MASARYKOVA UNIVERZITA
FAKULTA INFORMATIKY

Použitie BPMN pre malé 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: RNDr. Tomáš Ludík

Poďakovanie

Na tomto mieste by som sa chcel rád poďakovať vedúcemu mojej diplomovej práce pánovi RNDr. Tomášovi Ludíkovi, za jeho podporu a smerovanie pri písaní tejto práce.

Zhrnutie

TBD

Kľúčové slová

Business Process Modeling Notation, Unified Modeling Language, Vodopád, Iteratívny / inkrementálny vývoj, Agilné metódy vývoja, Business Driven Development, Servisne Orientovaná Architektúru

Obsah

Kapitola 1

Úvod

Pri riešení 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 navrhnúť čo najlepšie riešenie. Žiadna metodika nám nezaistí splniteľnosť projektu, ale jej využitie minimalizuje riziko krachu projektu.

Najfrekventovanejšie modelovacie metodiky súčasnej doby sú veľmi rozsiahle a silné nástroje. Definujú veľké množstvo rolí a zavádzajú komplexné procesy, čím dokážu zvládať veľké projekty. Vnášajú tým do vývoja veľký prínos, prostredníctvom ktorého je projekt lepšie zvládnuteľný a zároveň sa i 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 metodík pri akomkoľvek vývoji. Napriek tomu sa naskytuje priestor pre hľadanie nových ciest ich riešení. Namiesto využívania rozsiahlych používaných a overených metodík sa treba zamerať na ich esenciálne časti. Na základe týchto častí sa vybuduje ľahko zvládnuteľná a flexibilná metóda.

Malé softvérové projekty sú väčšinou spracúvané neveľkým počtom pracovníkov tak zo strany vývojára, ako aj zo strany klienta. Objavuje 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 daného systému. Podrobne sa tu popisuje rozšírený, i keď možno nie príliš známy, 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. Kontrastujú s nimi tradičné štruktúrované metodiky ako Unified Process (UP), ktoré stavajú na definovaných postupoch a roliach. Podrobnejšie sa venujeme najmä tým, z ktorých čerpáme inšpiráciu na zostavenie vlastnej metódy.

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

vyberajú relevantné časti. Spája sa v nej agilný prístup a tradičné štruktúrované metodiky. Metóda čerpá inšpiráciu z Business Driven Development (BDD) z dielne IBM. Na zachytenie požiadaviek a identifikáciu modulov v projektovanom systém používa hierarchiu BPMN diagramov. Jednotlivé moduly sú následne modelované pomocou tradičných UML diagramov.

Záverečná piata kapitola overuje vhodnosť definovanej metódy. Na základe jej využitia je vytvorená prípadová štúdia, popisujúca správu vedeckého časopisu. Pomocou uvedených nástrojov modeluje hierarchiu procesov prebiehajúcich pri fungovaní správy vedeckého časopisu. Na vzniknutej procesnej mape sú identifikované procesy, ktoré je možné automatizovať. Následne sú určené komponenty, ktoré sú pomocou UML modelované, a v závere je načrtnutá implementácia.

Kapitola 2

Modelovacie nástroje

2.1 Business Process Modeling Notation

V roku 2004 vydal Business Process Management Initiative (BPMI) štandard BPMN 1.0. Cieľom tohto 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 dôraz na to, aby bolo možné pomocou neho zachytiť aj komplexné firemné procesy. Pre lepšie zvládnutie týchto protichodných požiadaviek bolo navrhnuté malé množstvo kategórií, ktoré napomáhajú ľahšej orientácii v základných typoch. V rámci každej zo základných kategórií je možné modifikovať definované elementy prostredníctvom rozširujúcich informácií. Rozšírenia však nesmú narúšať základné charakteristiky elementov, čím by znižovali ich zrozumiteľnosť.
Základné kategórie elementov sú:

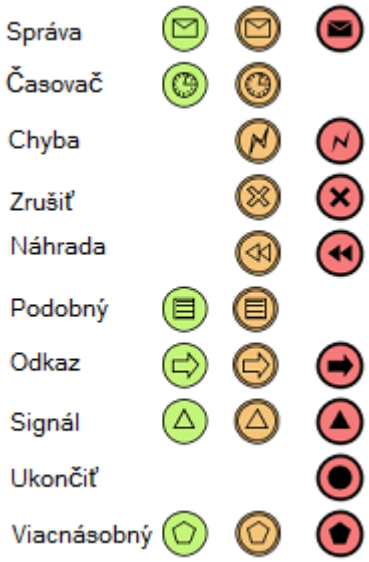
- Tokové objekty (Flow Objects)
- Spojovacie objekty (Connecting Objects)
- Plavecké dráhy (Swimlanes)
- Artefakty (Artifacts)

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 udalosti (Start), v priebehu používame medziľahlé (Intermediate) a tok ukončujeme koncovými (End) udalosťami.
Pre každú udalosť môžeme do krúžku umiestniť symbol spresňujúceho spúšťača alebo výsledku. V BPMN je definovaných desať podtypov udalostí. Nie všetky podtypy sa však môžu používať s každým typom udalostí. Na obrázku [Obr. 2.1] sú znázornené všetky podtypy.
Medziľahlými udalosťami môžeme nadviazať na aktivity, čím určíme ich alternatívne ukončenia. Napríklad vypršanie času na 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. Typmi aktivít sú: proces (Process), podproces (Sub-Process), 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.

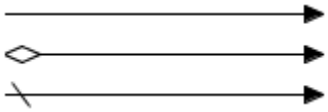
Brána (Gateway)

Bránou sa zabezpečuje rozdelenie a zlúčenie sekvenčného toku. Reprezentuje ho diamant, v ktorého strede sú zobrazené spresňujúce symboly. Tie špecifikujú, o aký typ vetvenia ide. Na výber máme rozhodovacie alebo paralelné delenie. K týmto deleniam sú definované zodpovedajúce zlučovania.

2.1.3 Spojovacie objekty (Connecting Objects)

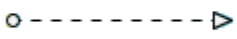
Tokové objekty sa spájajú a vytvárajú základnú kostru firemného procesu. Spojovacie objekty zabezpečujú toto prepojenie a taktiež umožňujú pripájanie artefaktov.

Sekvenčný tok (Sequence Flow)



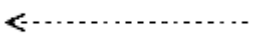
Sekvenčný tok reprezentuje neprerušovaná čiara s vyplnenou šípkou. Vyznačuje poradie, v akom sú aktivity v procese vykonávané.

Tok správ (Message Flow)



Tok správ reprezentuje prerušovaná čiara s prázdnou šípkou. Používa sa na znázornenie interakcie medzi dvomi separátnymi účastníkmi procesu, ktorí sú schopní prijímať a odosielať správy. Tok správ môže byť podmienený. Graficky sa podmienená správa označí umiestnením diamantu na začiatku čiary a podmienka je zaznamenaná v názve toku. Pri používaní podmienených správ je dôležité, aby bola splnená aspoň jedna podmienka na to, aby proces mohol pokračovať. Pre zaistenie pokračovania toku sa môže použiť implicitný tok, ktorý sa značí prešikrtnutím na začiatku čiary. Uplatňuje sa v prípadoch, ak sa všetky ostatné podmienené správy vyhodnotia záporne.

Asociácia (Association)



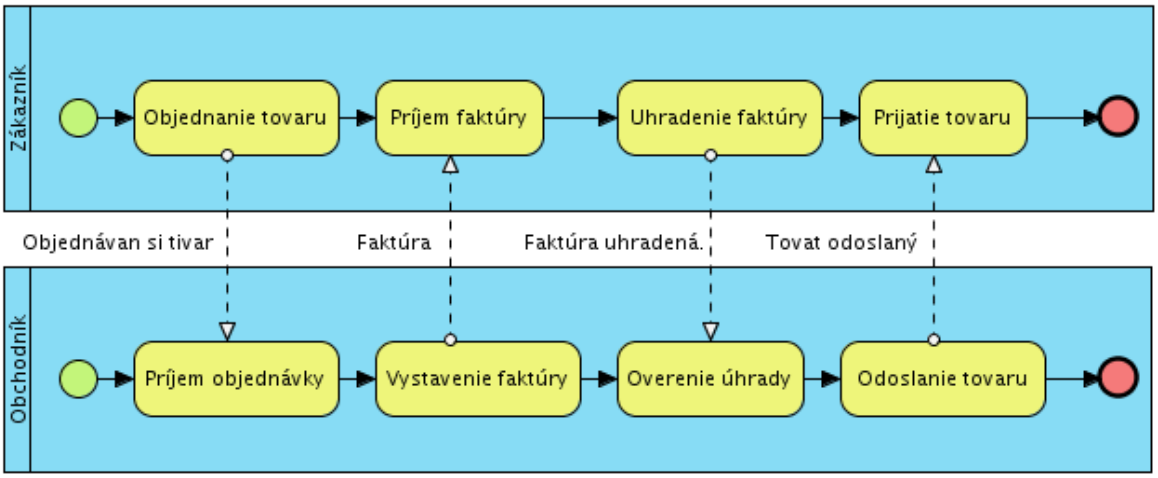
Na znázornenie asociácie sa používa prerušovaná čiara. Pomocou nej priraďujeme k tokovým objektom textové popisy alebo iné objekty, ktoré nepatria do skupiny tokových objektov. Pridaním šípky k prerušovanej čiare môžeme určiť smer priradenia.

