FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

ZADÁNÍ DIPLOMOVÉ PRÁCE

Název:	Optimalizace vývoje aplikací na platformě IBM BPM
Student:	Bc. Ivan Prokipčák
Vedoucí:	Ing. Pavel Náplava
Studijní program:	Informatika
Studijní obor:	Webové a softwarové inženýrství
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	Do konce zimního semestru 2019/20

Pokyny pro vypracování

Seznamte se s platformou pro vývoj aplikací IBM BPM, analyzujte nejčastější problémy, které při vývoji vznikají a navrhněte a ověřte sadu doporučení, které tyto problémy a jejich dopady minimalizují. Postupujte následovně:
- představte obecný koncept funkčnosti a vývoje aplikací na obecné platformě BPMS,
- omezte se dále jen na platformu IBM BPM a shrňte klíčové vlastnosti této platformy,
- proveďte průzkum nejčastějších problémů, vzniklých při vývoji a jejich praktických dopadů,
- navrhněte sadu doporučení (best practices), které tyto problémy a jejich dopady minimalizují,
- doporučení ověřte na konkrétní praktické implementaci (po dohodě s vedoucím práce),
- v závěru práce proveďte, i s ohledem na provedený průzkum, obecné dopady Vámi vytvořené sady doporučení a postupy jak pro údržbu, tak další rozvoj těchto doporučení.

Seznam odborné literatury

Dodá vedoucí práce.

Ing. Michal Valenta, Ph.D.
vedoucí katedry

doc. RNDr. Ing. Marcel Jiřina, Ph.D.
děkan

V Praze dne 12. března 2018

České vysoké učení technické v Praze
Fakulta informačních technologií
Katedra softwarového inženýrství

Diplomová práca

Optimalizace vývoje aplikací na platformě IBM BPM

Bc. Ivan Prokipčák

Vedúci práce: Ing. Pavel Náplava

9. mája 2018

Poďakovanie

V prvom rade by som chcel poďakovať môjmu vedúcemu práce Ing. Pavlovi Náplavovi, za jeho odborné vedenie a cenné rady v oblasti koncepcie, Bc. Tomášovi Malinkovičovi za konzultácie a mentoring po technickej stránke a Mgr. Miroslavovi Dzurenkovi za praktické informácie v oblasti BPM. Veľká vďaka patrí aj mojim rodičom a najbližším, ktorí pri mne stáli a podporovali ma počas celej doby štúdia.

Prehlásenie

Prehlasujem, že som predloženú prácu vypracoval(a) samostatne a že som uviedol(uviedla) všetky informačné zdroje v súlade s Metodickým pokynom o etickej príprave vysokoškolských záverečných prác.

Beriem na vedomie, že sa na moju prácu vzťahujú práva a povinnosti vyplývajúce zo zákona č. 121/2000 Sb., autorského zákona, v znení neskorších predpisov, a skutočnosť, že České vysoké učení technické v Praze má právo na uzavrenie licenčnej zmluvy o použití tejto práce ako školského diela podľa § 60 odst. 1 autorského zákona.

V Prahe 9. mája 2018 .....

Odkaz na túto prácu

Prokipčák, Ivan. *Optimalizace vývoje aplikací na platformě IBM BPM.* Diplomová práca. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2018.

Abstrakt

Diplomová práca sa zaoberá optimalizáciou webových aplikácií na platforme IBM Business Process Manager, kde je na základe štatistiky najčastejších problémov formulovaná sada najlepších postupov, ktorej správnosť a univerzálna využiteľnosť je prakticky overená na zvolenom systéme.

Klíčová slova Business Process Manager, BPMN, najlepšie postupy, optimalizácia, webová aplikácia

Abstract

The aim of this diploma thesis is optimization of web applications on the IBM Business Process Manager platform, where the best pracites are formulated based on the statistics of common bugs and problems, whose correctness and universal usability is practically verified on the selected system.

Keywords Business Process Manager, BPMN, methodology, best practices, optimization, web application

Obsah

Zoznam obrázkov

Zoznam tabuliek

Úvod

Po absolvovaní stáže v spoločnosti IBM Česká Republika, spol s r.o. ako BPM Consultant, v súčasnosti dva roky pracujúci na pozícii BPM Developer, som sa pri vývoji stretol s častým nepochopením či nedodržaním patternov a princípov, ktoré vedú k dobre udržiavateľnému a jednoducho rozšíriteľnému kódu pri vývoji procesných aplikácií.

