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

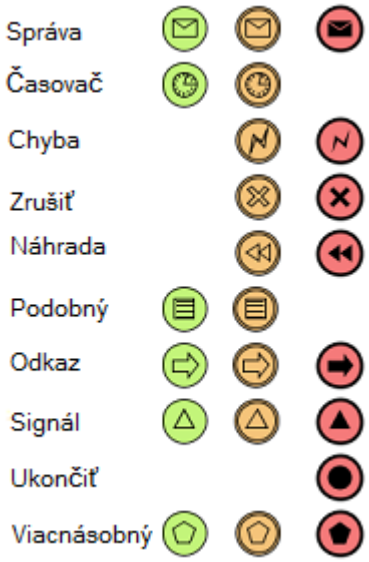
Tokové objekty sú základné objekty BPD udávajú správanie firemného procesu. Definované sú tri Flow Objects:

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.

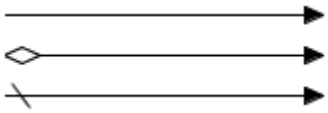
Brána (Gateway)

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)



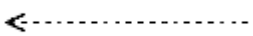
Sekvenční tok reprezentuje neprerušovaná čiara s vyplnenou šípkou. Vyznačuje poradie v akom sú aktivity vykonávané v procese.

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)



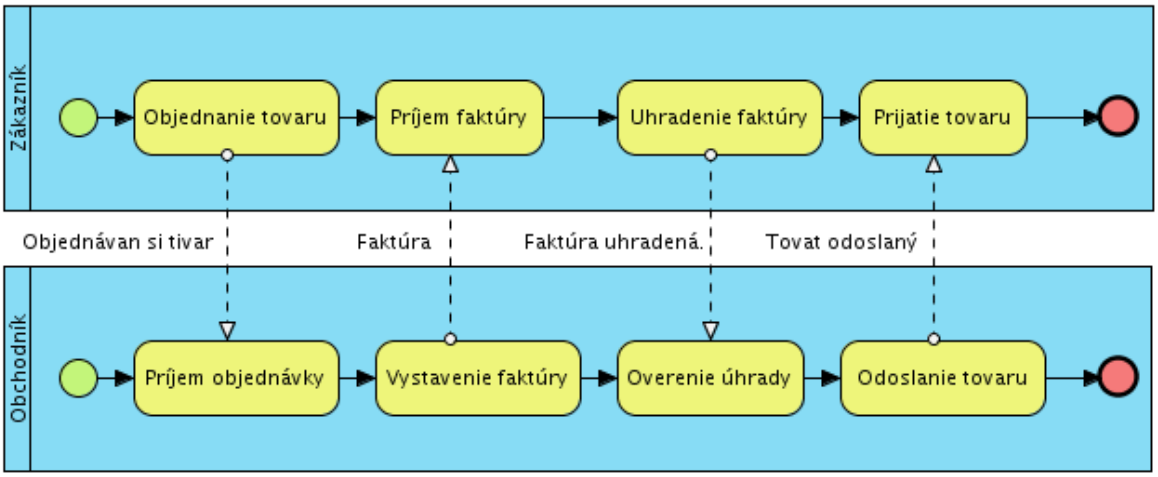
Na znázornenie asociácie sa používa prerušovaná čiara. Pomocou nej priraďujeme k tokovým objektom textové popisky alebo iné objekty ktoré nepatria do skupiny tokových objektov. Pridaním šípky k prerušovanej čiare môžeme určiť smer priradenia.