2.1.4 Plavecké dráhy (Swimlanes)

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

Pool

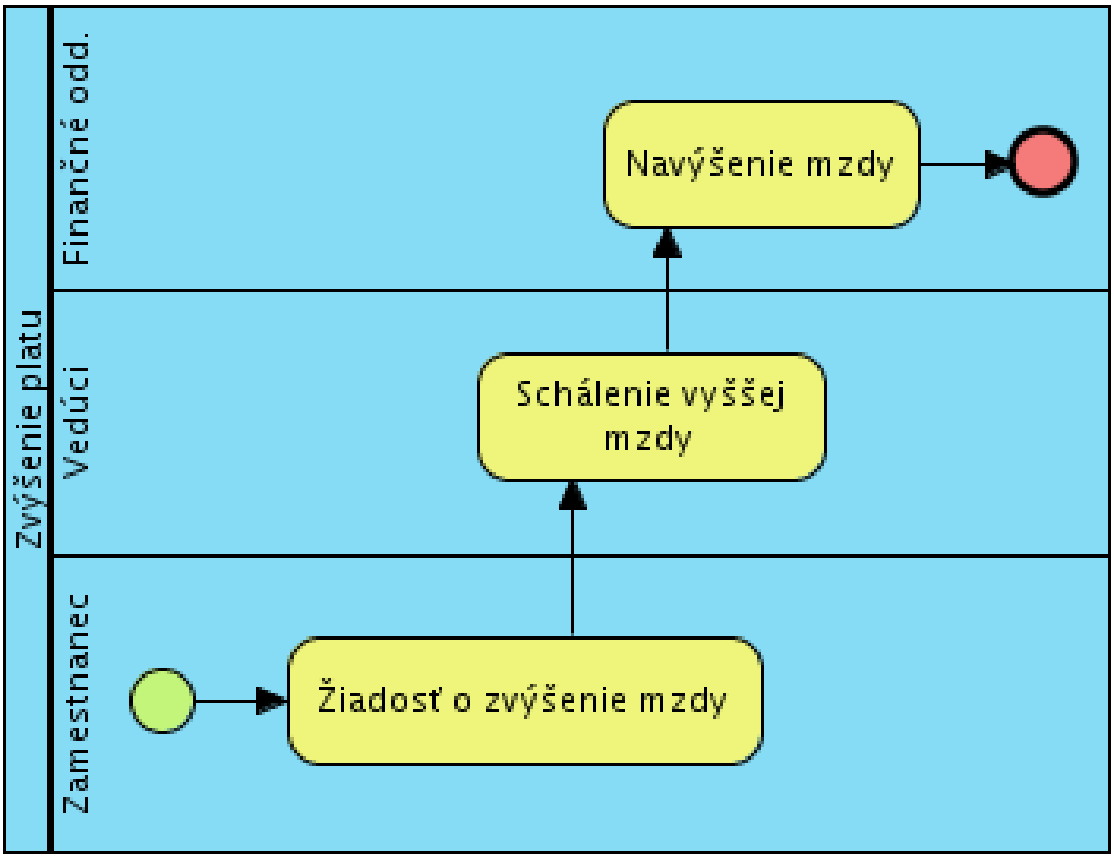
Pool ohraničuje proces a graficky vymedzuje jeho hranice. V rámci jedného poolu sa nachádza len jeden proces. Interakcia medzi poolmi prebieha pomocou správ. Pooly sa v diagrame používajú na zachytenie dvoch separátnych firemných entít alebo účastníkov. Proces každého účastníka je uzavretý v jeho poole, čím je stanovené jeho jasné ohraničenie. Zachytený proces nemôže interagovať s okolitými procesmi pomocou sekvenčných tokov. Na interakciu medzi 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. Slúži na usporiadanie a kategorizáciu aktivít. Môže 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]

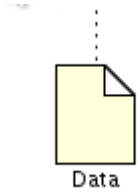


Obrázok 2.3: Príklad využitia dráhy.

2.1.5 Artefakty (Artifacts)

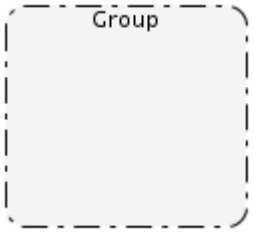
Artefakty neovplyvňujú základnú štruktúru procesu budovanú z aktivít, brán a sekvenčných tokov. Ponúkajú však spresňujúce informácie o elementoch procesu. Užívateľ si môže sám doplniť sadu artefaktov na uľahčenie a sprehľadnenie diagramov. V BPMN sú preddefinované len tri typy artefaktov.

Dátový objekt (Data Object)



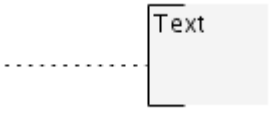
Dátový objekt slúži na zobrazenie toku dát v procese. Pomocou neho modelujeme, aké dáta sú požadované a aké dáta systém produkuje. Dátový objekt je k aktivitám pripájaný pomocou asociácie. Graficky je reprezentovaný obdĺžnikom s ohnutým rohom.

Skupina (Group)



Skupina je znázorňovaná prerušovaným obdĺžnikom a používa sa na dokumentačné a analytické účely. Nijako neovplyvňuje tok procesu.

Poznámka (Annotation)



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

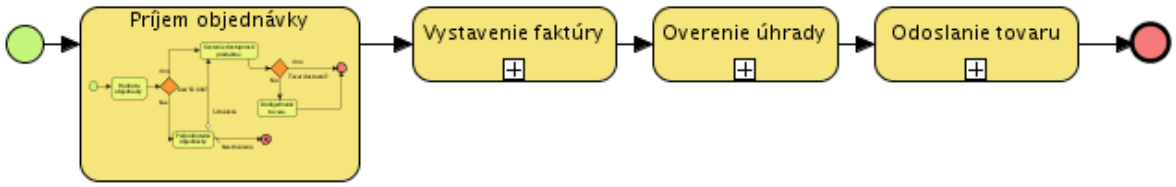
2.1.6 Využitie BPMN

Pomocou BPMN sme schopní modelovať procesy na rôznych úrovniach. Od detailného popisu jednotlivých čiastočných procesov ku globálnej orchestrácii firemných procesov, ktoré

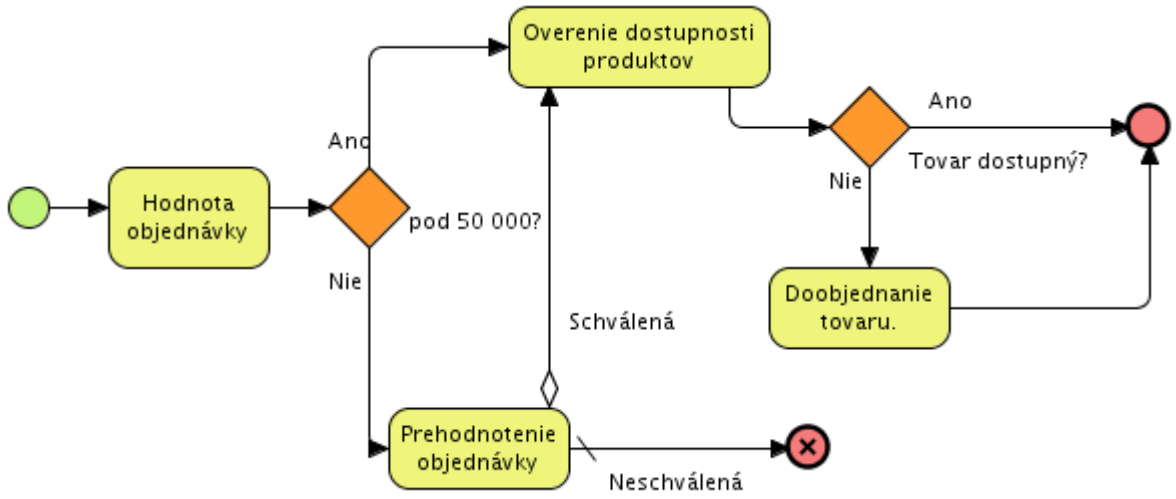
sa javia ako čierna skrinka. BPMN tým oslovuje rôznorodé publikum, ktorému predáva široké spektrum informácií na rozličných úrovniach detailu. Podľa miery záberu moderovaných procesov sa BPD delia na dve základné skupiny.

Kooperatívne medzifiremné procesy Kooperatívny typ diagramu zachytáva medzifiremné 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 medzifiremného procesu.

Interné firemné procesy Interné procesy firmy sú zachytené v hierarchii diagramov. Najvyššia úroveň zachytáva hlavný firemný proces, ktorý je prostredníctvom podprocesov 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 znázornený 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 najrozšírenejším modelovacím nástrojom. Jeho uplatnenie je široké a nie je používaný len v oblasti vývoja softvéru. Vzhľadom na

jeho veľké rozšírenie a predpokladanú zrejmosť notácie sa ním táto práca nebude podrobne zaoberať. Zmienené budú len niektoré diagramy, ktoré budú v práci využité.

2.2.1 Diagram prípadov použitia

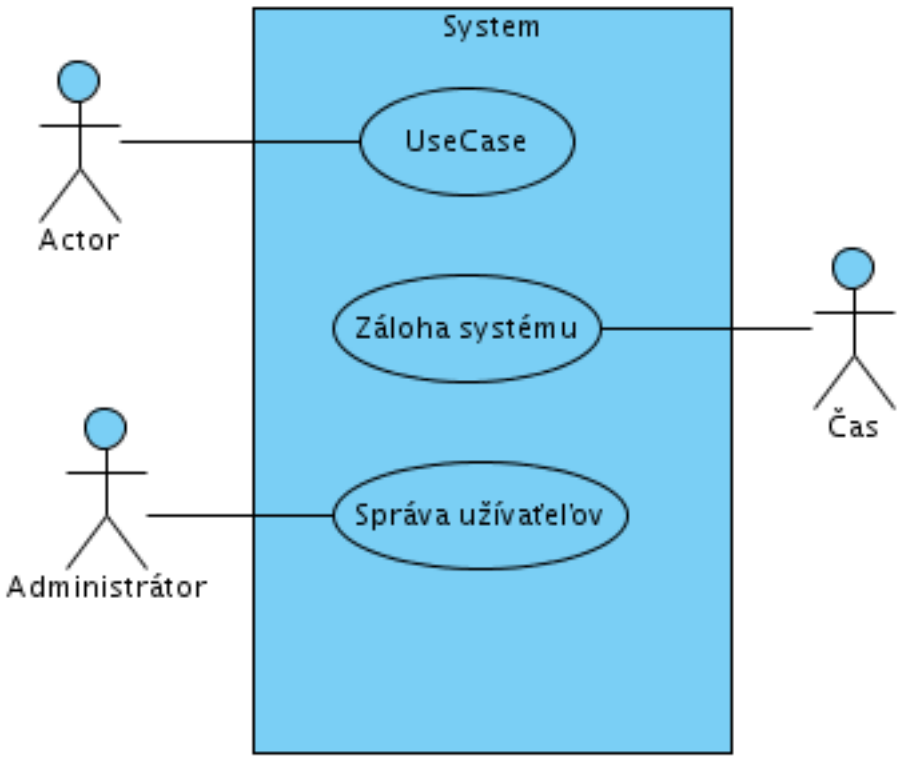
Diagram prípadov použitia (Use Case Diagram) slúži na zachytenie základných funkčných požiadaviek, ktoré má systém spĺňať. Diagram sa skladá z troch základných častí:

Hranice systému Vymedzujú modelovanú oblasť.