Hlavným nedostatkom je chýbajúca dokumentácia a informácie zodpovedajúce aktuálnemu stavu BPM technológie. Každý nový vývojár znova naráža na problémy, ktoré už boli riešené, prípadne neprodukuje dostatočne kvalitný kód zodpovedajúci určitým konvenciám. Tento fakt spôsobuje že výsledné projekty sú po implementačnej stránke ťažko udržateľné a každé rozšírenie je čoraz komplikovanejšie. Vplýva to i na celkové fungovanie dodanej aplikácie a práca s takýmto systémom vedie k častej frustrácii zo strany užívateľa.

Na podnety skúsenejších vývojárov a neustále zvyšujúce sa množstvo problémov s rozšíriteľnosťou či udržateľnosťou väčších projektov som sa rozhodol v rámci diplomovej práce definovať sadu pravidiel, ktorá je založená na štatistikách chybovosti naprieč niekoľkými projektami založenými na rovnakej technológii, a jej prínos následne prakticky overiť v laboratórnych podmienkach.

Cieľom je, aby vzniknutá sada doporučení bola uchopiteľné a univerzálne aplikovateľná naprieč celým spektrom projektov postavených na technológií IBM BPM a výsledné produkty boli dobre fungujúce z hľadiska developmentu i využiteľnosti z pohľadu užívateľov.

Štruktúra práce

V prvej kapitole predstavím a podrobne rozoberiem pojem Business Process Management. Ďalšia kapitola sa zaoberá nástrojmi postavenými na tejto platforme a predstavuje ich základné vlastnosti. V tretej kapitole naviažem podrobným rozborom zvoleného BPM nástroja, od jeho architektúry po vývoj.

Následujúca kapitola obsahuje štatistiku najčastejších chýb a zmenových požiadavkov v rámci projektov postavených na zvolenom nástroji. Informácie z tejto kapitoly ďalej použijem ako základ k formulácií pravidiel v kapitole 5. V poslednej, 6. kapitole popisujem testovanie a zhodnotím prínosy i univerzálnosť využitia definovanej sady pravidiel.

Kapitola 1

Úvod do BPM

V tejto kapitole objasním samotný pojem BPM a načrtnem jeho históriu. Ďalej sa budem venovať výhodám a principom technológie samotnej až napokon rozoberiem niektoré typy nástrojov využívajúce BPM.

1.1 Definícia Business Process Managementu

Pojem biznis proces sa stal novým paradigmom produktivity a nahradil tak funkcionálne či procedurálne myslenie v spoločnostiach za procesné. [1] ho definuje ako "kolekciu prepojených úloh, ktoré končia pri dodávke služby alebo produktu klientovi. Podnikový proces je často definovaný ako sada aktivít a úloh, ktoré po dokončení dosiahnu cieľ organizácie. Proces musí obsahovať jasne definované vstupy a jediný výstup."

Biznis process je často vizualizovaný formou diagramov či logických krokov. Obor, ktorý formalizuje túto metódu sa nazýva Business Process Management (BPM).

Podľa Nathaniel Palmer Business Process Management (BPM) je [2] "*disciplína zahŕňajúca ľubovoľnú kombináciu modelovania, automatizácie, realizácie, riadenia, merania a optimalizácie tokov podnikových aktivít, na podporu podnikových cieľov, systémov, zamestnancov, zákazníkov a partnerov v rámci podniku aj mimo jeho hraníc.*"

V skratke, BPM definuje sadu princípov, metód a nástrojov, ktoré umožňujú indentifikovať, navrhnúť, vykonať a monitorovať biznis proces.

BPM je často chybne zamieňaný s pojmom workflow, ktorý sa od neho nepatrne líši. Zach Messler svojou analógiou formou Road Trip tento rozdiel vysvetľuje takto:

"*Ak ste niekedy absolvovali náročnejšiu cestu či výlet naprieč mestami viete, že je veľa vecí, ktoré musíte urobiť. Zaobstarať dorpavný prostriedok, vybrať miesto, naplánovať načasovanie množstva vecí. Krátky výlet z cesty sem a tam? To je presný a priamočiary postup typu - tu sú kroky, a po ich splnení*

sa dostaneš do cieľa. Čo v prípade, že Vašou úlohou je kontrola dopravy? Súčasťou kontroly premávky sú samozrejme všetky jednotlivé automobily, ktoré robia svoje individuálne cestné výlety. Ale čo ak by ste sa nachádzali v hlavnom riadiacom stredisku a sledovali dopravné toky a optimalizovali dopravné modely v reálnom čase. Tiež analyzovali minulé problémy s dopravou, upravovali samotné cesty, aby ste optimalizovali dopravný tok, aby boli cestné výlety príjemnejšie. To je BPM. Koordinuje všetky rôzne procesy, údaje, systémy a ľudí. [3]

