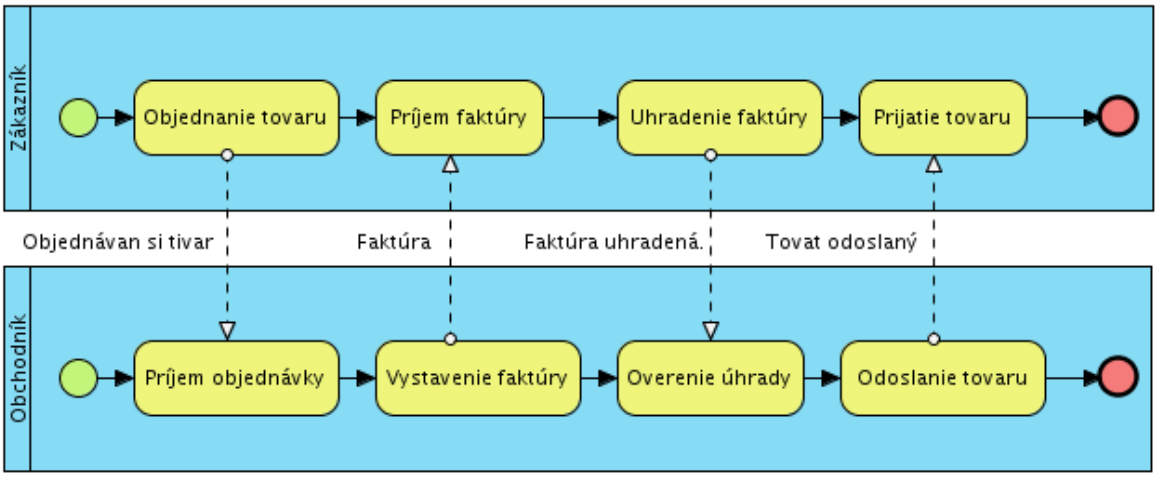
Na znázornenie asociácie sa používa prerušovaná čiara. Pomocou nej priraďujeme k tokovým objektom textové popisky alebo iné objekty ktoré nepatria do skupiny tokových objektov. Pridaním šípky k prerušovanej čiare môžeme určiť smer priradenia.

2.1.4 Plavecké dráhy (Swimlanes)

Plavecké dráhy sa používajú ako prostriedok pre organizáciu aktivít. Opticky sa pomocou nich oddeľujú zodpovednosti roli alebo usporiadanie činnosti v procese.

Pool

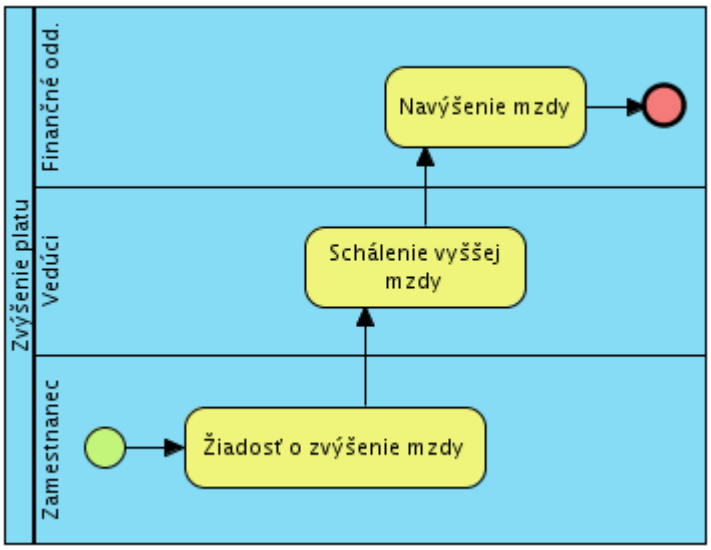
Pool ohraničuje proces a graficky vymedzuje jeho hranice. V rámci jedného poolu sa nachádza len jedne proces. Interakcia medzi poolmi prebieha pomocou správ. Poolý sa v diagrame používajú pre zachytenie dvoch separátnych firemných entít alebo účastníkov. Proces každého účastníka je uzavretý v jeho poole čím je stanovené jeho jasné ohraničenie. Zachytený proces nemôže interagovať z okolitými procesmi pomocou sekvenčných tokov. Pre interakciu medy dvomi poolmi je určený mechanizmus toku správ. Správy nesmú byť použité v rámci jedného poolu. [Obr. 2.2]



Obrázok 2.2: Príklad využitia poolu.

Dráha (Lane)

Dráha delí pool na menšie časti po celej jeho dĺžke. Slúžia na usporiadanie a kategorizáciu aktivít. Môžu napríklad znázorňovať role, oddelenia alebo funkcie organizácie. Komunikácia medzi jednotlivými dráhami prebieha pomocou sekvenčných tokov. Toky správ sa nesmú používať na komunikáciu medzi tokovými objektmi v dráhach jedného poolu. [Obr. 2.3]



Obrázok 2.3: Príklad využitia dráhy.

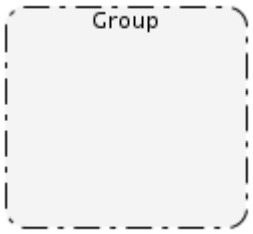
2.1.5 Artefakty (Artifacts)

Artefakty neovplyvňujú základnú štruktúru procesu budovanú z aktivít, brán a sekvenčných tokov. Ponúkajú však spresňujúce informácie o elementoch procesu. Užívateľ si môže sám doplniť sadu artefaktov pre uľahčenie a sprehľadnenie diagramov. V BPMN sú preddefinované len tri typy artefaktov.

Dátový objekt (Data Object)

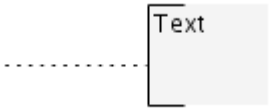
Dátový objekt slúži na zobrazenie toku dat v procese. Pomocou neho modelujeme aké dáta sú požadované a aké dáta systém produkujeme. Dátový objekt je k aktivitám pripájaný pomocou asociácie. Graficky je reprezentovaný obdĺžnikom s ohnutým rohom.

Skupina (Group)



Skupina je znázorňovaná prerušovaným obdĺžnikom a používa sa pre dokumentačné a analytické účeli. Nejak neovplyvňuje tok procesu.

Poznámka (Annotation)



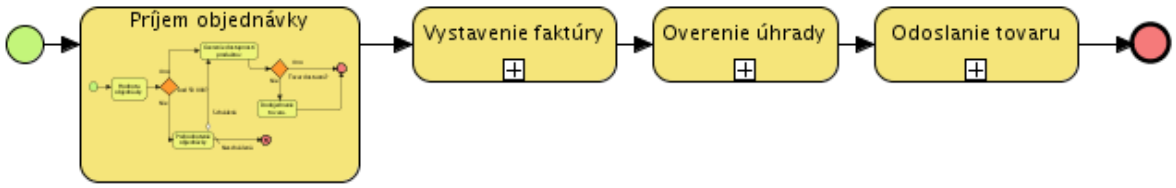
Anotácia slúži na zachytenie dodatočnej textovej informácie pre čitateľa diagramu. K objektu je pripojená pomocou asociácie.

2.1.6 Využitie BPMN

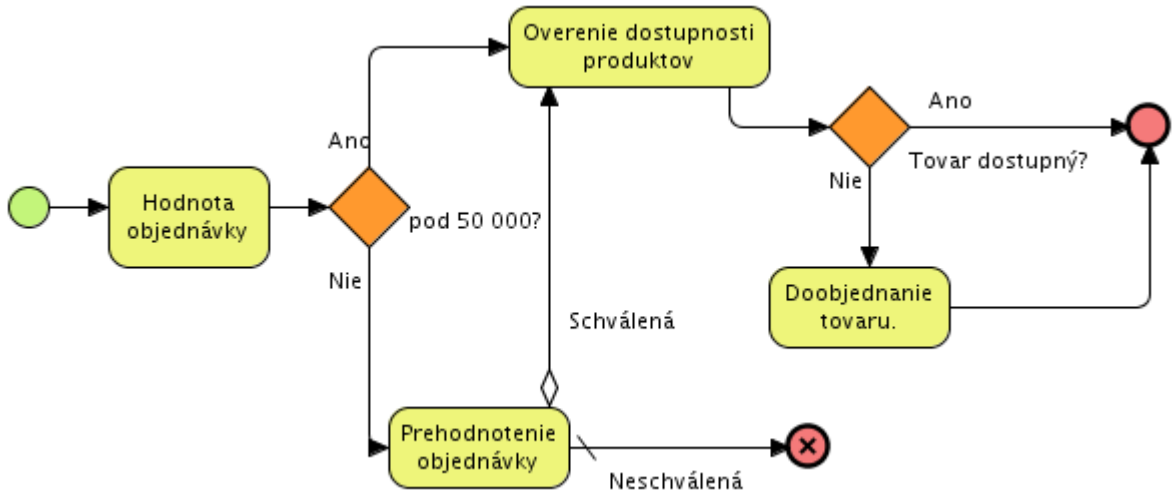
Pomocou BPMN sme schopný modelovať procesy na rôznych úrovniach. Od detailného popisu jednotlivých čiastočných procesov ku globálnej orchestrácii firemných procesov, ktoré sa javia ako čierna skrinka. BPMN tým oslovuje rôznorodé publikum, ktorému predáva široké spektrum informácii na rozličných úrovniach detailu. Podla miery záberu moderovaných procesov sa BPD delia na dve základné skupiny.