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

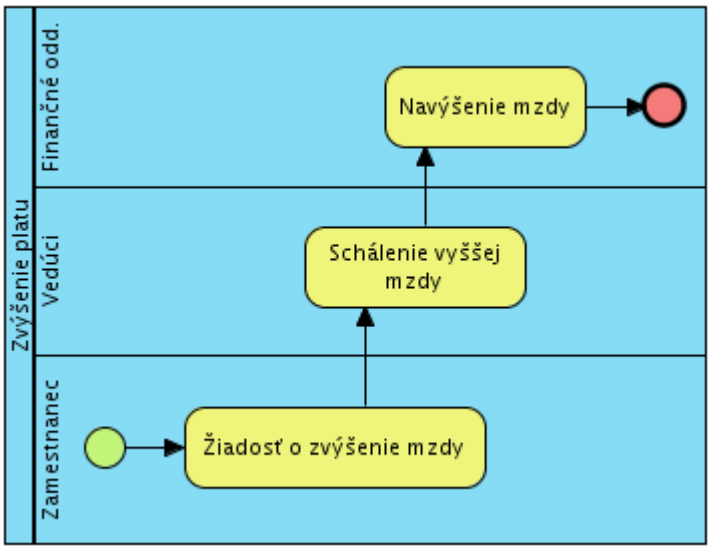
Pool ohraničuje proces a graficky vymedzuje jeho hranice. V rámci jedného poolu sa nachádza len jedne proces. Interakcia medzi poolmi prebieha pomocou správ. Poolý sa v diagrame používajú pre zachytenie dvoch separátnych firemných entít alebo účastníkov. Proces každého účastníka je uzavretý v jeho poole 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.

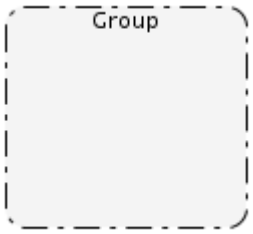
2.1.5 Artefakty (Artifacts)

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)

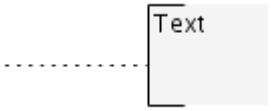
Dátový objekt slúži na zobrazenie toku dat v procese. Pomocou neho modelujeme aké dáta sú požadované a aké dáta systém produkujeme. Dátový objekt je k aktivitám pripájaný pomocou asociácie. Graficky je reprezentovaný obdĺžnikom s ohnutým rohom.

Skupina (Group)



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

Poznámka (Annotation)



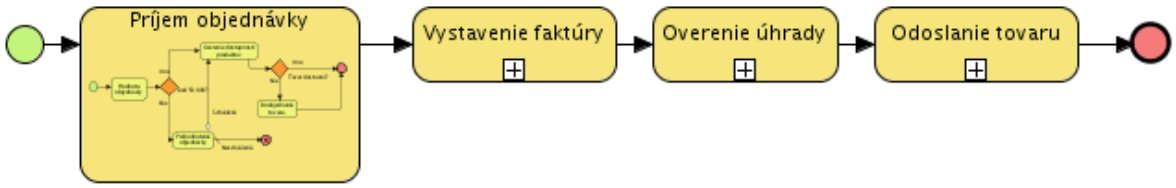
Anotácia slúži na zachytenie dodatočnej textovej informácie pre čitateľa diagramu. K objektu je pripojená pomocou asociácie.

2.1.6 Využitie BPMN

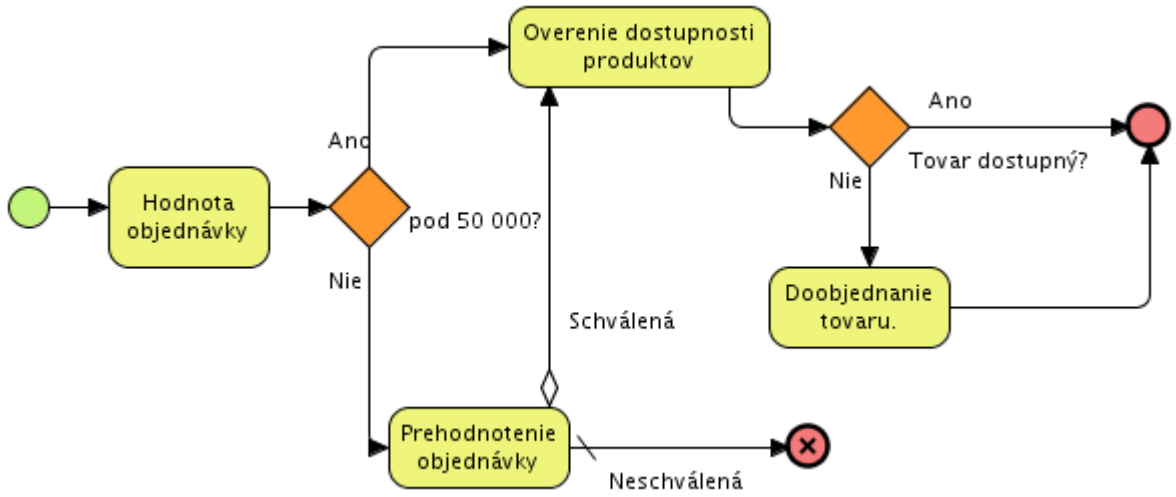
Pomocou BPMN sme schopný modelovať procesy na rôznych úrovniach. Od detailného popisu jednotlivých čiastočných procesov ku globálnej orchestrácii firemných procesov, ktoré sa javia ako čierna skrinka. BPMN tým oslovuje rôznorodé publikum, ktorému predáva široké spektrum informácii na rozličných úrovniach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