1.2 História a vznik Business Process Managementu

Techniky návrhu biznisových procesov siahajú až k začiatkom osemnásteho storočia, kde ich riadením sa začal ako prvý zaoberať Adam Smith (1723-1790). Hlavnou myšlienkou jeho výskumu bolo delenie práce, kde sa každá časť výrobného procesu špecifikuje a venuje len určitej časti namiesto porkytia celého výrobného procesu. Potenciál tohoto systému opísal na príklade výroby kolíkov. [4]

V roku 1911 sa v článku s názvom "Princípy vedeckého manažmentu"začal Taylorizmus, vytvorený Frederickom Taylorom (1856-1915). Tento systém sa zaoberal hodnotou procesov, analýzou a integráciou pracovných postupov s cieľom zlepšiť produktivitu práce. Taylorizmus sa zameral na zlepšenie efektívnosti odstránením nepotrebných krokov a akcií, a nahradil procesy založené na paletových pravidlách s presnými postupmi vyvinutými prostredníctvom metodického preskúmania. V článku [5] sa píše, že počas práce s firmou Bethlehem Steel Taylor zistil, že obmedzenie lopatovej záťaže surovín na 21 1/2 libier prinieslo najefektívnejšie výsledky. Namiesto toho, aby pracovníci museli odhadnúť hmotnosť svojich nákladov, Taylor zadal špeciálne lopaty, ktoré držali presne takú váhu. Hoci frustrovaní manažéri v oceliarni zaistili Taylorovo prepustenie, táto koncepcia odvtedy ovplyvnila všetko od výroby automobilov až po dizajn kuchyne v reštauráciách. [6]

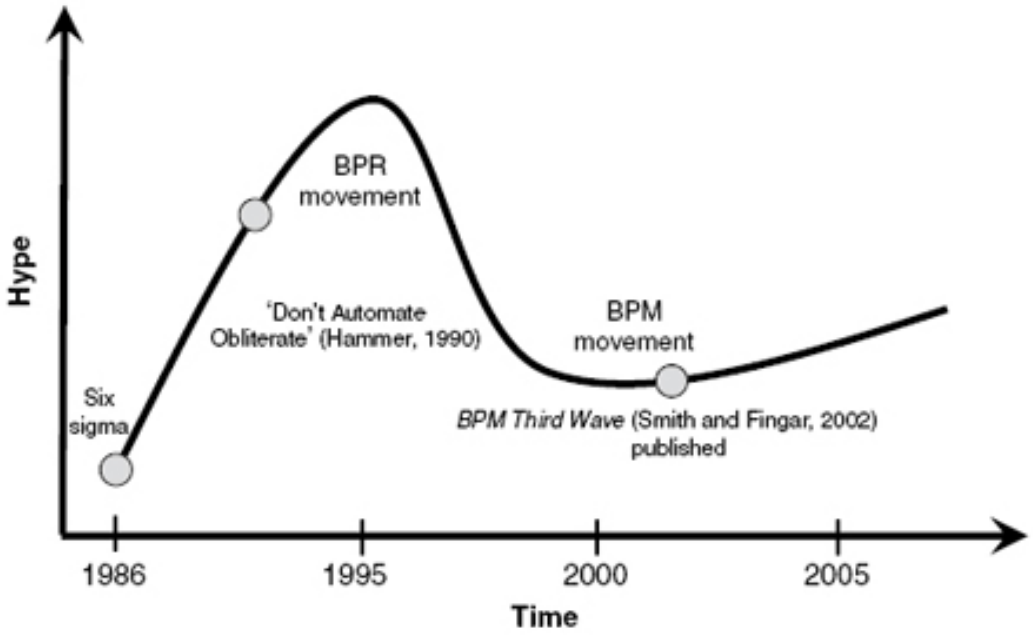
Na začiatku 20. storočia urobil Henry Ford revolúciu výroby, keď nainštaloval prvú montážnu linku do svojej automobilovej továrne v blízkosti Detroitu v štáte Michigan. Koncept montážnej linky však existoval už celé stáročia, no moderná montážna linka vyvrcholila divoko rozdielnymi experimentami a zameniteľnosťou, inováciami a logikou. Bola vytvorená pre zlepšenie výroby a získanie konkurenčnej výhody. [7]