Kooperatívne medzi-firemné procesy Kooperatívny typ diagramu zachytáva medzi-firemné procesy. Jeho hlavným cieľom je znázornenie vzťahov medzi dvomi a viacerými procesmi. Dôraz je kladený na modelovanie vzájomnej komunikácie. Obrázok 2.2 je príkladom kooperatívneho medzi-firemného procesu.

Interné firemné procesy Interné procesy firmy sú zachytene v hierarchii diagramov. Najvyššia úroveň zachytáva hlavný firemný proces, ktorý je skrz podprocesy podrobne popísaný. Najnižšia úroveň podrobne modeluje všetky činnosti, ktoré prebiehajú v procese. Príkladom procesu vysokej úrovne je obrázok 2.4. Diagram sa skladá z podprocesov, ktoré reprezentujú nižšie úrovne. Rozkreslením niektorého z podprocesov dostávame podrobný diagram jeho fungovania. Takýto diagram je znázornení na obrázku 2.5.



Obrázok 2.4: Interný proces vysokej úrovne.



Obrázok 2.5: Interný proces vysokej úrovne.

2.2 Unified Modeling Language

Unified Modeling Language (UML) je v súčasnej dobe najrozšírenejších modelovacích nástrojov. Jeho uplatnenie je široké a nie je používaný len v oblasti vývoja softvéru. Pre jeho veľké rozšírenie a predpokladanú zrejmosť notácie sa nim táto práca nebude podrobne zaoberať. Zmienené budú len niektoré diagramy, ktoré v práci využijeme.

2.2.1 Diagram prípadov užitia

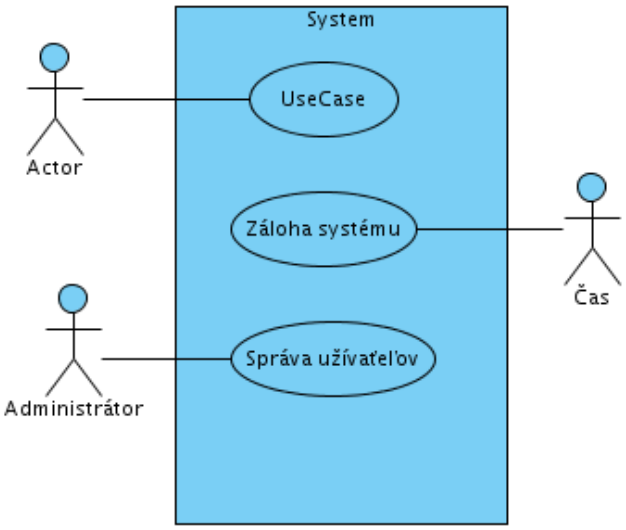
Diagram prípadov užitia (Use Case Diagram) slúži na zachytenie základných funkčných požiadavkov, ktoré ma systém splňovať. Diagram sa skladá z troch základných častí:

Hranice systému Vymedzujú modelovanú oblasť.

Aktér Predstavuje entitu (rola, systém, čas) mimo systém, ktorá so systémom spolupracuje.

Prípad užitia Zachytáva ucelenú funkčnú jednotku systému.

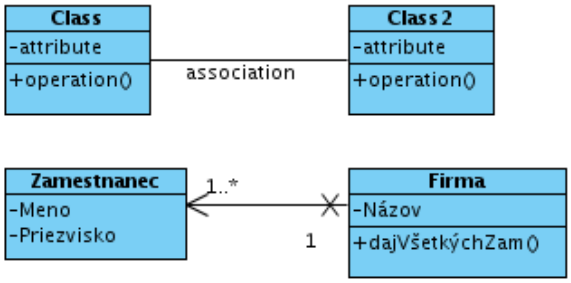
Pomocou týchto častí modelujeme interakciu okolitého sveta s modelovaným systémom. Diagram nesmie obsahovať interakciu medzi aktérmi a ani medzi prípadmi užitia. Súčasťou diagramu je aj detailná dokumentácia jednotlivých prípadov užitia. Na obrázku 2.6 je jednoduchý príklad diagramu.



Obrázok 2.6: Príklad na diagram prípadu užitia.

2.2.2 Diagram tried

Diagram tried graficky zachytáva statickú štruktúru systému. Jeho stavebnými prvkami sú triedy a asociácie. Triedy obsahujú atribúty a operácie a môžu sa hierarchicky radiť pomocou generalizácie a špecializácie. Asociácii slúžia na modelovanie vzájomných vzťahov medzi tried. Môžeme určiť ich smer a kardinalita. Na obrázku 2.7 je príklad diagramu tried.



Obrázok 2.7: Príklad na diagram tried.

Kapitola 3

Prístupy vyvoja softvéru

V súčastnosti sa pri vývoji softvérových produktov v drvivej väčšine prípadov používa objektovo orientovaná analýza a návrh. V tomto prístupe sa entity modelovaného systému reprezentujú pomocou objektov, ktoré zachytávajú ich stav, správanie sa a identitu. Pre uľahčenie zvládnutia problému sa využívajú rôzna úroveň abstrakcie. Zakrývajú sa ňou nepodstatných časti problému a sústredí sa pozornosť na podstatne aspekty.

Pri využití objektovo orientovanej analýzy sa v najväčšej miere používajú dva modely: Vodopád a iteratívny/inkrementálny vývoj. Črty týchto modelov nájdeme vo všetkých moderných modelovacích metodikách.

3.1 Vodopád

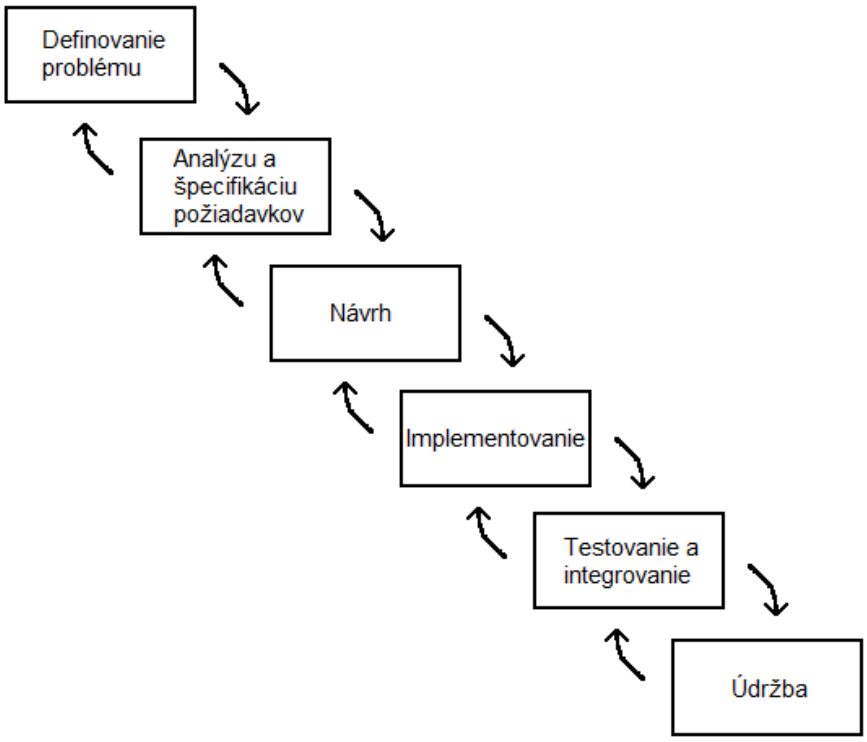
Vodopád patrí medzi najtradičnejšie modely vývoja. Historicky je to prvá ucelené metodika pre vývoj softvéru. Definuje jasné postupy pri zvládaní projektu počas celého jeho životného cyklu. Rozdeľuje projekt na základe vykonávaných aktivít na:

- Definovanie problému
- Analýzu a špecifikáciu požiadavkov
- Návrh
- Implementovanie
- Testovanie a integrovanie
- Údržba

Tieto etapy sa zoradia za seba a postupne sa začnú sekvenčne vykonávať. Ďalšia aktivita môže začať až keď skončí predchádzajúca. Pri výskyte chyby sa projekt vracia späť do etapy, v krtej chyba vznikla a musí byť opravená. Po opravení chyby sa proces spúšťa od toho miesta kde chyba nastala. 3.1

Výhodou aj nevýhodou vodopádu je jeho jednoduchosť a ľahká pochopiteľnosť. Umožňuje ľahkú kontrolu postupu práce pomocou sledovania výstupov jednotlivých etáp vývoja. V súčasnej dobe však už nedokáže pokryť väčšie projekty pre ich zložitosť.

Pri práci podla modelu vodopád sa rýchlo narazí na viaceré úskalia. Jedným z nich je správne odhadnutie času prechodu z jednej etapy projektu do nasledujúcej. Problémom môže byť aj to že vodopád neumožňuje prekrývania sa etáp. Najväčším problémom však je neskoré odhalenie chyby v analýze alebo návrhu, ktoré sa prejaví až pri testovaní. Takáto



Obrázok 3.1: Schéma životného cyklu Vodopád.

chyba vracia projekt na jeho úplný začiatok a môže ľahko viesť k jeho neúspechu. Na tieto ťažkosti naväzuje nakoniec problematický odhad ceny projektu.

Spôsob akým vodopád funguje prináša ešte jednu veľkú nevýhodu. Počas celej doby trvania vývoja nemá zákazník žiadnu možnosť zistiť či dodaný systém bude odpovedať jeho predstavám a zasahovať do jeho vývoja. Nemá možnosť získať funkčné podčasti systému, s ktorými môže už pracovať, ale musí čakať až na ukončenie vývoja. Po ukončení projektu ľahko nastane situácia, pri ktorej je zákazník prekvapený z výsledku ktorý obdrží.