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

Prípad použitia Zachytáva ucelenú funkčnú jednotku systému.

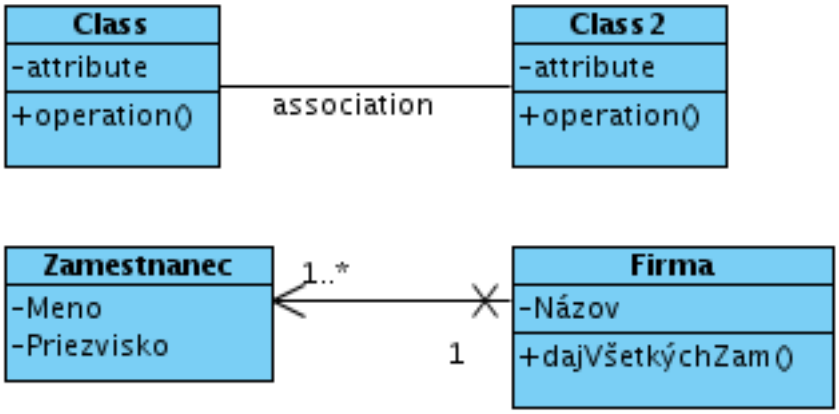
Pomocou týchto častí modelujeme interakciu okolitého sveta s modelovaným systémom. Diagram nesmie obsahovať interakciu medzi aktérmi, ani medzi prípadmi použitia. Súčasťou diagramu je aj detailná dokumentácia jednotlivých prípadov použitia. Na obrázku 2.6 je jednoduchý príklad diagramu.



Obrázok 2.6: Príklad na diagram prípadu použitia.

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 triedami. Môžeme určiť ich smer a kardinalitu. Na obrázku 2.7 je príklad diagramu tried.



Obrázok 2.7: Príklad na diagram tried.

Kapitola 3

Prístupy k vyvoju softvéru

V súčasnosti sa vo vývoji softvérových produktov v prevažnej 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. Na uľahčenie zvládnutia problému sa využíva rôzna úroveň abstrakcie. Zakrývajú sa ňou nepodstatné časti problému a pozornosť sa sústredí na podstatné 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 ucelená metodika na 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 požiadaviek
- Návrh
- Implementovanie
- Testovanie a integrovanie
- Údržbu

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

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 podľa 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

chyba vracia projekt na jeho úplný začiatok a môže ľahko viesť k jeho neúspechu. Napokon na tieto ťažkosti nadväzuje 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 zodpovedať jeho predstavám, a zasahovať do jeho vývoja. Nemá možnosť získať funkčné podčasti systému, s ktorými už môže pracovať, ale musí čakať až na ukončenie vývoja. Po ukončení projektu môže ľahko nastať situácia, kedy je zákazník prekvapený z výsledku, ktorý dostane.

3.2 Iteratívny / inkrementálny vývoj

Iteratívny / inkrementálny vývoj sa snaží o zníženie rizika zlyhania projektu. Celý projekt je rozdelený na časti podľa budúcej funkcionality systému. Pre každú časť sa vykoná analýza, návrh, implementácia a testovanie. Výsledný systém sa vybuduje z podčastí. Rozdelenie projektu umožní skoršiu detekciu chýb a hlavne nie je nutné prepracúvať celý systém, ale len časť, v ktorej sa vyskytla chyba. Pri využití tohto modelu vznikajú problémy s integráciou. Vzniká réžia, ktorá musí zabezpečovať funkcionalitu neúplného systému, ako napríklad vytvorenie protéz. Tvorí sa priestor na vznik nových chýb pri integrácii častí systému. Okrem technických problémov sa komplikuje aj časový návrh práce na projekte, lebo nie každá iterácia zaberá rovnaký čas. Úvodné iterácie 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, ktoré sú osobitne vytvárané 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 vymazaná a nahradená. Preferuje sa prepísanie zlého kódu namiesto jeho obchádzania. Tento postup sa v prevažnej väčšine prípadov spája s inkrementálnym vývojom a veľmi dobre spolu fungujú.

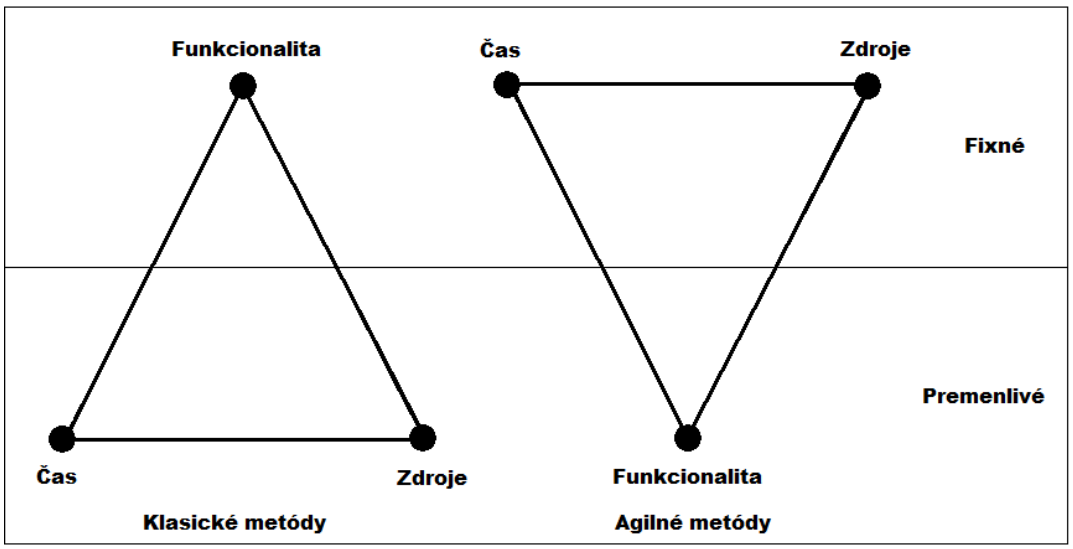
3.3 Agilné metódy vývoja

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 3.1 vidíme, že klasické metódy vývoja sa zakladajú na fixnej funkcionalite, ktorá je daná špecifikáciou požiadaviek. Funkcionalita je hlavným meradlom úspešnosti projektu. V prípade nesplnenie požiadaviek projekt zlyháva. Čas a prostriedky na daný projekt sa odvíjajú od funkcionality. Často sa preto stáva, že dochádza k posúvaniu termínov odovzdania a k zvyšovaniu zdrojov na zvládnutie projektu.

Agilné metódy sa k problému stavajú úplne opačne. Za fixné považujú zdroje a čas potrebný na zvládnutie projektu. Zákazníkovi dodajú nie vždy úplnú, ale pre neho najpodstatnejšiu funkcionalitu vždy v čase, keď ju potrebuje. Programy vyvíjané agilnými metódami

sú ľahko rozšíriteľné, a preto nie je problém s dodaním zvyšných častí systému. Ďalším dôsledkom nefixnej funkcionality je možnosť menenia požiadaviek zákazníka počas vývoja. Vyvíjaný produkt tým lepšie spĺňa zákazníkove potreby.



Obrázok 3.1: Porovnanie klasického a agilného prístupu.

3.3.1 Manifest agilného programovania

K agilným metódam vývoja patria viaceré metodiky. Všetky majú spoločný základ v častých kontrolách a úpravách, vysoko kvalifikovaných samoorganizovaných tímoch a zainteresovanosti zákazníka na procese. Filozofia agilných metód je zachytená v ich manifeste:

- Osoby a interakcia majú prednosť pred procesmi a nástrojmi
- Fungujúci softvér má prednosť pred obsiahlou dokumentáciou
- Spolupráca so zákazníkom je uprednostnená pred vyjednávaním o zmluvách
- 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 tím, 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 celý tím fyzicky situovaný na jednom mieste. Umožňuje to lepšiu spoluprácu na projekte a tým aj rýchlejšie produkovanie kódu. Programovanie často prebieha v skupinkách, ktorých členovia sa striedajú pri kódovaní daného úseku programu.

Manažéri a vývojári sú v tíme 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 v záujme dosiahnutia čo najlepších výsledkov.

Fungujúci softvér má prednosť pred obsiahlou dokumentáciou

Klasické postupy stavajú dokumentáciu na základe špecifikácie požiadaviek. Snažia sa tak zachytiť funkcionalitu budovaného systému. Výsledkom vývojového procesu je systém, ktorý zákazník na začiatku procesu definoval. Manažment sa týmto postupom 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 požiadaviek 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 požiadaviek, 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 samotný kód. Agilné 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 na 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 pracovanie so systémom počas jeho vývoja.

Spolupráca so zákazníkom je uprednostnená pred vyjednávaním o zmluvách

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

Klasické metódy dojednávania 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 požiadaviek. Klasické postupy sa pomocou plánovania snažia zamedziť zlyhaniu projektu a tým šetriť náklady na projekt. Ich výsledok však nemusí splniť momentálne očakávania zákazníka.