Kooperatívne medzi-firemné procesy Kooperatívny typ diagramu zachytáva medzi-firemne 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-firemneho 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

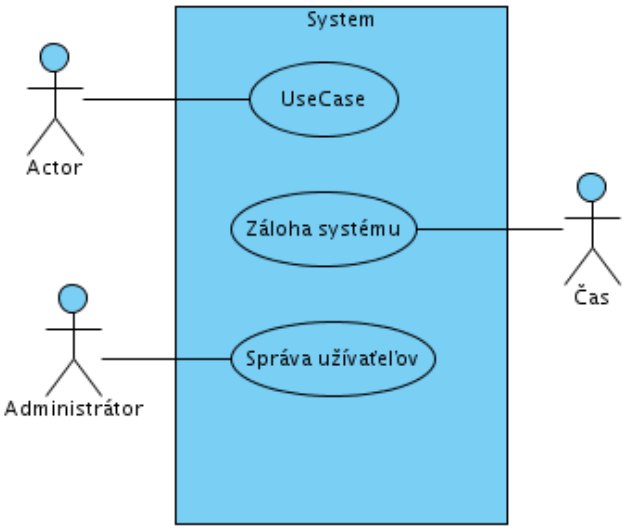
Diagram prípadov užitia (Use Case Diagram) slúži na zachytenie základných funkčných požiadavkov, ktoré ma systém splňovať. Diagram sa skladá z troch základných častí:

Hranice systému Vymedzujú modelovanú oblasť.

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

Prípad užitia Zachytáva ucelenú funkčnú jednotku systému.

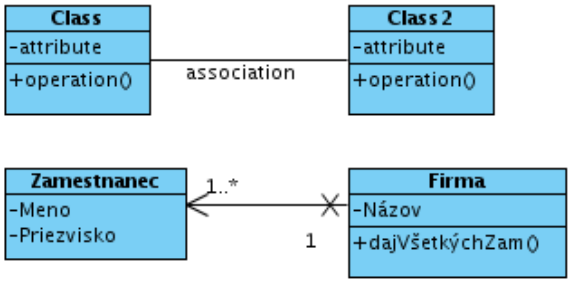
Pomocou týchto častí modelujeme interakciu okolitého sveta s modelovaným systémom. Diagram nesmie obsahovať interakciu medzi aktérmi a ani medzi prípadmi užitia. Súčasťou diagramu je aj detailná dokumentácia jednotlivých prípadov užitia. Na obrázku 2.6 je jednoduchý príklad diagramu.