3.2 Iteratívny / inkrementálny vývoj

Iteratívny / inkrementálny vývoj sa snaží o zníženie rizika zlyhania projektu. Celý projekt je rozdelený na časti podla budúcej funkcionality systému. Pre každú časť sa vykoná analýza, návrh, implementácia a testovanie. Výsledný systém sa vybuduje z podčastí. Rozdelením projektu umožníme skoršiu detekciu chýb a hlavne nemusíme prerábať celý systém ale len časť, v ktorej sa chyba vyskytla.

Pri využití tohoto modelu vznikajú problémy s integráciou. Vzniká réžia, ktorá musí zabezpečovať funkcionalitu neúplného systému ak napríklad vytvorenie protéz. Tvorí sa priestor na vznik nových chýb pri integrácii častí systému. Okrem technických problém sa komplikuje aj časový návrh práce na projekte, lebo nie každá iterácia zaberá rovnaký čas. Úvodné iterácia sa predlžujú z analytických dôvodov aby bol nastávajúci systém dobre pochopený. Záverečné iterácie zas v sebe zahrňujú sprievodné činnosti projektu ako zaškolenie užívateľov.

Pomocou inkrementálneho vývoja je systém tvorený z nezávislých funkčných častí, ktorá sú osobitne vytvárane za pomoci vodopádu alebo iteratívneho vývoja. Celkový systém dostávame spojením jednotlivých častí do jedného celku. Inkrementálny vývoj je založený na filozofii pridávania k existujúcim častiam systému.

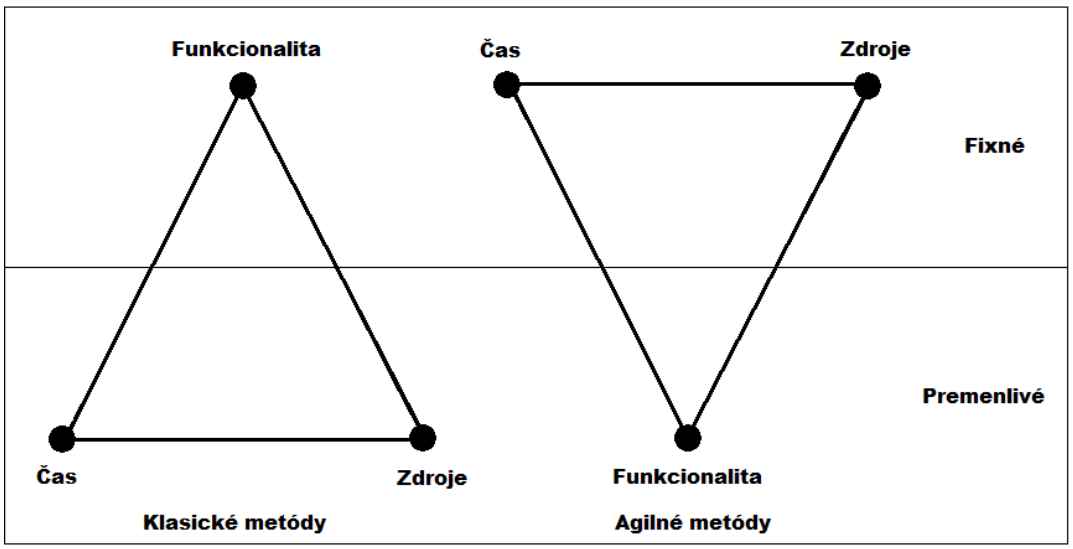
Pri iteratívnom vývoji sa postupuje cestou zdokonaľovania, rozširovania a opravovania už existujúceho systému. Nemalá časť kódu je ďalšími iteráciami pripisovaná prípadne umazaná a nahradená. Preferuje sa prepísanie zlého kódu namiesto jeho obchádzania. Tento postup sa v drvivej väčšine prípadov spája s inkrementálnym vývojom a veľmi dobre spolu fungujú.

3.3 Agilné metódy vývoja

Súčasný svet sa veľmi rýchlo mení. Pri vývoji softvéru sa preto kladie veľký doraz na rýchlosť a flexibilitu. Aplikácie sa počas vývoja musia prispôsobovať meniacim sa podmienkam a byť čo najskôr k dispozícii zákazníkovi.

Z obrázku 3.2 vidíme, že klasické metódy vývoja zakladajú na fixnej funkcionalite, ktorá je daná špecifikáciou požiadavkov. Funkcionalita je hlavným merítkom úspešnosti projektu. V prípade nesplnenie požiadavkov projekt zlyháva. Čas a prostriedky sa pre daný projekt odvíjajú od funkcionality. Často sa preto stáva, že dochádza k posúvaniu termínov odovzdania a navyšovanie zdrojov na zvládnutie projektu.

Agilné metódy sa k problému stavajú presne naopak. Za fixné považujú zdroje a čas potrebný na zvládnutie projektu. Zákazníkovi dodáva nie vždy úplnú, ale pre neho najpodstatnejšiu funkcionalitu vždy v čase, keď ju potrebuje. Programy vyvíjané agilnými metódami sú ľahko rozšíritelne a preto nie je problém s dodaním zvyšných častí systému. Ďalším dôsledkom nefixnej funkcionality je možnosť menenia požiadavkov zákazníka počas vývoja. Vyvíjaný produkt tým lepšie spĺňa zákazníkove potreby.



Obrázok 3.2: Porovnanie klasického a agilného prístupu.

3.3.1 Manifest agilného programovania

Medzi agilná metódy vývoja patria viaceré metodiky. Všetky majú spoločný základ vo častých kontrolách a úpravách, vysoko kvalifikovaných samo-organizovaných týmoch a zainteresovanosti zákazníka v procese. Filozofia agilných metód je zachytená v ich manifeste:

- Osoby a interakcia majú prednosť pred procesmi a nástrojmi
- Fungujúci softvér ma prednosť pred obsiahlou dokumentáciou
- Spolupráca so zákazníkom je uprednostnená pred vyjednávaním zmluv
- Reagovanie na zmenu má prednosť pred nasledovaním plánu

Osoby a interakcia majú prednosť pred procesmi a nástrojmi

Tento bod vyjadruje zameranie metódy na človeka a jeho schopnosti a skúsenosti. Uprednostňovaný je malý efektívny tým, ktorého členovia medzi sebou intenzívne komunikujú pri riešení problémov. Preferované je riešenie problémov pomocou komunikácie tvárou v tvár, ktorá je rýchlejšia a efektívnejšia ako iné formy komunikácie. Je preto dobré aby bol cely tým situovaný fyzicky na jedno miesto. Umožňuje to lepšiu spoluprácu na projekte a tým aj rýchlejšie produkovanie kódu. Programovanie často prebieha v skupinkách, ktorej členovia sa striedajú pri kódovaní daného úseku programu.

Manažéri a vývojári sú v týme na rovnakej úrovni. Pri rozhodnutiach, ktoré sa týkajú technických riešení, majú hlavné slovo vývojári. Manažéri majú za úlohu odstraňovať problémy netechnického rázu, ktoré by mohli projekt ohroziť. Obidve skupiny navzájom intenzívne komunikujú pre dosiahnutie najlepších výsledkov.

Fungujúci softvér ma prednosť pred obsiahlou dokumentáciou

Klasické postupy stavajú dokumentáciu na základe špecifikácie požiadavkov. Snažia sa tak zachytiť funkcionalitu budovaného systému. Výsledkom vývojového procesu je systém, ktorý zákazník na začiatku procesu definoval. Manažment sa týmto postupom sa snaží minimalizovať možnosť zlyhania projektu. Klasický prístup má však dva výrazné nedostatky. Počíta s presnou predstavou zákazníka o nastávajúcom systéme, ktorú obyčajne zákazník nemá, a nereaguje na zmeny v požiadavkách. Od definovania požiadavkov po odovzdanie systému uplynie veľa času. Svet však nestojí a taktiež ani požiadavky zákazníka na systém nestagnujú. Výsledkom klasických postupov je preto často systém, ktorý presne spĺňa špecifikáciu požiadavkov ale nespĺňa aktuálne potreby zákazníka.

Agilné metodiky sa snažia tejto nepružnosti vyhnúť. Nemajú jasne stanovené požiadavky na systém na začiatku vývoja, preto nie je možné zostaviť klasickú dokumentáciu. Šetrí sa tým čas potrebný na vývoj. Za kľúčovú časť dokumentácie je považovaný vlasý kód. Agilne metodiky priniesli koncept jednoduchosti ? neprodukovať viac ako je treba a nesnažiť sa tvoriť dokumenty, ktoré zachytávajú budúcnosť. Šetrí sa tým veľa úsilia, ktoré by bolo potrebné na vyhľadávanie v rozsiahlych dokumentáciách a udržovanie týchto dokumentov v aktuálnom stave.

Za hlavný návrh systému je považovaný budovaný zdrojový kód. Tento prístup umožňuje rýchly presun do fázy programovania, ktorá priamo zachytáva požiadavky zákazníka. Preskakuje sa tým modelovanie rôznych abstraktných modelov, čím sa šetrí čas. Vzniká tu však riziko, že zákazník musí byť pripravený na pracovania so systémom počas jeho vyvíja.

Spolupráca so zákazníkom je uprednostnená pred vyjednávaním zmluv

Agilné metódy sú založené na flexibilite a ich sila spočíva v možnosti rýchleho sa prispôsobenia meniacim sa požiadavkam. Pri vývoji sa intenzívne komunikuje so zákazníkom a funkcionalita sa vytvára a upravuje podľa jeho predstáv. Zákazník sa stáva členom vývojového, spolupracuje na odsúhlasovaní rozhodnutí a ovplyvňuje vývoj.