Agilné metódy volia iný prístup. Zmenám sa nedá vyhnúť, a preto sa proti nim 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 vo vývoji postupuje. Zákazník má možnosť ovplyvňovať prioritu jednotlivých funkcií: 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 časté a rýchle zmeny vo vývoji softvéru v dnešnom dynamicky sa meniacom svete. Dôraz je kladený na vývin softvéru, ktorý zodpovedá trhovým trendom a firemným potrebám. Do popredia sa vyzdvihuje súlad IT riešení s firemnými požiadavkami.

Klasické softvérové riešenia boli tvorené na základe stanovených požiadaviek. Systémy vzniknuté týmto spôsobom boli do veľkej miery neflexibilné. Ich rozširovanie bolo veľmi komplikované a nákladné. Znovu použitie funkcionality 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 s potrebami firmy.

Pri dosahovaní potrebného prepojenia firemných potrieb a IT sa môžeme odraziť od modelovania firemných procesov a rôznych metrík, ako napríklad návratnosť investícií (return of investments ROI), kľúčových indikátorov výkonu (key performance indicaotrs KPI) a ďalších.

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

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

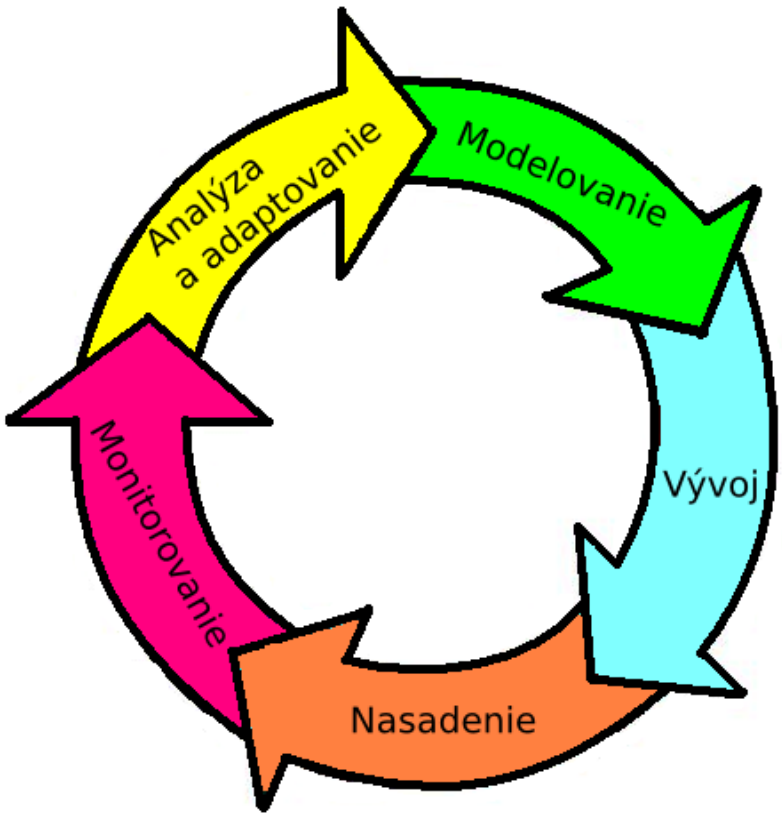
3.4.1 BDD model činností

Vzniká snaha na uzavretie priepasti medzi firemnými potrebami a IT riešeniami. Toto prepojenie však musí zostať pružné a prístupné pre vytváranie 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 služieb (services), ktoré sú platformovo nezávislé. Využitie SOA vyžaduje BDD prístup, ktorý spracúva firemné ciele a požiadavky zhora nadol, do návrhov, vývoja a testovania. Získavame tým prostredie, v ktorom sa aplikácie tvoria zo znovu použiteľných služieb alebo z nových služieb, ktoré sú vyvíjané podľa aktuálnych firemných potrieb. Tým sa zabezpečuje potrebná flexibilita v IT.

Nasledujúci obrázok 3.2 znázorňuje vysokoúrovňový 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 odo-



Obrázok 3.2: BDD model činností.

vzdanie 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 požiadaviek na systém. Modely vzniknuté v predchádzajúcej fáze sú hlavným vstupom. Identifikujú sa v nich prípady použitia (use case), od ktorých sa odvíja návrh služieb. Stále sa pritom berie ohľad na firemné procesy, ktoré zachytávajú vzájomné prepojenie jednotlivých prípadov použitia. Po tom, ako je služba implementovaná, pokračuje sa ďalšou fázou, a to nasadením. Služba je nasadená na aplikačný server, kde je verejne prístupný, a je možné ju vyhľadať.

Za uvedením služby do prevádzky následne začína fáza monitorovania. Zbierajú sa v nej informácie o behu služby v skutočnom čase, spracovávaných dátach a hláseniach. Sledovanie je doplnené o širokú paletu meraní podľa vopred definovaných metrík a výkonnostných parametrov. Fáza monitorovania je podstatná pre určenie funkčnosti a správnosti vytvorenej služby. Správne vytvorená služba musí spĺňať všetky požadované parametre.

Na záver nastupuje analyzovanie nazbieraných dát z predchádzajúcej fázy. Dáta sú dané na analýzu architektom, návrhárom a vývojárom. Všetci zúčastnení na 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 sa nové kolo vývoja. Ustavičným vylepšovaním existujúceho systému umožňuje tento mechanizmus rýchle prispôsobenie sa meniacim sa podmienkam.

V nasledujúcej časti bude pozornosť podrobnejšie venovaná vývojovému životnému cyklu.

3.4.2 Analýza firemných požiadaviek

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 najzaneprázdnenejšími manažérmi a vedúcimi. Výsledky treba zachytiť a zdokumentovať. Dokumenty by mali obsahovať minimálne:

- Firemnú víziu.
- Firemné ciele (krátkodobé a dlhodobé), ktorými uskutočňujú víziu spoločnosti.
- Firemné požiadavky vysokej úrovne, ktoré napomáhajú dosiahnuť ciele.
- Problémy s existujúcimi 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 služieb. Stanovuje sa ideálny firemný proces, ktorý je treba sa snažiť dosiahnuť.

Každá organizácia alebo jej sektor má vymedzenú firemnú funkcionalitu, ktorú podporuje alebo poskytuje. Pomocou BPM je možné modelovať tieto funkcionality. Každá úloha je pridelená role, ktorá zodpovedá nejakej entite alebo skupine entít. Rola môže byť pridelená viacerým entitám a jedna entita môže vystupovať vo viacerých rolách.

Firemný proces môže byť po analýze reprezentovaný postupnosťou aktivít a úloh. Úloha je najmenšia celistvá jednotka funkcionality, ktorá má 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.

Celistvý úsek procesu obyčajne zodpovedá istej funkčnej jednotke firmy, niekedy je však nevyhnutné priradiť funkčnú jednotku do organizácie procesu. Získavame tým hlboké pochopenie rozloženia zodpovedností a úloh vo firme, ktoré neskôr 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 nimi simulácie, prostredníctvom ktorých získavame informácie na 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 použitia

Pri modelovaní prípadov použitia je potrebné využiť firemné procesy, pomocou ktorých vytvárame jednotlivé prípady použitia. Roly z firemných procesov sú prevedené na aktérov a prípady použitia sa dajú získať prevedením jednotlivých firemných procesov, prípadne podprocesov. Vlastný prevod modelov poskytuje značnú mieru flexibility.

Na prevod nie je jasne definovaný formálny postup, ale je možné naznačiť použiteľný mechanizmus prevádzania BPMs na modely prípadov použitia.

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

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

Jedným spôsobom, ako sa k tomuto problému možno postaviť, je zavedenie krokov. Jeden krok sa dá definovať ako postupnosť úloh, ktoré môžu byť vykonané bez prerušenia tou istou rolou. Napríklad uloženie „ulož rezerváciu“ môže byť prípad použitia, ale „ulož rezerváciu a zašli dovolenkový leták“ nemôže byť prípadom použitia, lebo uplynie istý čas od momentu, ako je rezervácia uložená, až do momentu, keď je možné vytvoriť, prispôsobiť a zaslať dovolenkový leták užívateľovi. Na firemné procesy sa ale dá pozerať aj ako na postupnosť krokov, pričom každý krok je prípad použitia s požadovaným výstupom a správaním.

Hneď, ako sme schopní identifikovať prípad použitia, je nutné popísať jeho hlavné kroky cez aktivity, ktoré v ňom prebiehajú. Takýmto spôsobom získavame popis prípadu použitia pomocou postupnosti krokov.

Všetky identifikované prípady použitia majú význam, lebo vznikli identifikovaním krokov vo firemnom procese. Každý prípad použitia zodpovedá aspoň jednému firemnému procesu, či už celému alebo jeho oddielu, a je ho možné spojiť aspoň s jedným firemným procesom. Sú identifikované všetky prípady použitia, lebo každému kroku vo firemných procesoch je pridelený zodpovedajúci prípad použitia s popisom hlavných krokov.

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

3.4.5 Modelovanie služieb – SOA

Ak chce spoločnosť využívať servisne orientovanú architektúru (SOA), musí si vytvoriť portfólio služieb, ktoré spoločnosť poskytuje. Buď pre interné alebo externé použitie. Pri dosahovaní SOA môže spoločnosť dosiahnuť rôzne úrovne. Väčšina spoločností zavádza webové služby, no sú aj omnoho vyššie úrovne implementovania SOA do spoločnosti. Určenie úrovne, na ktorej sa firma nachádza v rámci presadzovania SOA, je sám o sebe zložitý proces. Na to, aby spoločnosť dosiahla skutočné SOA, je potrebné pochopiť, prijať a nasledovať metodiku.