Podobne ako v priemyselnom veku minulého storočia, informačný vek si vyžiadal novú paradigmu podnikateľského procesu pre 21. storočie. Namiesto optimalizácie fyzickej práce potrebnej na výrobu tovaru musia moderné organizácie nájsť spôsob, ako optimalizovať služby poskytované špecialistami. Hoci sa nachádzame v informačnom veku, špecialisti môžu byť stále riadení

pomocou procesov a postupov navrhnutých pred desiatimi rokmi na vytvorenie fyzického produktu, služby alebo transakcie.

V dnešnom náročnom ekonomickom prostredí je BPM dôležitejšie než kedykoľvek predtým. Dnešný pracovný produkt je s najväčšou pravdepodobnosťou digitálny, obsahovo riadený a viacvláknový. Hodnota, efektivita a účinnosť sa merajú skôr v minútach a sekundách než v hodinách a dňoch. Keďže maximalizácia produktivity a efektívne využívanie ľudských a technologických zdrojov sa stávajú čoraz dôležitejšími pre prežitie podnikov. BPM slúži ako koncept, kde prax a súbor podporných technológií ponúka transformačné príležitosti na modernizáciu a automatizáciu procesov. Technológia je kľúčom k tejto transformácii. Rovnako ako technológia umožnila "Taylorizmus"v priemyselnom veku, tak to ako BPM umožňuje pre informačnú dobu dnes.

Peter Drucker opísal bežnú chybu väčšiny iniciatív na optimalizáciu procesov: "Nie je nič také zbytočné, ako efektívne robiť to, čo by sa nemalo robiť vôbec."Jednoducho používať technológiu na automatizáciu alebo urýchlenie procesu, ktorý je zle konštruovaný povedie k rýchlejšej a automatizovanejšej neefektívnosti - čo nie je cieľom BPM. Naopak, BPM založená na informáciách požaduje, aby sa všetky prítomné technológie vylepšili, a aby výsledný proces bol čo najlepšie optimalizovaný.



Obr. 1.1: Vývoj priaznivosti BPM v čase

1.3 Prečo práve BPM

V prvom rade je dôležité si uvedomiť, že BPM "nie je produkt alebo technológia. BPM je komplexný prístup k riadeniu a zlepšovaniu účinku a efektívnosti obchodných procesov". [8]

BPM umožňuje spravovať procesy a podporovať firemnú iniciatívu, ako napríklad zlepšenie kvality výrobkov, skrátenie času uvedenia produktu na trh, zvyšovanie spokojnosti zákazníkov, nárast ziskových marží atď.

Využitie BPM odcenia práve spoločnosti, v ktorých je ich prevádzka orientovaná procesne. To znamená, že každý požiadavok či iniciatíva, má svoj začiatok a koniec, do ktorého sa dostane určitým, jasne daným, postupom. Ten sa nazýva proces. Príkladom je žiadosť o hypotéku v banke, ktorá je zadaná a musí prejsť procesom schvaľovania, pričom za každú časť je zodpovedný niekto iný. V závere je získaná odpoveď o vyhovujúcom či nevyhovujúcom žiadateľovi, prípadne je posudzovaná iným oddelením, čo môže viesť k spusteniu ďalšieho procesu.

V súčasnosti existuje stále veľa podnikov a biznisových procesov, ktoré sú stále spravované manuálne, a dokonca nástrojmi ako sú emaily a tabuľky. Je teda evidentné, že neprítomnosť BPM, nemusí nutne znamenať problém, no riziko je určite väčšie.

Medzi najčastejšie problémy, ktoré v sebe nesú manuálne riadené procesy sú:

Viditeľnosť a kontrola Viditeľnosť a kontrola procesov sa stáva náročnou, ak nie nemožnou, ak je spustený a riadený manuálne, alebo spadá do viacerých podnikov, systémov či oddelení. Jednou z možností je pokúsiť sa manuálne zachytiť potrebné údaje ako sú napríklad doby spracovania. Je to však náchylné k chybám. Z operačného hľadiska sa pre vlastníkov procesov a ľudí pracujúcich s procesom stáva náročným získať prehľad o tom, ako to robia. Napríklad z pohľadu manažéra tímu sa stáva ťažké získať prehľad o tom, na čom tím pracuje, na kom je konkrétna úloha alebo aké celkové množstvo práce musí vykonať. To môže mať veľký vplyv na spokojnosť klientov.