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

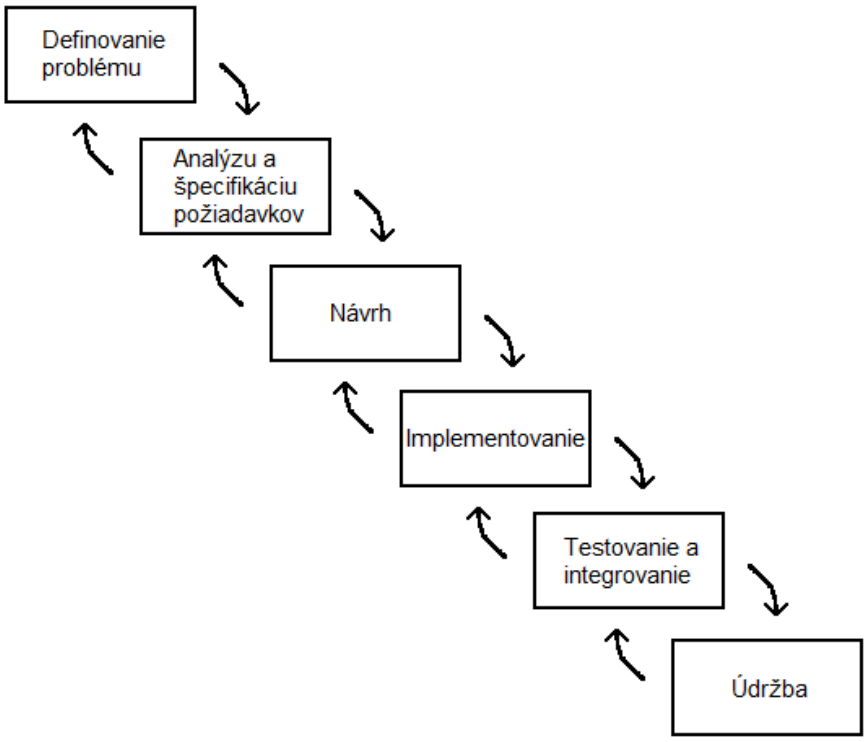
Vodopád patrí medzi najtradičnejšie modely vývoja. Historicky je to prvá ucelené metodika pre vývoj softvéru. Definuje jasné postupy pri zvládaní projektu počas celého jeho životného cyklu. Rozdeľuje projekt na základe vykonávaných aktivít na:

- 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.

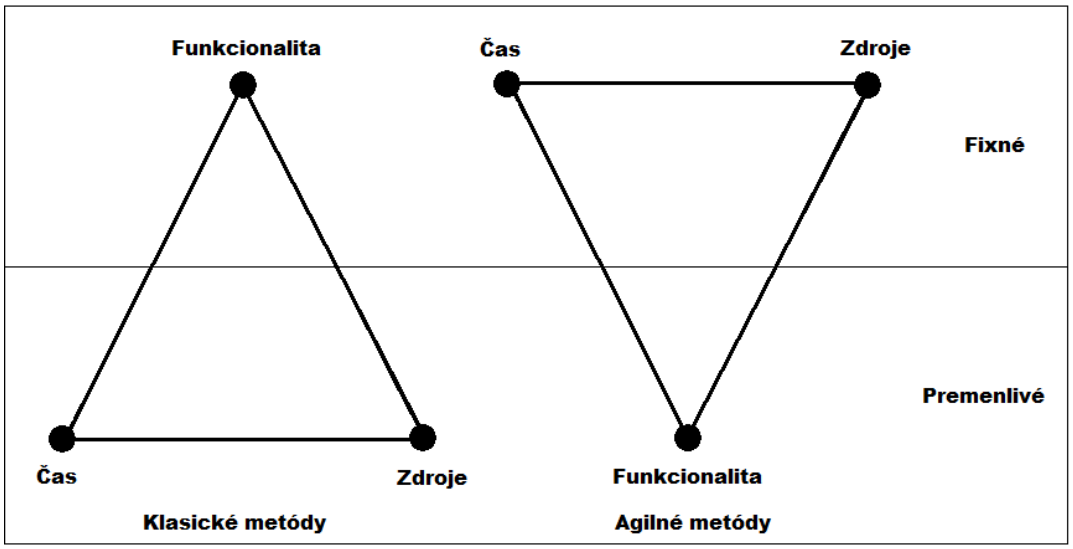
Pri iteratívnom vývoji sa postupuje cestou zdokonaľovania, rozširovania a opravovania už existujúceho systému. Nemalá časť kódu je ďalšími iteráciami pripisovaná prípadne umazaná a nahradená. Preferuje sa prepísanie zlého kódu namiesto jeho obchádzania. Tento postup sa v drvivej väčšine prípadov spája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

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 Unified Process

3.5 Business Driven Development

Kapitola 4

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>

Dodatok A

Príloha A