Proces vývoja informačného systému pre firmu, v ktorom sa využije SOA architektúra, bude pokračovať spracovaním výsledkov firemnej analýzy na identifikovanie a navrhnutie služieb. Je potrebné začať stavať na výsledkoch minulej fázy, na firemnej vízii a cieľoch organizácie. Pri automatizácii firemných procesov treba dodržať prioritné kritériá. Pri ich určení úzko spolupracuje IT a firemný manažment. Musia prehodnotiť kľúčové problémy firmy ako napríklad:

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

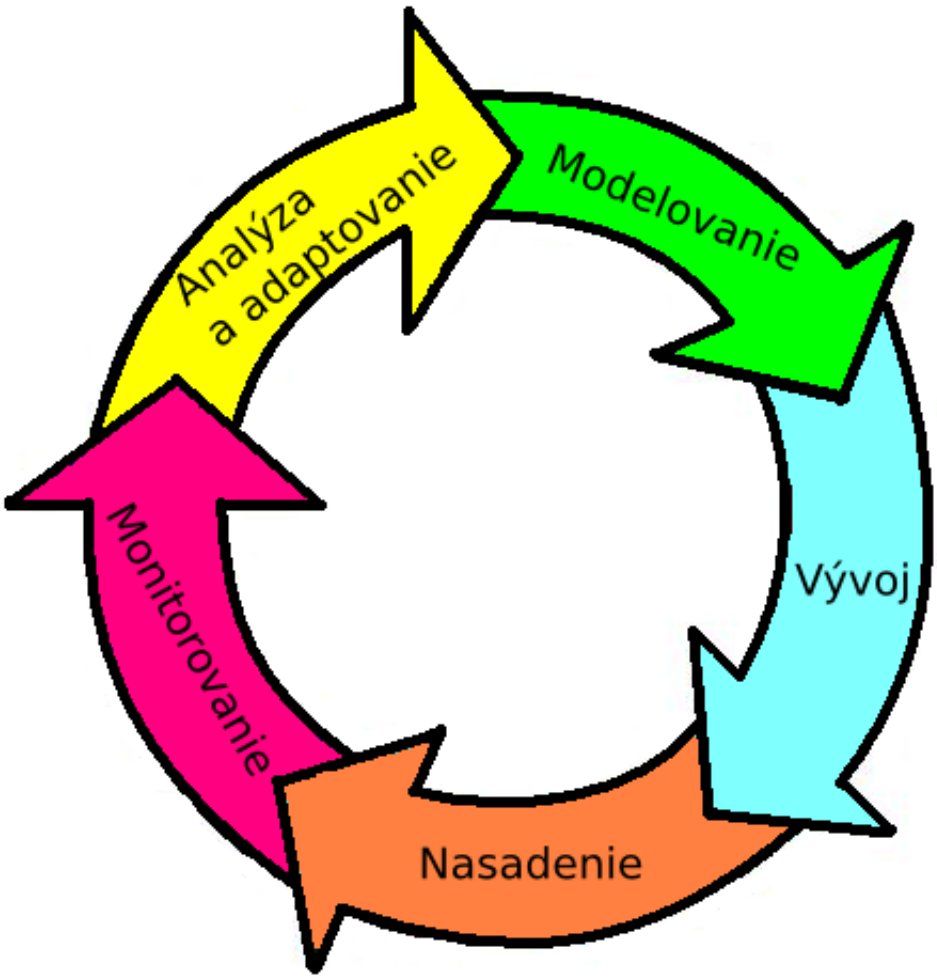
Na základe takýchto analýz môže byť stanovený plán, v akom poradí budú procesy vyvíjané. Procesy sú vlastnené funkčnými časťami firmy. Pár týchto jednotiek môže byť na začiatku vybraných, aby sa na ne IT sústredilo a získal sa tým krátkodobý okamžitý prospech. Akákoľvek služba, ktorá môže byť identifikovateľná, musí podporovať ciele firmy, ktoré sa snaží sčasti alebo úplne vyriešiť. Týmto si možno pomôcť pri uprednostňovaní funkčných častí firmy pre IT vývoj. Pre veľmi dôležité zložky firmy budú následne identifikované a definované služieb. Problémy s existujúcimi firemnými procesmi môžu byť použité ako hlavné ciele zavádzania služieb. V každom prípade pri určovaní priority zamerania IT treba mať na pamäti firemné potreby.

Každý firemný proces, ako aj prípady použitia, ktoré boli identifikované v predchádzajúcich fázach vývoja systému, môžu byť pripísané na zoznam kandidátskych služieb. No nie všetky kandidátske služby sú aj realizované. Zmyslom SOA je zostaviť portfólio služieb, ktoré budú znovu použiteľné v čím viacerých firemných procesoch. Implementácia týchto služieb musí byť taktiež dobre premyslená, aby sa jednotlivo implementované časti dali využiť v rôznych implementáciách služieb.

Obrázok nám poskytuje náhľad na rôzne úrovne S0A. Zobrazuje kroky firemného procesu, ktoré sú prevedené na jednu službu alebo na skupinu viacerých služieb. Taktiež zobrazuje, ako môže byť služba implementovaná komponentmi. Tieto komponenty môžu byť pozostatky bývalého systému, komerčne dostupné riešenia alebo úplne od nuly vyvíjané komponenty.

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

- Analýza zhora dolu cez procesné modely.
- Analýza už existujúceho systému a aplikácií.
- Vytvorenie prvotného modelu služieb pre kandidátske služby.
- Jasné stanovenie popisov vlastností a kvality služieb.
- Zviazanie každej služby s firemným cieľom a zabezpečenie, aby sa služba podieľala aspoň na jednom firemnom procese.



Obrázok 3.3: Úrovne servisne orientovanej architektúry

3.4.6 Systémový návrh a vývoj

Počas vývojovej fázy tím pracuje s počiatočným prípadom použitia a modelom služieb. Vnára sa do modelu a špecifikuje funkcionalitu pre každý prípad použitia.

Systémový návrh

Na vytvorenie modelu systémových komponentov využijeme popis funkcionality jednotlivých prípadov použitia. Model komponentov popisuje rozhranie komponentov, ktoré ponúkajú, a zachytáva taktiež vzťahy medzi jednotlivými komponentmi v celom systéme. Delenie systému na komponenty zodpovedá navrhnutým službám v modeli služieb. V modeli môžu byť zahrnuté aj komponenty, ktoré priamo neimplementujú služby, ale poskytujú len podpornú funkcionalitu. Komponenty, ktoré nemajú rozhrania, sa nedajú externe využívať, tak sa využívajú len na vnútornú komunikáciu. Pre dôležité prípady použitia sú vytvárané sekvenčné diagramy cez rozhrania komponentov. Takto je systém navrhnutý, aby komponenty z modelu komponentov medzi sebou dobre komunikovali cez rozhranie.

V tejto fáze sa taktiež identifikujú a dokumentujú nefunkčné požiadavky na systém. Nevyužíva sa len vytvorenie SLA pre služby, ale sú tiež 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 služieb, ich popisy a spôsob vyvolania. Nie všetky kroky procesov sa však implementujú ako priame vyvolania služieb. Jedným z hlavných dôvodov, 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 makronávrhu na modelovanom diagrame komponentov a operačného modelu pristúpime k mikronávrhu, ktorý zahŕňa modelovanie diagramov tried a sekvenčných diagramov pre každý komponent 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 (napr. Java 2 Platform, Enterprise Edition), programovací jazyk (napr. 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 služby sú medzi sebou previazané pomocou nástroja na koordinovanie procesov, ktorý poskytuje možnosť previazať služby cez rozhrania, ktoré poskytujú podľa navrhnutých firemných procesov.

Dôležitým aspektom takéhoto prístupu je, že jedna služba môže byť použitý vo viacerých procesoch, čo je hlavným prínosom BDD prístupu. Firemné procesy vznikajú preskladaním služieb podľa potreby. Tento prístup poskytuje veľkú flexibilitu, lebo nevyhovujúce procesy sú vytvorené z nových lepších služieb a vyvinuté služby sa zas používajú v iných procesoch. Nové procesy nie sú tvorené úplne od nuly, ale môžu byť tvorené už z existujúcich služieb. 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 musí byť po dokončení vývoja 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.

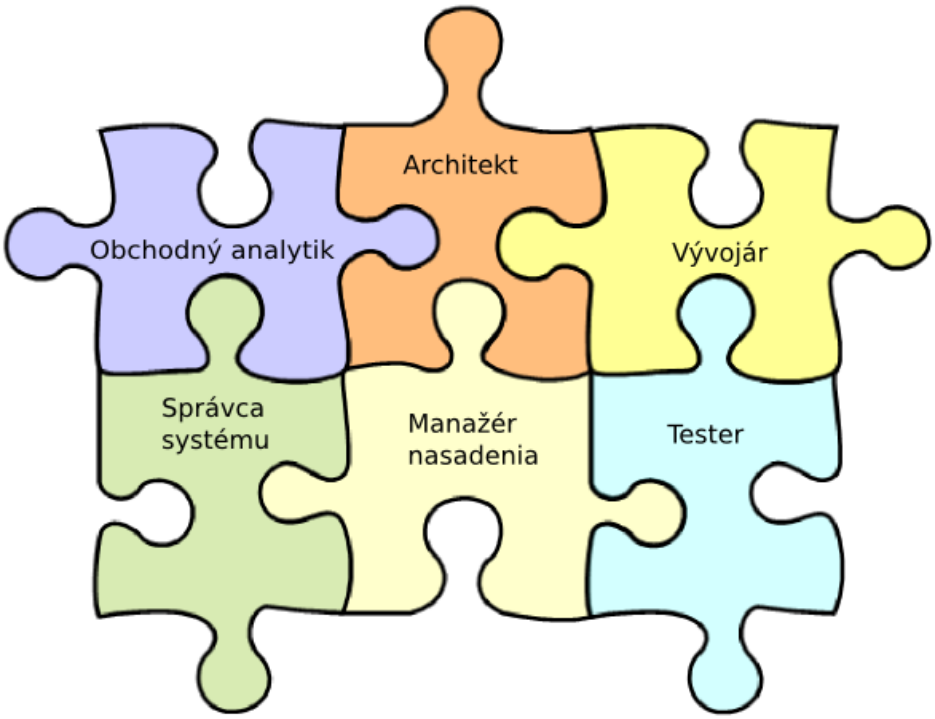
Ako prvé sa prevedie podrobná analýza implementovaného zdrojového kódu. Je tu snaha odhaliť miesta, ktorých prepísaním zvýšime výkon systému. Ak po ukončení úpravy kódu nezískame žiadne priblíženie sa k požadovanému výkonu pristúpime k opatrnej zmene firemného procesu. Prevedená zmena na procese musí 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 3.4 znázorňuje šesť najdôležitejších rolí, ktoré sú vyžadované na vykonanie BDD vývojového cyklu.