Klasické metódy zjednávania kontraktu sa opierajú o špecifikáciu budúceho systému. Podľa nej sa vypočíta fixná cena projektu. Pre agilné metódy tento postup nie je aplikovateľný, lebo konečná funkcionalita budúceho systému sa začne rysovať až počas samotného vývoja. Agilné metódy musia byť preto podporené novým druhom kontraktu, ktorý nie je založený na fixnej cene dodávky.

Vniká tu potreba pre nový obchodný vzťah, ktorý je založený na úzkej spolupráci zákazníka s dodávateľom. Zákazník má veľkú moc v ovplyvňovaní celého vývoja a nesie aj veľkú časť zodpovednosti za úspech projektu.

Reagovanie na zmenu má prednosť pred nasledovaním plánu

V súčasnom rýchlo sa meniacom svete je ťažké používať prediktívne metodiky alebo definovať stabilnú špecifikáciu požiadavkov. Klasické postupy sa pomocou plánovania snažia zamedziť zlyhaniu projektu a tým šetriť náklady na projekt. Ich výsledok však nemusí splniť momentálne očakávania zákazníka.

Agilne metódy volia iný prístup. Zmenám sa nedá vyhnúť a preto sa proti ním nesnažia bojovať. Vzniká tu snaha o minimalizáciu nákladov na prevádzanie potrebných zmien počas vývoja. Vývoj striktne nenasleduje stanovený plán. Na začiatku je identifikovaná počiatočná funkcionalita, ktorá je počas vývoja prispôsobovaná. Jednotlivým funkčným častiam je pridelená priorita, podľa ktorej sa pri vývoji postupuje. Zákazník ma možnosť ovplyvňovať prioritu jednotlivých funkcii. Vyžiadať doplnenie novej funkcionality systému, modifikovanie už existujúcich častí a vymazanie nevhodnej funkcionality

3.4 Business Driven Development

Business driven development (BDD) je moderný robustný prístup k tvoreniu softvéru. Podobne ako Agilné metodiky sa snaží reagovať na časte a rýchle zmeny pri vývoji softvéru v dnešnom dynamicky sa meniacom svete. Dôraz je kladení na vývin softvéru, ktorý zodpovedá trhovým trendom a firemným potrebám. Do popredia sa vyzdvihuje súlad IT riešení a firemných požiadaviek.

Klasické softvérové riešenia boli tvorené na základe stanovených požiadavkov. Systémy vzniknuté týmto spôsobom boli do veľkej miery neflexibilné. Ich rozširovanie bolo veľmi obtiažne a nákladne. Znovupoužitie funkcionalit taktiež nebolo veľké. V súčasnom meniacom sa svete nemá takýto prístup miesto. Firmy musia byť každodenne pripravené na meniace sa prostredie, aby si zachovali konkurencieschopnosť. Informačné systémy musia preto držať krok s týmito zmenami a podporovať firemné procesy. Kľúčovým sa stáva prepojenie produktov IT a potrieb firmy.

Pre dosiahnutie potrebného prepojenia firemných potrieb a IT sa môžeme odraziť od modelovania firemných procesov a rôznych metrík, ako napríklad návratovosť investícii (return of investments ROI), kľúčových indikátorov výkonu (key performance indicaotrs

KPI) a dalších.

Ako hlavný prvok premostenia firemných potrieb a IT je vhodne využiť modelov firemných procesov (Business proces models BPMs). Je preto dôležité aby IT odborníci boli schopný tieto diagramy čítať a pochopiť.

Analyzovaním procesov môžeme dospieť k tomu, že požadovaná funkcionalita už existuje a môže byť znovu použitá a sú prípadne do programovane len malé časti na prepojenie už existujúcich modulov. V prípade že sa nedá znovu použiť žiadna časť systému, budeme vyvíjať od začiatku.

3.4.1 BDD model činností

Vzniká snaha na uzavretie priepasti medzi firemnými potrebami a IT riešeniami. Toto prepojenie však musí zostať pružné a prístupné pri vytváraní IT riešení. Tieto trendy viedli k zavedeniu servisne-orientovanej architektúry (Service-Oriented Architecture SOA).

SOA poskytuje rámec (framework) spolu s princípmi a smernicami na vytváranie znovu použiteľných, zlučiteľných a nastaviteľných servisov (services), ktoré sú platformovo nezávislé. Využitie SOA vyžaduje BDD prístup, ktorý spracúva firemné ciele a požiadavky z hora na dol do návrhov, vývoja a testovania. Získavame tým prostredie, v ktorom sa aplikácie tvoria zo znovu použiteľných servisov alebo z nových servisov, ktoré sú vyvíjané podla aktuálnych firemných potrieb. Tým sa zabezpečuje potrebná flexibilita v IT.

Nasledujúci obrázok 3.3 znázorňuje vysoko-úrovňový náhľad na typickú postupnosť činností v BDD metodike.

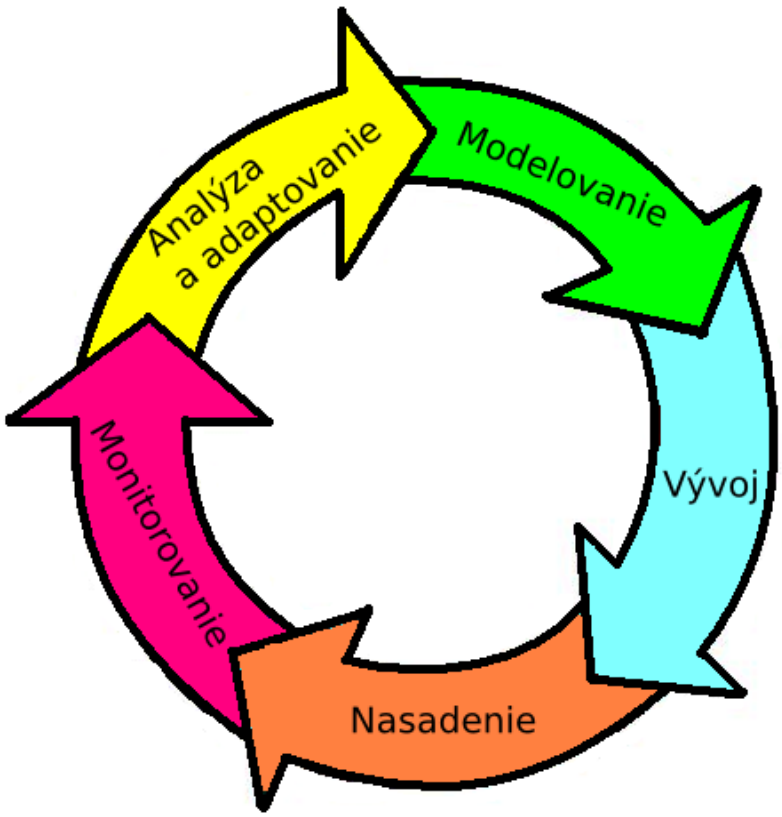
Prvým korkom vo vývoji nového IT riešenia je navrhnutie firemného procesu. Odporúča sa začať modelovaním kľúčových firemných procesov. Výsledkom modelovania je predanie firemných požiadaviek na systém IT riešiteľskému tímu. Navrhnutým firemným procesom musí byť zo strany firemných návrhárov určená dôležitosť, ktorá je vypočítaná pomocou rôznych metrík ako napríklad ROI a KPI.

Po ukončení modelovania firemných procesov nastupuje fáza zbierania požiadavkov na systém. Modely vzniknuté v predchádzajúcej fáze sú hlavným vstupom. Identifikujú sa v nich prípady užitia (use case), od ktorých sa odvíja návrh servisu. Stále sa pritom berie ohľad na firemné procesy, ktoré zachytávajú vzájomné prepojenie jednotlivých prípadov užitia. Po tom ako je servis implementovaný, pokračuje sa ďalšou fázou a to nasadením. Servis je nasadený na aplikačný server, kde je verejne prístupný a je možné ho vyhľadať.

Následne za uvedením servisu do prevádzky začína fáza monitorovania. Zbierajú sa v nej informácie o behu servisu v skutočnom čase, spracovávaných dátach a hláseniach. Sledovanie je doplnenie o širokú paletu meraní podla vopred definovaných metrík a výkonnostných parametrov. Fáza monitorovania je podstatná pre určenie funkčnosti a správnosti vytvoreného servisu. Správne vytvorený servis musí spĺňať všetky požadované parametre.

Na záver nastupuje analyzovanie nazbieraných dát z predchádzajúcej fázy. Dáta sú predané na analýzu architektom, návrhárom a vývojárom. Všetci zúčastnení pri vývoji vyhodnotia zozbierané údaje a na ich základe navrhnú vylepšenia systému. Niekedy sa taktiež zmeny môžu prejaviť aj na firemnej úrovni zmenou firemných pravidiel a externých rozhraní. Zmenami sa proces opäť posúva do fázy modelovania, čím sa postup činností uzatvára a začína nové kolo vývoja. Ustavičným vylepšovaním existujúceho systému umožňuje tento mechanizmus rýchle prispôsobenie sa meniacim sa podmienkam.

V nasledujúcej časti sa budeme podrobnejšie venovať vývojovému životnému cyklu.



Obrázok 3.3: BDD model činností.