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 neskor-



Obrázok 3.4: Hlavné role využívané v BDD.

ších 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 robustnejšia 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 činností 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 na základe funkcionality. Pre každú čiastočnú úlohu sa potom vykonáva analýza, návrh, implementácia a testovania. Prebieha tu vlastne malý vodopád. Pracovanie s menšími funkčnými časťami, uľahčuje zvládnutie aj zložitejších projektov. Koncový produkt získame po vykonaní všetkých iterácií. Často sa tento prístup kombinuje s inkrementálnym vývojom, ktorého

prínosom je postupné vyvíjanie systému v samostatných častiach. Projekt je rozdelený na inkrementy, ktoré sa osobitne vyvíjajú. Inkrementy sú po ich dokončení integrujú do výsledného riešenia. Podrobnejšie boli obidva modely popísané v tretej kapitole 3.2.

Z vopred získaných znalostí je ako najvhodnejší prístup vybraný iteračno-inkrementálny model. Umožňuje nám vyvíjať aplikáciu postupne, čím znižuje riziko zlyhania projektu a taktiež umožňuje dodávať zákazníkovi priebežne verzie systému, ku ktorým sa zákazník môže vyjadriť. Vybraný model si mierne poopravíme pre naše potreby. Keďže pracujeme s malými softvérovým projektami, nie je zapotreby využívať veľké množstvo iterácií a inkrementov. Určenie počtu jednotlivých vývojových oddielov bude hlavne závislé na funkcionalite budúceho systému a na tom ako bude systém dodávaný zákazníkovi.

4.1.2 Agilné prvky

Pri súdobom vývoji aplikácií sa stáva jedným z najdôležitejších faktorov schopnosť vývojárov reagovať na rýchlo meniace sa prostredie a požiadavky na vyvíjaný systém. Klasické metódy sú v súčasnej dobre ťažko efektívne využiteľné, lebo neposkytujú riešenia vo chvíľach, keď sú zapotreby. Problémom zvýšenia schopnosti prispôsobenia sa aktuálnym potrebám zákazníka sa zaoberala časť práce o agilných metódach vývoja 3.3.

V navrhovanej metóde sa budeme snažiť využiť vlastnosti agilných metód, aby bola zabezpečená čo najvyššia flexibilita vývoja.

Jedným z hlavných bodov v agilných postupoch je vysoká miera zainteresovania zákazníka do procesu vývoja. Napriek tomu že navrhovaná metóda sa má zaoberať malými projektami, ktoré by nemali byť veľmi komplikované na pochopenie, musí prevládať snaha o zapojenie zákazníka do vývoja. Pri dôležitých rozhodnutiach o poradí vývoja častí aplikácie a pri otáznych bodoch vývoja by zákazník nikdy nemal chýbať. V prípade nezáujmu zo strany objednávateľa môže nastať zníženie kvality, prípadne dodanie úplne nežiadúceho systému.

Cieľom agilných metodík je čo najrýchlejšie sa dostať do fáze programovania. Modelovacia a dokumentačná činnosť je až na druhom mieste. V novej metóde je tiež snaha o ušetrenie nákladov rýchlejším pristúpením k programovaniu. Modelovanie a dokumentovanie sa však nepovažuje za činnosť ktorá spomaľuje vývoj. Dobrý návrh je priam naopak spôsob ako si čas pri vývoji ušetriť. Napriek tomu sa ale pre efektivitu obmedzíme na tie najpotrebnejšie modely systému, ktorými je jasne stanovená funkcionalita. Drobné implementačné detaily sa zbytočne pracne nemodelujú, ale vychádzajúc z kvalifikovanosti programátora, aby daný problém vyriešil.

Spolupráca a komunikácia je v riešiteľskom tíme maximálne dôležitá. Pracovníci musia byť ochotný pomáhať si jedne druhému. Rozširujú tým svoje znalosti, zvyšujú priemernú kvalifikovanosť tímu a dokážu sa skôr vysporiadať s problémami. Minimalizuje sa formálna komunikácia, ktorá často spôsobuje len zbytočnú záťaž na vývojový proces a oddaľuje ľudí v tíme. Z charakteru projektov, pre ktoré je metóda navrhovaná vyplýva, že na ich zvládnutie nebude potreba veľké množstvo pracovníkov. Pre tieto dôvody by nemalo byť obtiažne splniť vyššie spomínané požiadavky na tímovú prácu. Všetky tieto kroky sú vedené na vytvorenie pozitívnej, konštruktívnej atmosféry v riešiteľskom týme, ktorá motivuje pracovníkov k podávaniu maximálneho výkonu a produkovania kvalitných riešení.

4.1.3 Využitie BPMN

BPMN slúži ako grafický jazyk, ktorý je špecializovaný pre popis firemných procesov. Podrobný popis syntaxe bol podaný v druhej kapitole. 2.1. Modelovanie firemných procesov bude slúžiť na objasnenie fungovania organizácie a presne určenie procesov, ktoré v nej prebiehajú. Vďaka týmto procesom ľahko identifikujeme časti, ktoré je možné automatizovať. Identifikujeme prípady užitia a namodelované procesy využijeme ako grafické znázornenie krokov, ktoré treba pre daný prípad užitia vykonať. Touto činnosťou získame sadu diagramov, ktorá zobrazuje firemné procesy prepojené na diagramy prípadov užitia.

Diagramy firemných procesov sú pre zákazníka veľmi intuitívne a podrobne popisujú systém, akým organizácia pracuje alebo akým sa určitá modelovaná činnosť vykonáva. V kontraste s klasicky používaným diagramom prípadov užitia, ktoré zachytávajú len heslovite požadovanú funkcionalitu, vidí zákazník presné postupy, s ktorými sa môže ľahko stotožniť. Získavame tak silný graficky model, v ktorom sa zákazník ľahko dokáže orientovať. Nad vypracovanými diagramami je možnosť ľahšej komunikácie so zákazníkom, lebo má presnejšiu predstavu o tom, čo sa v budúcom systéme ocitne. Môže komentovať presnosť procesov a prípadne spolu s vývojárom diskutovať o ich vylepšení.

4.1.4 Inšpirácia BDD

Definovaná metóda čerpá veľkú inšpiráciu z Business Driven Development-u, ktorému sme sa venovali v tretej kapitole. 3.4 Preberá z BDD hlavnú myšlienku priblíženia vývoja k potrebám zákazníka. Produkované systémy majú spĺňovať aktuálne požiadavky a majú byť doručené čo najskôr, aby poskytli pre klienta potrebnú konkurenčnú výhodu.

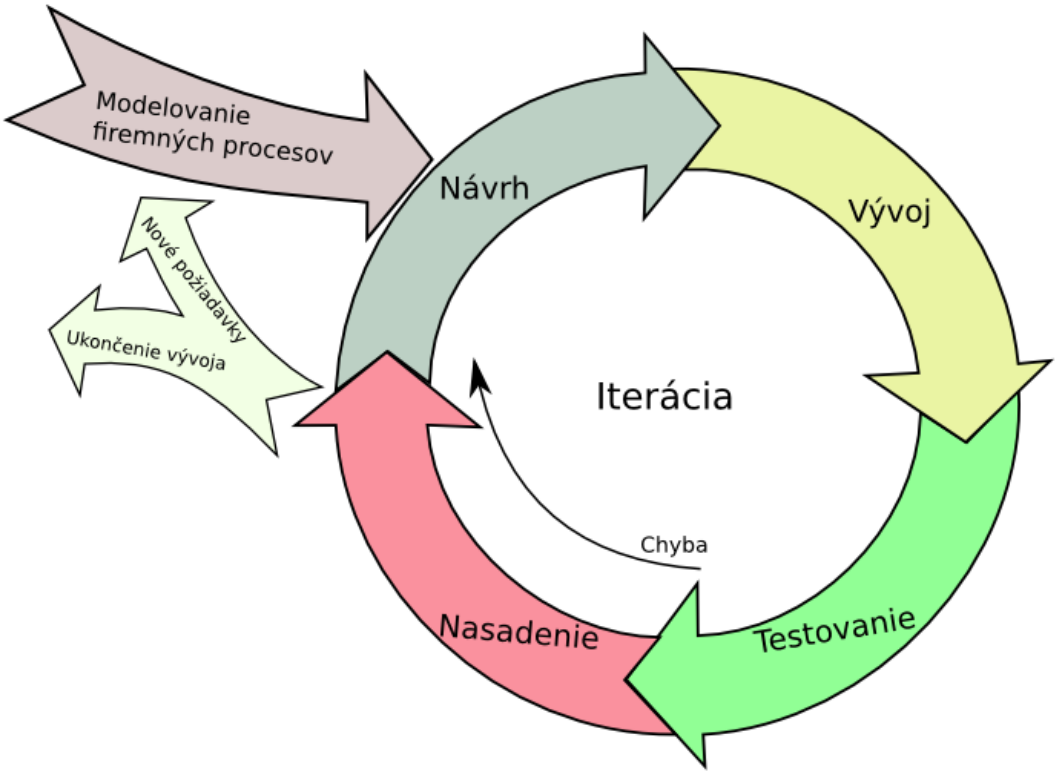
Nová metóda vychádza zo životného cyklu definovaného v BDD. Zjednodušuje cely cyklus a upravuje jednotlivé fázy, aby zaberali čo najmenej času a úsilia. Zmeny ale neovplyvňujú robustnosť riešenia a nevnášajú neistotu z neúspechu projektu. Podobne ako bol pretransformovaný životný cyklus, sú pozmenené aj role. Ich kompetentnosti sú v niektorých prípadoch rozšírené a niektoré role sú naopak okresané prípadne úplne odstránené.

Podstatnou inspiráciu bola prvá fáza, ktorá sa venuje firemných procesom a ich modelovaniu. Stala sa hybnou silou novo-navrhnutej metódy. Nový postup mení spôsob a význam použitia modelovania firemných procesov. Využíva ich na zachytenie požiadavkov na systém a podobné grafické znázornenie procesov v organizácii, z ktorého je ľahké určiť prípady užitia a ich popis. Klient tak získava zrozumiteľnejší náhľad na budúci systém.

Novo-definovaná metóda prináša aj nové špecifiká, ktoré sú ovplyvnené hlavne myšlienkami plynúcimi z manifesta agilného programovania. Sú to hlavne myšlienky týkajúce sa prace v tíme a potlačenia nadbytočnej byrokracie. Ako bolo už vyššie zmienené, práca v tíme má byť inšpiratívna a povzbudzujúca k podávaniu čo najlepších výsledkov.

4.2 Životný cyklus

Pre navrhovanú metódu je definovaný životný cyklus, ktorý je zobrazený na obrázku 4.1. Pokrýva vývoj od návrhu až po nasadenie systému. Má cyklický charakter, ktorý nám naznačuje iteratívno-inkrementálne vylepšovanie systému pri vývoji. Proces vývoja začína modelovaním firemných procesov. Po tejto fázy nastane identifikácia funkčných častí na zákla-



Obrázok 4.1: Schéma životného cyklu navrhovanej metódy.

de, ktorej sa projekt rozdelí na menšie podčasti. Jednotlivými časťami je určená priorita a postupne prechádzajú vývojovými fázami. Ako prvou prechádzajú návrhom systému. Vyprodukované modely slúžia ako vstup do vývojovej fázy, ktorá je nasledovaná testovaním. Po ukončení testovania sa použiteľná časť systému zavádza do prevádzky. Cyklus pokračuje spracovaním ďalšej funkcionality. Pri výskyte nových požiadavkov je na ne možné reagovať pozmenením modelov a preusporiadaním postupnosti funkčných častí. Po ukončení poslednej iterácia a v prípade nevyskytnutia sa nových požiadavkov je vývoj softvéru ukončený.

Modelovanie firemných procesov – Účelom prvej fáze vývojového procesu je za pomoci klienta zozbierať požiadavky na budúci systém. Spísať špecifikáciu požiadavkov a hlavne namodelovať firemné procesy. Po odsúhlasení modelov zákazníkom sa určia prípady užitia a rozdelí sa funkcionalita. Zákazník určí prioritu jednotlivých ďalej vyvíjaných častí.

Návrh – Do návrhovej fázy vstupujú modely z predchádzajúcej činnosti. Tieto modely sú spracované na návrh budúceho systému. Prípady užitia sú popísané pomocou krokov vyplývajúcich z krokov firemných procesov. V prípade potreby sú v zriedkavých prípadoch bližšie modelované sekvenčnými diagramami. Statickú štruktúru budovaného systému zachytávame pomocou diagramu tried.

Výsledkom návrhovej činnosti je skromná sada diagramov, ktoré musia byť kombinované s vopred vzniknutými diagramami firemných procesov. Získaná výsledná sada modelov nám dostatočne popisuje budúci systém. Ako posledný bod návrhovej

činnosti je určená architektúra, pomoci ktorej bude systém vyvíjaný (napr. Java, ASP .Net atď.).

Vývoj – Pri implementácií budúceho systému vychádzame z vopred vypracovaných modelov a zvolenej architektúry. Riešenie implementačných detailov na najnižších úrovniach je ponechané na odbornej znalosti programátora. Problémy s porozumením diagramov či s kódovaním určitej časti systému je riešené komunikáciou a dohodou v rámci riešiteľského tímu. Konečným produktom vývoja je testovateľná časť systému.

Testovanie – Odhaľovanie chýb je dôležitou časťou každej metódy používanej pri súčasnom vývoji softvéru. Z dôvodu zabezpečenia maximálnej kvality každého produktu nesmie byť táto fáza opomenutá ani v navrhovanej metóde. Pri testovaní prejde systém sadou štandardných postupov testovania ako napr. automatizovanými testami, testami typu biela a čierna skrinka, akceptačnými testami atď. Softvér, ktorý spĺňa požiadavky na funkčnosť, výkon a kvalitu je posunutý do ďalšej fázy životného cyklu. Naopak pri nesplnení funkčných požiadavkov sa projekt posúva do ďalšej iterácie, v ktorej budú chyby odstránené.

Nasadenie – Na záver je otestovaná časť systému nasadená do prevádzky. Zákazník má možnosť pracovať s odovzdanou časťou systému. V prípade vzniknutia neočakávaných nedostatkov produktu môže zákazník vyžiadať úpravu vyvíjaného riešenia. Po úspešnom nasadení verzie systému si môže zákazník určiť nasledujúcu funkcionalitu, ktorá bude spracovaná a pridaná do produktu. V prípade nezmenia sa požiadavkov pokračuje cyklus podľa vopred stanovenej priority vývoja častí.

4.3 Využité role

Procesy prebiehajúce v navrhnutej metóde vyžadujú na zvládnutie znalosti z viacerých oborov informatiky a manažmentu. Osoby pracujúce na vývoji softvéru sa ocitajú v niekoľkých preddefinovaných roliach. Každá z rolí vykonáva špecifickú úlohu počas životného cyklu tvorenia produktu a je nepostrádateľná na úspešné dosiahnutie cieľa.

Pretože sa venujeme produkovaním malých softvérových riešení, nie je pravdepodobné, že riešiteľský tím bude obsahovať veľké množstvo pracovníkov. Vďaka tomu, že na produkovaní riešenia sa bude podieľať pomerne malý tím, umožní to intenzívnu komunikáciu a podporu medzi jeho členmi. Jednotlivé osoby sa počas trvania projektu môžu dostať do viacerých rolí v prípade, že splňujú potrebnú kvalifikáciu.

Navrhovaná metóda definuje nasledujúce role:

Firemný analytik – Jeho úlohou je komunikovať zo zákazníkom, pochopiť jeho potreby a fungovanie prostredia, do ktorého bude systém vyvíjaný. Na základe zozbieraných znalostí vypracuje firemný analytik sadu modelov, ktoré zachytia procesy prebiehajúce v organizácií. Opierajúc sa o tieto diagramy komunikuje zo zákazníkom o jich korektnosti a spoločne s návrhárom systému môžu diskutovať o možných vylepšeniach procesov.

Návrhár – Jeho hlavnou úlohou je zachytenie dynamickej a statickej štruktúry systému. Využíva na to modely vyprodukované pri analýze podniku a komunikuje o riešeniach

s firemným analytikom. Vo firemných procesoch identifikuje prípady užitia, čím zväzuje tieto modely s modelmi, ktoré zachytávajú návrh systému. Má na starosti navrhnutie vykonateľných technických riešení k požiadavkám od zákazníka. Úlohou návrhára je tiež voľba vhodnej architektúry systému po dohode so zákazníkom.

Vývojár – Má na starosti prevedenie navrhnutých modelov do fungujúcej a nasaditeľnej podoby. Väčšiu nízko úrovňových otázok týkajúcich sa implementačných detailov má na starosti vyriešiť sám. V prípade nepochopenia zadania alebo pri prípadných komplikáciách pri svojej práci môže konzultovať ich riešenia s návrhárom.

Tester – Má na starosti overenie funkcionality a zaistenie kvality odovzdaného systému. Tým že pracuje s konečnou funkčnou podobou časti systému, je táto roľa rozšírená aj o zodpovednosti týkajúce sa nasadenia systému. Pomáha zákazníkovi po predaní produktu z jeho nasadením a poskytuje mu informácie týkajúce sa jeho fungovania. Prípadné konfigurovanie produktu spadá tiež do povinnosti testera.

Pre zákazníka nie je stanovená rola, lebo jeho pozícia pri vývoji odpovedá jeho osobe. Má však vo všetkých prebiehajúcich procesoch významnú úlohu. Podieľa sa pri vývoji produktu a to nie len počas počiatočnej fázy, keď sú zbierané požiadavky na systém, ale aktívne sa podieľa aj na ovplyvňovaní vyvíjaného produktu počas celej doby vývoja. Neseriózny prístup k vývoju zo strany objednávateľa vážne narušuje úspešnosť zvládnutia projektu a kvalitu odovzdávaného produktu.

4.4 Vývoj softvérového projektu

V nasledujúcej časti tejto práce sa budeme venovať postupu, akým bude softvér vyvíjaný pomocou novo vyvíjanej metódy pre spracovanie malých softvérových projektov. Pozornosť bude venovaná každej fázy životného cyklu a podrobne budú popísané role a činnosti, ktoré sa jej zúčastňujú.

4.4.1 Špecifikácia požiadavkov

Úvodná činnosť pri vyvíjaní softvérového projektu musí byť venovaná zbierania poznatkov o systéme, ktorý budeme tvoriť. Táto činnosť spadá do fázy modelovania firemných procesov. Firemný analytik intenzívne komunikuje zo zákazníkom. Pri vzájomnej komunikácií si analytik objasňuje nároky, ktoré bude zákazník klásť na budúci systém. V prípade, že s budúcim systémom bude pracovať širší okruch pracovníkov vo firme, musia byť do zbierania požiadavok zahrnutý aj oni.

Na základe zozbieraných požiadavkov od zákazníka je zostavený dokument špecifikácie požiadaviek na budúci systém. Pretože pracujeme z neveľkými projektami špecifikácia by nemala byť extrémne rozsiahla a zákazník by nemal mať problém preštudovať ju a vyjadriť k nej svoje pripomienky. Po dokončení špecifikácie požiadaviek a jej odsúhlasení zo strany klienta môžeme postúpiť k ďalšej činnosti.

Míľnikom pre ukončenie prvej časti projektu je odsúhlasená špecifikácia požiadavkov.