3.4.2 Analýza firemných požiadavkov

Prvým a veľmi podstatným krokom pri vývoji softvéru je pochopenie firemných požiadaviek. Získavame ich pomocou komunikácie so zainteresovanými osobami a pozorovaním už existujúceho systému. Je veľmi dôležité komunikovať so všetkými vrstvami pracovníkov vo firme aj s najzaneprázdnenými manažérmi a vedúcimi. Výsledky treba zachytiť a zdokumentovať. Dokumenty by mali obsahovať minimálne:

- Firemnú víziu.
- Firemné ciele (krátkodobé a dlhodobé), ktorými uskutočňujeme víziu spoločnosti.
- Firemné požiadavky vysokej úrovne, ktoré napomáhajú dosiahnuť ciele.
- Problémy s existujúcim firemnými procesmi.

Taktiež je veľmi dôležité pochopiť prostredie a organizačnú štruktúru firmy. Rozčleniť firmu na funkčné celky, ktorým je jasne pridelená firemná funkcionalita vyššej úrovne. Tieto poznatky treba zachytiť a sformalizovať do firemnej doménovej matice (business domain matrix).

3.4.3 Modelovanie firemných procesov

Modelovanie firemných procesov (Business process modeling ? BPM) je technika na vizuálne zachytenie firemných procesov pomocou postupnosti aktivít a rozhodovacích uzlov. Účelom BPM je vytvoriť modely, ktoré inžinierske skupiny môžu použiť na implementovanie servisov. Stanovuje sa ideálny firemný proces, ktorý sa snažíme dosiahnuť.

Každá organizácia alebo jej pod časť má vymedzenú firemnú funkcionalitu, ktorú podporuje alebo poskytuje. Pomocou BPM modelujeme tieto funkcionality. Každá úloha je pridelená roli ktorá odpovedá nejakej entite alebo skupine entít. Rola môže byť pridelená viacerým entitám a jedna entita môže vystupovať vo viacerých roliach.

Firemný proces po analýze môže byť reprezentovaný postupnosťou aktivít a úloh. Úloha je najmenšia celistvá jednotka funkcionality, ktorá ma pre užívateľa význam. Zložitejšie procesy sa podrobnejšie modelujú v podprocesoch, z ktorých sa skladá hlavný proces. Príklady procesov sú uvedené na obrázkoch 2.4 a 2.5.

Obyčajne celistvý úsek procesu odpovedá istej funkčnej jednotke firmy, niekedy je však zapotreby priradiť funkčnú jednotku organizácie procesu. Získavame tým hlboké pochopenie rozloženia zodpovedností a úloh vo firme, ktoré neskor zužitkujeme pri návrhu a vývoji softvérového projektu.

Navrhnuté modely sa dajú testovať prípadne sú do nich zavedené monitorovacie parametre aby bolo uľahčené meranie výkonu a funkčnosti procesov. Vykonávame nad nimi simulácie, skrz ktoré získavame informácie pre vylepšenie modelov.

Modelovaním BPM získavame sadu modelov, ktoré slúžia ako hlavný vstup do nasledujúcej fázy. Plne popisujú všetky funkčné časti firmy a procesy v nich prebiehajúce. Poskytujú tým náhľad na vzťahy medzi jednotlivými sekciami firmy. Na základe týchto znalostí sa ľahko vytvorí dátový model budúceho systému.

3.4.4 Modelovanie prípadov užitia

Pri modelovaní prípadov užitia využívame firemne procesy, pomocou ktorých vytvárame jednotlivé prípady užitia. Role z firemných procesov sú prevedené na aktorov a prípady užitia získame prevedeným jednotlivých firemných procesov prípadne podprocesov. Vlastný prevod modelov poskytuje značnú mieru flexibility.

Pre prevod nie je jasne definovaný formálny postup, ale je možné naznačiť použiteľný mechanizmus pre prevádzanie BPMs na modely prípadov užitia.

Nie je vhodné mapovať jednu aktivitu či úlohu v procese na jedne prípad užitia. Prípady užitia by pri takomto postupe znázorňovali len jednu interakciu pretože úloha odpovedá jednej interakcii. Podla definície jazyka UML je ale prípad užitia úplná postupnosť interakcii. Znamená to, že jedne prípad užitia by mal zahŕňať všetky interakcie, ktoré privedú systém do takého stavu, aby mohla byť vykonávaná operácia znovu spustená. Vyplýva z toho, že zachytávanie jednotlivých interakcii v prípadoch užitia ne je vhodne.

Podobne mapovanie celého firemného procesu na prípad užitia by bolo značne zložité a neprehľadné. Vyskytovalo by sa tu veľké množstvo rolí a vznikal by veľký počet alternatívnych ciest.

Jedným spôsobom ako sa k tomuto problému môžeme postaviť, je zavedenie krokov. Jeden krok si definujeme ako postupnosť úloh, ktoré môžu byť vykonané bez prerušenia tou istou rolou. Napríklad uloženie ?ulož rezerváciu? môže byť prípad užitia ale ?ulož re-

zerváciu a zašli dovolenkový leták? nemôže byť prípadom užitia, lebo uplynie istý čas od momentu ako je rezervácia uložená a momentu, keď je možné vytvoriť, prispôsobiť a zaslať dovolenkový leták užívateľovi. Na firemné procesy sa ale môžeme pozerať aj ako na postupnosť krokov, pričom každý krok je prípad užitia s požadovaným výstupom a správaním.

Ako náhle sem schopný identifikovať prípad užitia, popíšeme jeho hlavné kroky skrz aktivity, ktoré v ňom prebiehajú. Takýmto spôsobom získavame popis prípadu užitia pomocou postupnosti krokov.

Všetky identifikované prípady užitia majú význam, lebo vznikli identifikovaním krokov vo firemnom procese. Každý prípad užitia odpovedá aspoň jednému firemnému procesu, či už celému alebo jeho pod časti, a je ich možné spojiť aspoň s jedným firemným procesom. Sú identifikované všetky prípady užitia, lebo každému kroku vo firemných procesoch je pridelený odpovedajúci prípad užitia s popisom hlavných krokov.

Ak sa nám úspešne podarí vykonať všetky tieto prevody, zvyšujeme tým pravdepodobnosť úspechu IT projektu.

3.4.5 Modelovanie servisov – SOA

Ak chce spoločnosť využívať servisne orientovanú architektúru (SOA) tak si musí vytvoriť portfólio servisu, ktoré spoločnosť poskytuje. Buď pre interné alebo externé použitie. Pri dosahovaní SOA môže spoločnosť dosiahnuť rôzne úrovne. Väčšina spoločnosti zavádza webové služby, ale sú však omnoho vyššie úrovne implementovania SOA v spoločnosti. Určenie úrovne na ktorej sa firma nachádza v rámci presadzovania SOA je zložitý proces sám o sebe. Na to aby spoločnosť dosiahla skutočne SOA je potrebné pochopiť, prijať a nasledovať metodiku.

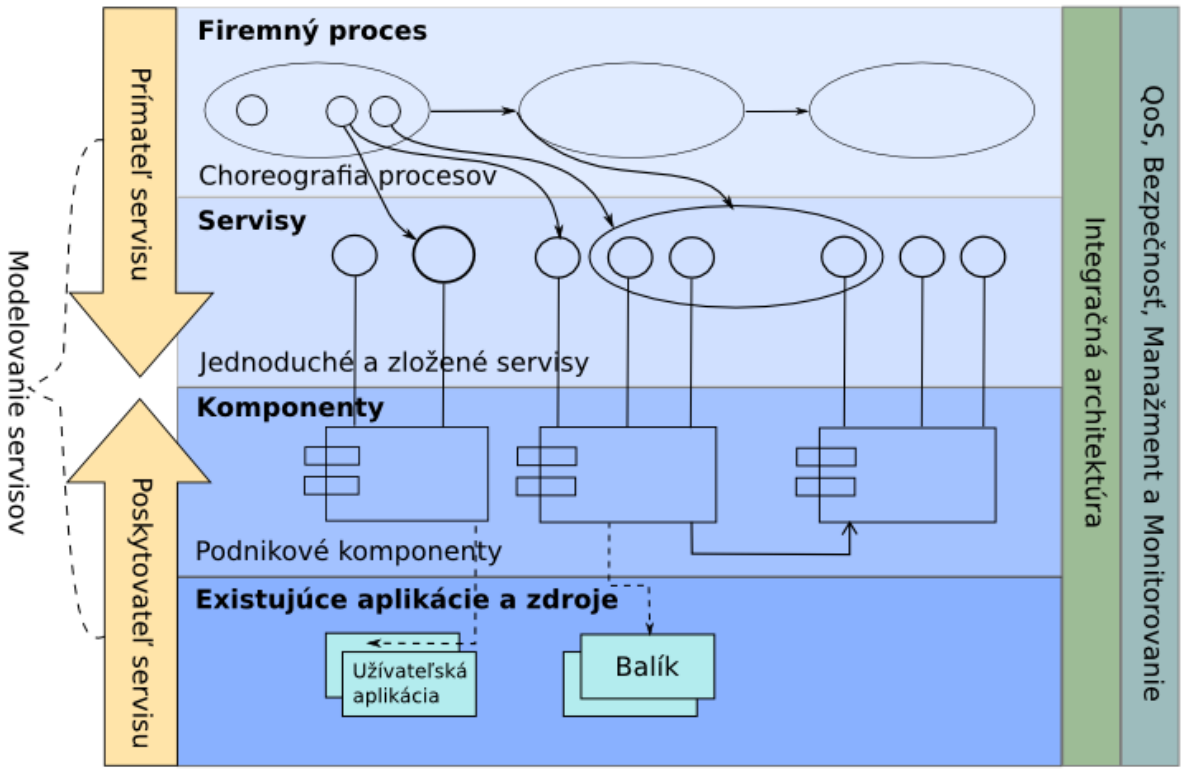
Proces vývoja informačného systému pre firmu do ktorého využijeme SOA architektúru bude pokračovať spracovaním výsledkov firemnej analýzy , na identifikovanie a navrhnutie servisov. Začneme stavať na výsledkoch minulej fázy a to na firemnej vízií a cieľoch organizácie. Na automatizáciu firemných procesov musíme dodržať prioritné kritéria. Pri ich určení úzko spolupracuje IT a firemní manažment. Musia prehodnotiť kľúčové problémy firmy ako napríklad:

- Určenie primárnych procesných slabostí a úzkych miest
- Problematické body vytknuté zákazníkom
- Procesy, ktoré musia byť škálovateľné z nárastom objemu transakcií

Na základe takýchto analýz môže byť stanovený plán v akom poradí budú procesy vyvíjané. Procesy sú vlastnené funkčnými časťami firmy. Pár týchto jednotiek môže byť na začiatku vybraných aby sa na ne IT sústredilo a získal sa tým krátkodobý okamžitý prospech. Akýkoľvek servis, ktorý môže byť identifikovateľný musí podporovať ciele firmy, ktoré sa snaží z časti alebo úplne vyriešiť. Týmto si pomôžeme pri uprednostňovaním funkčných častí firmy pre IT vývoj. Pre veľmi dôležité zložky firmy budú následne identifikované a definované servisy. Problémy s existujúcimi firemnými procesmi môžu byť použité ako hlavné ciele zavádzania servisov. V každom prípade pri určovaní priority zamerania IT musíme

mať na pamäti firemné potreby.

Každý firemný proces ako aj prípady užitia, ktoré boli identifikované v predchádzajúcich fázach vývoja systému môžu byť pripísané na zoznam kandidátskych servisov. Však nie všetky kandidátske servisy aj realizované. Zmyslom SOA je zostaviť portfólio servisov, ktoré budú znovu použiteľné v čo najviac firemných procesoch. Implementácia týchto servisov musí byť taktiež dobré premyslená aby sa jednotlivo implementované časti dali využiť v rôznych implementáciách servisov.



Obrázok 3.4: Úrovne servisne-orientovanej architektúry

Obrázok 3.4 nám poskytuje náhľad na rôzne úrovne SOA. Zobrazuje kroky firemného procesu, ktoré sú prevedené na jeden servis alebo na skupinu viacerých servisov. Taktiež zobrazuje ako môže byť servis implementovaný komponentami. Tieto komponenty môžu byť pozostatky bývalého systému, komerčne dostupné riešenia alebo úplne z nuly vyvíjané komponenty.

Po zostavení listiny kandidátov na servisy sa snažíme dopracovať ku konečnej podobe servisného portfólia a vypracovať servisný model pre firmu. Základné činnosti, ktoré musia byť vykonané na dosiahnutie servisného modelu sú:

- Analýza z hora dole cez procesné modely.
- Analýza už existujúceho systému a aplikácií.
- Vytvorenie prvotného servisného modelu pre kandidátske servisy.

- Jasné stanovenie popisov vlastnosti a kvality servisov.
- Zviazanie každého servisu s firemným cieľom a zabezpečenie aby sa servis podielal aspoň na jednom firemnom procese.

3.4.6 Systémový návrh a vývoj

Počas vývojovej fázy tím pracuje s počiatočným prípadom užitia a servisným modelom. Zanára sa do modelu a špecifikuje funkcionalitu pre každý prípad užitia.

Systémový návrh

Na vytvorenie modelu systémových komponent využijeme popis funkcionality jednotlivých prípadov užitia. Model komponent popisuje rozhranie komponent, ktoré ponúkajú a zachytáva taktiež vzťahy medzi jednotlivými komponentami v celom systéme. Delenie systému na komponenty odpovedá navrhnutým servisom v servisnom modeli. V modeli môžu byť zahrnuté aj komponenty, ktoré priamo neimplementujú servis, ale poskytujú len podpornú funkcionalitu. Komponenty, ktoré nemajú rozhrania sa nedajú externé využívať, tak sa využívajú len na vnútornú komunikáciu. Pre dôležité prípady užitia sú vytvárané sekvenčné diagramy skrz rozhrania komponentov. Týmto je systém navrhnutý aby komponenty z modelu komponent medzi sebou dobre komunikovali skrz rozhranie.

V tejto fáze sa taktiež identifikujú a dokumentujú nefunkčné požiadavky na systém. Nevyužívajú sa len vytvorenie SLA pre servisy ale sú taktiež využité ako vstupy do operačného systémového modelu. Tento model okrem iného popisuje infraštruktúrové komponenty ako middleweare, zasielanie správ, správa súborov a ďalšie. Taktiež popisuje ako sú komponenty distribuované po sieti a ako sú nasadené na hardvéry.

Pred samotnou implementáciou modelovaných procesov musia byť najskôr definované implementácie servisov, ich popisy a spôsob vyvolania. Nie všetky kroky procesov sa však implementujú ako priame vyvolania servisov. Jedným z hlavných dôvod prečo sa komponenty vyvolávajú priamo cez ich rozhranie je zvýšenie výkonu systému odstránením nadbytočnej réžie vykonávania. Sú aj iné dôvody, prečo je vhodné zvoliť hybridný prístup pri implementácii procesov. Pri návrhu je preto dôležité uvedomiť si tieto potreby a zaznamenať ich pre ďalší vývoj.

Po ukončení makro-návrhu na modelovaním diagramu komponent a operačného modelu pristúpime k mikro-návrhu, ktorý zahŕňa modelovanie diagramov tried a sekvenčných diagramov pre každú komponentu systému. Pri návrhu postupujeme iteratívne a využívame overené návrhové vzory. Dodávame tým návrhu na robustnosti a spoľahlivosti.

Systémový vývoj

Počas vývoja sa prevádzajú modely z návrhovej časti do praktickej nahraditeľnej podoby. Vyberá sa technológia (nap. Java? 2 Platform, Enterprise Edition), programovací jazyk (nap. Java) a vývojové prostredie, v ktorom bude systém naprogramovaný. Firemné procesy sa prevádzajú do spustiteľnej formy využitím Business Process Execution Language (BPEL) a slúžia ako východiskový bod pre implementáciu procesov do programovacieho jazyka. Pre definície procesov je taktiež vybraná technológia. Vyvinuté servisy sú medzi sebou previazané pomocou nástroja na koordinovanie procesov, ktorý poskytuje možnosť previazať servisy skrz rozhrania ktoré poskytujú podľa navrhnutých firemných procesov.

Dôležitým aspektom takéhoto prístupu že jeden servis môže byť použitý vo viacerých

procesoch čo je hlavným prínosom BDD prístupu. Firemné procesy vznikajú preskladaním servisov podľa potreby. Tento prístup poskytuje veľkú flexibilitu, lebo nevyhovujúce procesy sú vytvorené z nových lepších servisov a vyvinuté servisy sa zas používajú v iných procesoch. Nové procesy nie sú tvorené úplne od nuly ale môžu byt tvorené už z existujúcich servisov. Minimalizuje sa tým čas a náklady na rozširovanie a úpravy systému, a IT môže ľahšie a rýchlejšie reagovať na potreby firmy.

Paralelne s vývojom je budovaná aj infraštruktúra pre budúci systém. Využíva sa pri tom operačný model systému. Súčasne po dokončení vývoja musí byť pripravený aj hardvér aby mohlo nastať nasadenie systému.

3.4.7 Nasadenie, monitorovanie a analýza zozbieraných dat

Po dovedení vývojovej snahy do bodu, keď je otestovaná a odobrená časť systému, nastáva jej nasadenie do skutočnej prevádzky. Tento okamih musí predchádzať zosynchronizovanie plánov projektu, aby bola pripravená infraštruktúra pre nasadzovaný systém. Nasadenie musí byť pozorne naplánované aby nenastalo preťaženie určitých častí systému užívateľmi. Je treba uvažovať o distribuovanom rozložení softvérových artefaktov a vytváraní zhlukov (clusters).

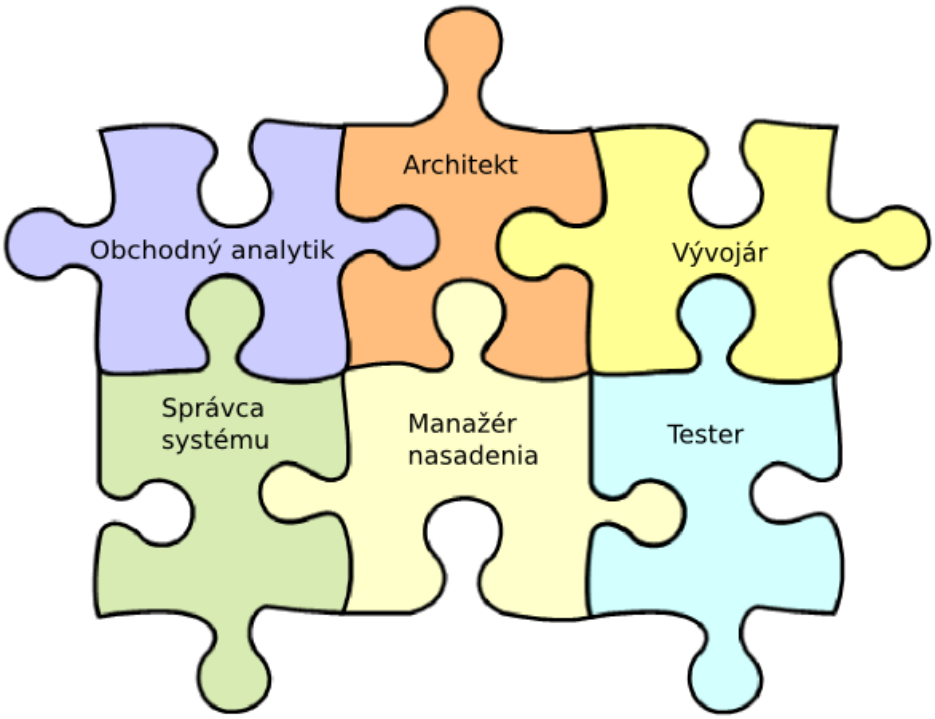
Behové prostredie pre firemné procesy slúžiť aj pre analyzovanie bežiacich procesov. Monitorovanie počas behu umožňuje merať výkon riešenia, vyhodnotiť jeho výkon a určiť či spĺňa požiadavky, ktoré boli preň stanovené. Všetky dáta zozbierané počas chodu systému sa ukladajú pre ďalšiu analýzu.