4.4.2 Návrh firemných procesov

Podobne ako predchádzajúca činnosť spadá aj návrh firemných procesov do prvej fázy životného cyklu. Skladá sa z viacerých činnosti, ktorých cieľom je graficky spracovať dokument špecifikácií požiadaviek získaný v predchádzajúcou činnosťou.

Podrobným preštudovaním špecifikácie a na základe získaných informácií z prostredia, kde bude systém nasadený, firemný analytik začne modelovať firemné procesy. Na ich modelovanie využíva BPMN. Vzniknuté modely podrobne popisujú postup činnosti, ktoré vedú k vykonaniu určitého procesu vo firme. Jednotlivé diagramy obsahujú okrem sledu úloh aj role zodpovedné za vykonanie určitých úloh. V procesoch identifikujeme pasáže, ktoré môžu byť automatizované a pridelíme ich vykonanie systémovej roli. Na prideľovanie činností k roliam použijeme koncepciu plaveckých dráh, ktoré poskytuje BPMN.

Odporúčaný spôsob návrhu firemných procesov je využitie prístupu modelovania zhora na dol. Získavame tým hierarchickú štruktúru diagramov, kde v najvyššej úrovni zachytávame globálny pohľad na hlavné činnosti vykonávané v organizácií. Postupnou dekompozíciou týchto hlavných aktivít získavame podrobný popis aktivít definovaných na vyššej úrovni. Úlohy na najnižšej úrovni dekompozície zachytávajú atomické činnosti, ktoré sú vo firme vykonávané, a využívajú sa neskôr ako kroky prípadov užitia.

Po namodelovaní kompletnej hierarchie procesov, konzultujeme výsledky zo zákazníkom a vykonávame prípadne nutné úpravy. Po odsúhlasení modelov zo strany zákazníka nasleduje porada z návrhárom systému. Celý systém je rozdelený na menšie časti podla funkcionality, ktoré predstavujú iterácie. Vymedzený oddiel musí poskytovať ucelenú funkcionalitu, prínosnú pre zákazníka. Jednotlivým častiam je priradená priorita na základe klientových požiadavkov, podla ktorej je systém postupne vyvíjaný. Zostavením plánu ne určenie poradia iterácií a inkrementov je ukončená prvá fáza životného cyklu.

Míľniky pre fázu návrhu firemných procesov sú hierarchická sada diagramov, ktorá popisuje budúci systém a plán iterácií a inkrementov podľa ktorého bude systém postupne vyvíjaný.

4.4.3 Návrh systému

Prvým významným rozhodnutím pre návrhára pri navrhovaní systému je zvolenie vhodnej architektúry pre budovaný produkt. Musí byť určený programovací jazyk, v akom bude systém implementovaný. Po dohode s klientom sú určené technológie, ktoré budu pri realizovaní projektu využité. Pri tejto činnosti je dôležité odborne poradiť klientovi, ktoré riešenie je pre jeho potreby najvhodnejšie.

Hlavnou činnosťou návrhovej fázy životného cyklu je spracovanie modelov firemných procesov. Identifikujeme v nich prípady užitia, ktoré graficky popíšeme časťami diagramov, z ktorých sme vychádzali. Jasne sa tým previažu diagramy firemných procesov s prípadmi užitia, ktoré budú ďalej podľa potreby spracúvané.

Určenie prípadov užitia by malo byť pomerne jednoduché a mechanické. Po preštudovaní firemných procesom vyberieme z nich časti, ktoré boli určené na automatizáciu a neprerušený sled krokov označíme ako prípad užitia. Neprerušenosťou myslime to, že počas vykonávania sekvencie úloh, nezasahuje do ich vykonávania žiadna rola identifikovaná v

procesoch.

Obyčajne sa diagramy prípadov užitia nepoužívajú ako primárna forma návrhového modelu, podľa ktorého sa programuje, ale pre jednoduchosť spracovávaného projektu a urýchlenie vývoja by mali stanovené modely postačovať. Zachytávajú dostatočne požadovanú dynamickú štruktúru budovaného systému. Implementačné detaily sú ponechané na programátorovi a jeho zručnosti. V prípade výskytu komplikovanejšieho prípadu užitia, pre ktorý by popis pomocou krokov z BPMN diagramu nestačil, môže byť pre objasnenie funkcie použitý sekvenčný diagram.

Nasleduje časť návrhovej fázy je zameraná na zachytenie statickej štruktúry systému. Pre grafické zachytenie tohoto pohľadu využijeme diagram tried. Návrhár pomocou tohoto diagramu rozčleňuje budúcu funkcionalitu do logických zoskupení a sprehľadňuje štruktúru kódu. Môže nim nastoliť šablónu pre programátora, vyznačením premenných a metód, ktoré budú v triede použité. Pri návrhu môžu byť použité návrhové vzory, čím sa zvyšujeme robustnosť riešenia.

Míľnikom pre ukončenie modelovania dynamickej štruktúry systému sú diagramy, ktoré zachytávajú statickú a dynamickú štruktúru systému. Dynamická je určená pomocou prípadov užitia, ktoré sú popísané pomocou krokov z firemných procesov a môžu byť rozšírené o sekvenčné diagramy. Statickú štruktúru určuje diagram tried.

4.4.4 Vývoj a testovanie

Vo vývojovej fáze sa vyššie uvedené modely prevádzajú kódovaním do spustiteľnej podoby. Programátori majú pomerne veľkú voľnosť pri realizovaní najnižšej úrovne dodaných návrhov, ale rastie aj ich zodpovednosť na úspešnosti riešenia. Je preto dobré aby programátori boli skúsený a schopný riešiť prichádzajúce implementačné výzvy. Prácou v tíme a komunikáciou medzi kolegami však môže byť ľahko doplnená neskúsenosť malého počtu programátorov.

????? spolupráca pre lepsie riesenie???

Míľnikom pre ukončenie fáze vývoje je ucelená spustiteľná časť budúceho systému.

Fáza testovania začína po dodaní prvej iterácie systému. Systém je podrobne testovaný, či spĺňa požadovanú funkcionalitu. Testery vykonávajú širokú paletu testov od manuáleneho testovania funkcionality systému až po automatizované testy. Testovaná časť produktu musí adekvátne reagovať na všetky správne aj nesprávne vstupy.

???Odhalenie chyby preskocenie nasadenia dalsia iterazia ak sa chyba nenajde pokracuje sa v nasadeni casti systemu ????

Míľnikom pre fázu testovania je správa o výsledkoch testov.

4.4.5 Nasadenie a odovzdanie produktu

Otestovaný produkt je nasadený do prevádzky a zákazník sa s nim môže oboznamovať a osobne ho vyskúšať. Pri výskyte nespokojnosti s obdržanou častou systému, môže vyjadriť svoju nespokojnosť a prípadné nové potreby. Tieto dodatočné požiadavky sú spracované a postúpené do fázy modelovania na zakomponovanie do požiadavkov na systém. Po prebe-

hnutí nasadenia systému pokračuje projekt ďalšou iteráciou.

Po nasadení kompletného systému a jeho odobrení zo strany zákazníka je projekt odovzdaný a ukončený. Dodatočne odhalené chyby a nové požiadavky od zákazníka, ktoré sa prejavia až pri dlhodobom používaní produktu sú riešené pomocou otvorenia nového projektu. Novy projekt sa bude riadiť popisovanou metódou a požadovanú funkcionalitu do systému doplní.

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

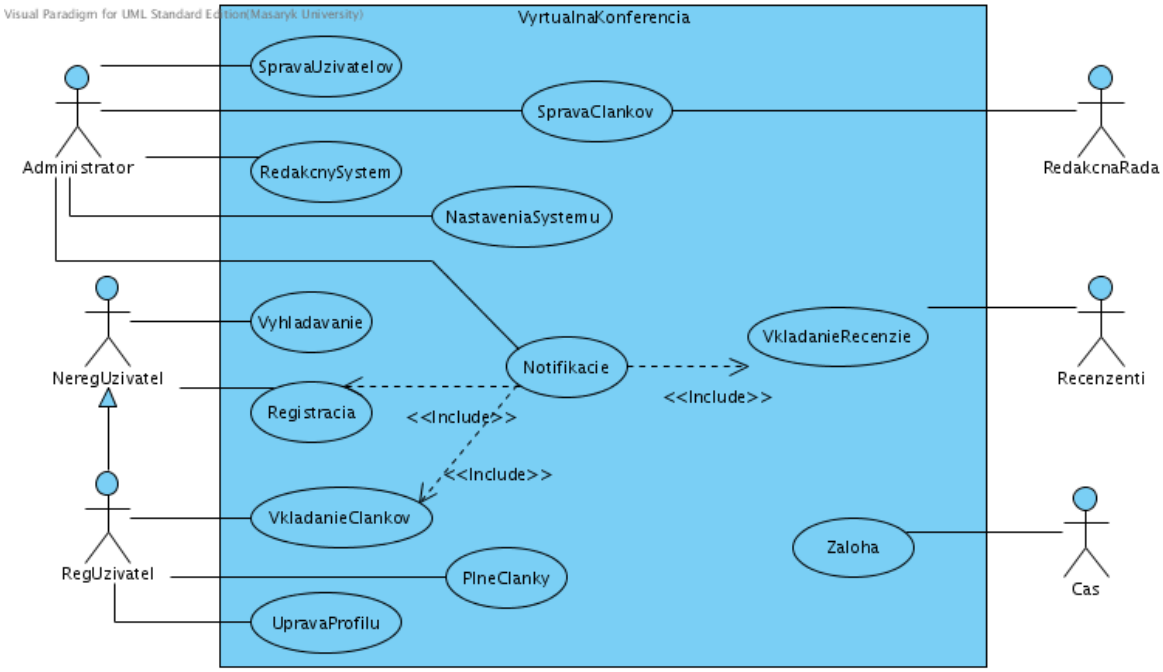
Informácie o uložených článkoch sa budu zobrazovať do prehľadného výpisu obsahujúceho základné informácie. Záznamy sa budú zobrazovať pre aktuálny rok. Staršie ročníky budú uložené v archíve.

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