Pri vyhodnocovaní výsledkov monitorovacej fázy sa využijú simulácie firemných procesov, ktoré boli vytvorené v rámci BPM. Dáta nazbierané za behu sa porovnávajú s očakávanými výsledkami, ktoré sme získali zo simulácii. Ak sa výsledky od seba zanedbateľne líšia, vývoj systému je označený za úspešný, lebo splnil zadané požiadavky. Naopak ak sa výsledky od seba značne líšia, musia byť prevedené ďalšie činnosti. Dáta zozbierané počas behu systému sa analyzujú a identifikuje sa krok, v ktorom vzniká najväčší rozdiel medzi požadovanými a dosiahnutými výsledkami.

Ako prvé sa prevedie podrobná analýza implementovaného zdrojového kódu. Je tu snaha odhaliť miesta, ktorých prepísaním zvýšime výkon systému. Ak po ukončení úpravy kódu nezískame žiadne priblíženie sa k požadovanému výkonu pristúpime k opatrnej zmene firemného procesu. Prevedená zmena na procese musí byť následne analyzovaná a musí byť určený jej vplyv na spoločnosť. Ak by tento vplyv bol zásadný a nevyhnutný pre spoločnosť, musí byť zmenený krok procesu a identifikovaný nedostatok je odstránený. V úvahu taktiež pripadá analýza infraštruktúry, na ktorej je systém nasadený. Jej prestavba môže zvýšiť výkon ale taktiež môže byť ľahko prekročený rozpočet preto je toto jedna z posledných možností.

3.4.8 Definované role

Pre úspešné zvládnutie projektu je potrebné zapojiť ľudí s rôznymi schopnosťami. BDD definuje role, ktoré napomáhajú k úspešnému prevedeniu IT projektu poháňaného firemnými cieľmi, víziami a potrebami. Obrázok 3.5 znázorňuje šesť najdôležitejších rolí, ktoré sú vyžadované na vykonanie BDD vývojového cyklu.



Obrázok 3.5: Hlavné role využívané v BDD.

Popis jednotlivých kľúčových rolí:

Obchodný analytik – je rola vysokej úrovne, ktorá ma na starosti obchodné analýzy a BPM. Vykonáva identifikáciu prípadov užitia a vytvára špecifikácie pre každý prípad užitia. Analytik v tejto role sa môže podieľať aj na špecializovanejších úlohách v neskorších fázach projektu.

Architekt – je rola vysokej úrovne zodpovedná za architektúru a návrh systému. Táto roľa zahrnuje špecializované pod-role ako napríklad aplikačný architekt, SOA architekt, vedúci návrhár a ďalší. Tieto role sú zodpovedné za rôznu architektonickú činnosť, ktorá súvisí z návrhom projektu.

Vývojár – je rola vysokej úrovne, ktorá má na starosti implementáciu navrhnutého riešenia. Opäť môže byť špecializovaná na pod-role určené na čiastočné úlohy ako napríklad databázoví programátor, vývojár, vývojár v jazyku java, vývojár webu a ďalší. Vývojári pracujú na rôznych úrovniach aplikačnej vrstvy podľa ich zamerania.

Tester – je rola zodpovedná za aktivity spojené s testovaním aplikácie pred jej nasadením do reálneho prevádzky. Tester vytvára testovacie skripty priamo podľa funkčných požiadavkov, ktoré vychádzajú z prípadov užitia. Tieto testovacie skripty sú následne spúšťané s rôznymi vstupnými dátami a je vyhodnocovaná správnosť vracaných hodnôt. Čím podrobnejšie sú testovacie prípady a ich vykonávanie, tím je robustnej-

šia aplikácia a minimalizuje sa v nej výskyt chýb.

Manažér nasadenia – je zodpovedný za nasadenie aplikácií na infraštruktúru v rôznych prostrediach. Taktiež je zodpovedný za činnosť spojenú z nasadzovaním aplikácií do cieľového prostredia. Napríklad vyvíjanie inštalačných skriptov, správne na-konfigurovanie aplikácie atď.

Správca systému – je zodpovedný za fungovanie aplikácie a jej správu počas jej behu a využívania. Táto rola taktiež môže byť zodpovedná za zbieranie dát počas chodu aplikácie, ich analyzovania a porovnania výsledkov voči požiadavkam na systém.

Kapitola 4

Návrh vývojovej metódy

Cieľom tejto kapitoly jej navrhnúť metódu pre vývoj malých softvérových projektov. Využijeme pri tom zozbierané znalosti o moderných prístupoch k navrhovaniu systémov. Metóda bude zameraná na maximálnu efektivitu. Vývoj postupujúci podľa navrhnutej metódy musí zvládnuť tím minimálnej veľkosti. Metóda bude založená na osvedčených a už dlhé roky používaných metodikách, ktoré budú tvoriť jej základ a bude vychádzať z ich pozitívnych vlastností.

Hlavnou snahou navrhovanej metódy je minimalizácia prostriedkov a úsilia, ktoré priamo nevedie k tvoreniu systému. Metóda bude slúžiť na vývoj malých projektov ktoré sú ľahko uchopiteľné pre odborníka a preto rozsiahle modelovanie problému nie je efektívne. Bude tu snaha o potlačenie nepotrebnej byrokracie a nadbytočného produkovania modelov a dokumentácii. Na druhej strane nesmie byť však ohrozená spoľahlivosť metódy, ktorá by sa prejavila zlyhaním projektu.

4.1 Charakteristika metódy

Pri modelovaní novej metódy musíme ako prvý krok zostaviť jej charakteristiku. Popíšeme tým základné vlastnosti, ktoré budú metódu charakterizovať a určovať akým spôsobom bude fungovať. Inšpiráciu pre stanovenie základných čŕt môžeme čerpať z historicky starších metodík. Môžeme sa opierať o výhody a nevýhody získané ich používaním. Na základe týchto znalosti vyberieme také vlastnosti, ktoré budú pre charakter nášho projektu najvhodnejšie.

4.1.1 Model životného cyklu

Ako prvé sa zamyslíme nad najvhodnejším modelom pre našu metódu. Medzi historicky najstaršie modely patrí model vodopád. Tomuto modelu sme sa venovali v tretej kapitole 3.1. Je založený na lineárnom postupe skrz fázy vývoja. Jeho prínosom je jasné definovanie jeho vývoja a jeho postupností. V súčastnosti je tento postup prekonaný, lebo len ťažko sa s ním zvládajú zložitejšie projekty. Hlavnou nevýhodou okrem neflexibility a zložitej opravy chýb je dlhý časový úsek od zadania projektu do doby kedy sa zákazník môže stretnúť s objednanou aplikáciou. Moderný návrh však nezabudol na vodopád, ale využíva jeho vylepšenia.

Jedným z najrozšírenejších modelov, ktoré vychádzajú z vodopádu a používajú sa v moderných návrhoch je iteratívny model. Rozdeľuje problém na menšie podproblémy. Uľahčuje nám tým zvládnutie aj zložitejších projektov. Koncový produkt získame po vykonaní všetkých iterácií. Často sa tento prístup kombinuje s inkrementálnym vývojom. Podrobnejšie boli oba modely popísané v tretej kapitole 3.2.

dasdasssssssssssssssssssssssssssssssssssssssssssssssssssssss???????? ??????????????????????, ?????????????????????????????? ?????????????????? ??????????????????

4.1.2 Agilné prvky

4.1.3 Využitie BPMN

4.2 Životný cyklus

4.3 Využité role

4.4 Vývoj softvérového projektu

4.4.1 Špecifikácia požiadavkov

4.4.2 Návrh firemných procesov

Identifikácia procesov Určenie hlavného procesu Dekompozicia procesu Určenie komponent

4.4.3 Naplánovanie iterácii a inkrementov

4.4.4 Popis prípadov užitia

Previazanie BPMN a prípadov užitia

4.4.5 Diagram tired

Kapitola 5

Prípadová štúdia

5.1 Špecifikácia požiadavkov

Účelom systému je zabezpečiť virtuálnu konferenciu. Webová aplikácia musí mať prvky redakčného systému, ktoré umožnia administrátorovi upravovať, pridávať a odoberať jej obsah.

Účelom aplikácie bude zbieranie a sprostredkovávanie článkov vo forme virtuálneho časopisu. Aplikácia bude zobrazovať na webové rozhranie prehľad základných informácií o každom článku a to:

- autora
- inštitúciu
- názov
- kľúčové slová
- anotáciu
- odkaz na plný text v PDF formáte

Aplikácia bude obsahovať vyhľadávanie v informáciách uložených v databáze. Vyhľadávanie bude možné podľa základných informácii okrem anotácie taktiež sa nebude vyhľadávať v samotnom texte článkov.

Ku každému článku bude diskusia kde budu môcť užívatelia vyjadriť svoj názor k článku a budú mať možnosť reagovať na komentáre iných užívateľov. Diskusia bude rozvrstvená podla logickej náväznosti komentárov.

Aplikácia bude rozlišovať užívateľov, ktorí k nej budu pristupovať. Každý užívateľ, ktorý sa neprihlási bude mať právomoci neregistrovaného užívateľa. Na registráciu bude k dispozícii formulár na vytváranie novích registorvaních užívateľov.

Administrátorský účet bude pridelený užívateľovi pri inicializačnom spustení webovej aplikácie. Prípadné ďalšie administrátorské účty musí vytvárať už existujúci administrátor.

Systém rozoznáva nasledovne užívateľské skupiny:

Administrátor - Administrátor bude mať možnosť manipulovať s obsahom stránok, spravovať užívateľov a má na starosti prvotnú konfiguráciu. V správe užívateľov bude

potvrdzovať nové žiadosti o registráciu, bude môcť pozmeňovať údaje o užívateľovi, meniť ich role a blokovať účty.

Neregistrovaný užívateľ - Bude mať možnosť prehliadať zoznam uložených článkov, môže v nich vyhľadávať ale nemá prístup k plným textom článkov.

Registrovaný užívateľ - Bude mať všetky práva neregistrovaného užívateľa a naviac aj prístup k plným textom článkov z ročníka, pre ktorý zaplatil členský poplatok. Taktiež bude môcť za základný členský poplatok vložiť jeden článok. Bude si môcť upravovať informácie v profile a meniť pristupové heslo.

Redakčná rada - Bude mať možnosť stopnúť uverejnenie príspevku s odôvodnením jeho pozastavenia.

Recenzenti - Budú mať prístup k textom článkov v upraviteľnej podobe a k odovzdaným článkom majú možnosť pridávať recenzovanú verziu.

Aplikácia bude obsahovať radu notifikácii. Pri založení nového účtu bude zaslaná informácia administrátorovi so žiadosťou o jeho potvrdenie po overení zaplatenia členského poplatku.

Pri každom vložení nového článku registrovaným užívateľom budú vybraný a oboznámený o tomto článku recenzenti, ktorých úlohou bude článok recenzovať.

Registrovaný užívatelia si môžu povoliť oznámenie o nových zverejnených článkoch cez mail a pre všetkých užívateľov bude k dispozícii RSS zdroj.

Po dovŕšení limitu na zostavenie čísla časopisu bude o tom informovaná redakčná rada a administrátor.

Pravidelne bude spúšťané zálohovanie databázy, ako aj uložených plných textov článkov, ktoré budú slúžiť na obnovenie dát v prípade poruchy. Kompletné zálohy budú ukladané na predurčené uložisko.

Redakčné prvky systému umožnia administrátorovi upravovať a dopĺňať webové rozhranie systému. Bude mať možnosť meniť logo portálu, texty na stránkach, pridávať a zneplatňovať nové stránky.

Všetky zmeny rozloženia stránok sa budú prejavovať v štruktúre menu stránky, ktoré bude najviac 2 úrovňové. Stránky budu môcť byť doplňované do ktorejkoľvek úrovne menu.

Webové rozhranie bude umožňovať zmenu vzhľadu pomocou dodávaných tém. Aplikácia bude vyhotovená s jednou štandardnou témou a s témou pre postihnutých.

Téma sa bude pre registrovaných užívateľov ukladať do ich profilu.

Nevyhnutná konfigurácia hotovej distribúcie prebehne pri jeho prvom spustení. Vytvori sa tu administrátorské konto a všetky nevyhnutné nastavenia aplikácie.

Celý nasledujúci beh systému bude automaticky a všetky prípadne zmeny nastavenia a obsahu sa budu diať cez webové rozhranie administrátora.

Webové rozhranie bude prehľadne a funkcionálne, zamerane na rýchle dosiahnutie požadovaných informácií. Každá stránka musí obsahovať tieto prvky:

- logo a základne údaje o organizácii zriaďujúcej virtuálnu konferenciu
- menu stránok webového rozhrania
- ak je užívateľ neprihlásený – možnosť prihlásiť sa do systému
- ak je užívateľ prihlásený – meno užívateľa a voľbu na odhlásenie sa

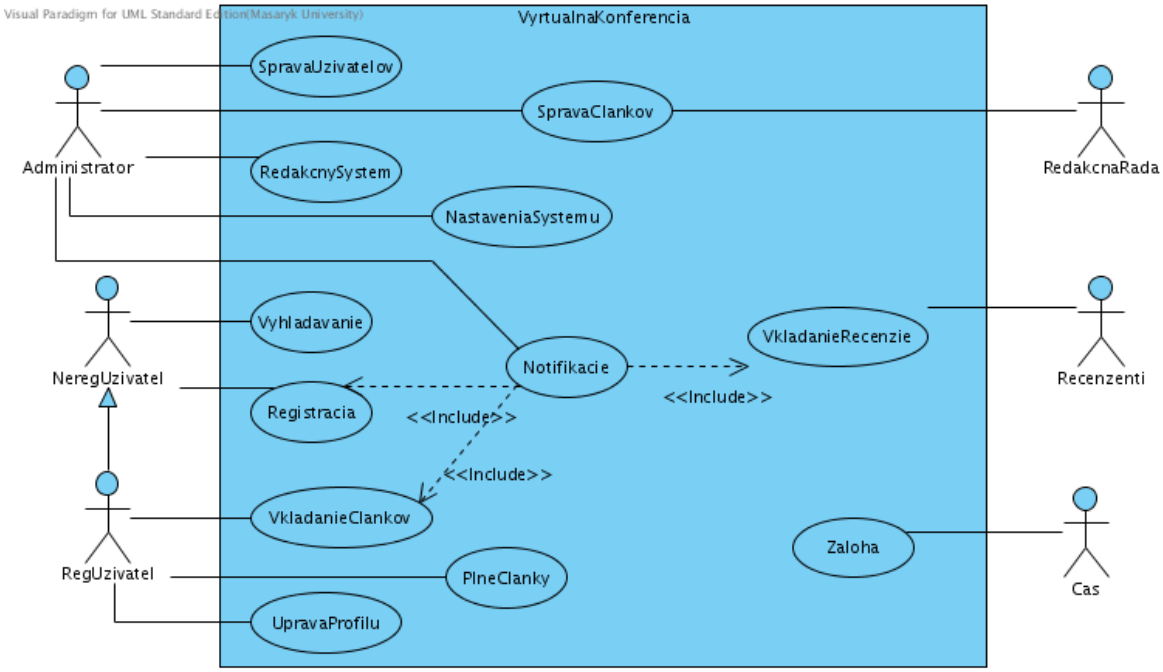
Informácie o uložených článkoch sa budu zobrazovať do prehľadného výpisu obsahujúceho základné informácie. Záznamy sa budú zobrazovať pre aktuálny rok. Staršie ročníky budú uložené v archíve.

Záznamy pre aktuálny ročník budu chronologicky usporiadané od najnovších po najstaršie. Archív bude usporiadaný podľa rokov a bude rovnako zoradený ako aktuálny ročník. Parameter zoraďovanie bude môcť užívateľ pozmeniť na meno, inštitúciu, názov a dátum. Taktiež bude možnosť zmeniť vzostupnosť alebo zostupnosť usporiadania.

V prípade, že záznamov bude viac ako limit zobrazenia na jednu stránku, zoznam sa stane viac stranovým. Užívateľ si bude môcť zvoliť koľko záznamov chce na jedne krát zobraziť.

Rozhranie vyhľadávania bude čo najjednoduchšie. Bude poskytovať voľby na určenie kategórie, v ktorej sa bude vyhľadávať:

- meno, inštitúcia, názov
- kľúčové slová
- ročník



Use case diagram virtuálnej konferencie.

5.2 Návrh firemných procesov

5.2.1 Identifikácia procesov

5.2.2 Určenie hlavného procesu

5.2.3 Dekompozicia procesu

5.2.4 Určenie komponent

5.3 Naplánovanie iterácii a inkrementov

5.4 Popis prípadov užitia

5.4.1 Previazanie BPMN a prípadov užitia

5.5 Diagram tired

Kapitola 6

Záver

zaver

Literatúra

[1] White, Stephen A. *Introduction to BPMN* [online].[cit. 2009-3-2]. Dostupné na internete: <http://www.omg.org/spec/BPMN/1.2/PDF>

[2] Object Management Group. *Business Process Modeling Notation (BPMN)* [online]. Jan 2009 [cit. 2009-3-2]. Dostupné na internete: <http://www.bpmn.org/Documents/Introduction%20to%20BPMN.pdf>

[3] Keith, Everette R. *Agile Software Development Processes: A Different Approach to Software Design* [online]. 1. Dec 2002 [cit. 2009-4-16]. Dostupné na internete: <http://www.agilealliance.com/system/article/file/1099/file.pdf>

[4] Mitra, T. *Business-driven development* [online]. Dec 2005 [cit. 2009-3-2]. Dostupné na internete: <http://www.ibm.com/developerworks/webservices/library/ws-bdd/>

Dodatok A

Príloha A