Zmeny procesov Pri manuálnych procesoch sa spoliehate na to, aby používateľ vedel čo má robiť a aký je ďalší krok procesu. V prípade zmien sa teda očakáva a spolieha na správne rozhodnutie užívateľa. Tu ale nastáva riziko chyby, čo spôsobuje, že je náročné proces zlepšovať a merať, pretože tu nie je istota, že sú porovnávané totožné vetvy.

Chýbajúce údaje Jedným z najväčších problémov manuálnych procesov je migrácia, ktorá je nákladná ako časovo tak i finančne. Migrácia manuálnych procesov na inú platformu či ich digitalizácia znamená opätovné zadávanie údajov, kde sa chybovosť v porovnaní s automatickou migráciou náramne zvyšuje.

Strata práce Pri manuálnych procesoch vždy existuje riziko staty či nedoručenia informácie komu je určená, výnimočne ak sa spoliehate na emaily, ktoré môžu byť ľahko prehliadnuteľné či zmazané. Táto situácia môže spôsobiť nesplnenie úlohy včas čo má znova negatívny vplyv na klienta.

BPM v tomto smere predstavuje vrstvu abstrakcie medzi používateľom a back-end systémami. Proces interaguje so systémom back-end, zvyčajne formou zbernicovej služby (ESB) a komunikuje s používateľmi prostredníctvom zabudovaných UI. Preto má používateľ jedno rozhranie, ktoré mu prezentuje informácie, naprieč všetkými systémami, potrebné k dokončeniu úlohy. Proces je modelovaný a vykonaný v rámci nástroja BPM. Prínosom je, že va vždy vykonáva ten istý proces, čo minimalizuje riziko. Taktiež sa procesy stávajú predvídateľnými, opakujúcimi a ľahko merateľnými. Proces spracováva aj smerovanie úloh, aby sa ubezpečil, že úlohu dostane vždy správna osoba. BPM tiež poskytuje plnú kontrolu a prehľadnosť v rámci priebehu celého procesu od jeho začiatku, ukončenie a archiváciu.

1.3.1 Životný cyklus zavedenia BPM

Zavádzanie BPM prebieha v niekoľkých fázach súhrnne zvaných ako "životný cyklus". Existuje viacero verzií životného cyklu BPM, väčšinou sa líšia v počte krokov. V článku "Pozícia spoločnosti Gartner na BPM"(2006) opísal "cyklus revízie procesov"nie menej ako 9 krokov. [9] Podľa Nakul Bharade počet krokov však nie je relevantný. Mnohé organizácie výrazne zvýšili svoju produktivitu a efektivitu už tým, že uplatnili pomerne jednoduchú verziu cyklu BPM. Táto verzia pozostáva zo šiestich krokov, ktoré sa dajú použiť o to efektívnejšie. [10]

1.3.1.1 Dizajn

V biznise všetko začína plánom. Tento plán v sebe obsahuje pravidlá, ktoré musia dodržiavať tí čo ho budú vykonávať. Aby bola stratégia BPM užítočná, musí byť procesne orientovaná a jej plán musí byť navrhnutý a štrukturovaný spôsobom, ktorý zabezpečí dodanie hodnoty zákazníkovi. Plánovanie procesov a stratégie v sebe zahŕňa pochopenie cieľov a fungovania organizácie. Ďalším krokom je identifikácia a vyčíslenie súčasných procesov, ktoré vyžadujú podrobný pohľad na existujúcu procesnú architektúru organizácie. Podľa typu procesov alebo činností, ktoré existujú v organizáciách ich môžeme rozdeliť na tieto tri:

1. Primárne procesy Ide o základné procesy spoločnosti, ktoré sú ľahko identifikovateľné z dôvodu ich vzájomnej funkčnej povahy a skutočnosti, že ide o procesy, ktoré priamo prinášajú zákazníkom hodnotu. Tieto procesy vychádzajú z hlavných aktivít podnikania. [11] ich kategorizoval na:

- procesy, ktoré vytvárajú produkt alebo službu (napr. vývoj produktu)

- procesy, ktoré produkujú výrobok alebo službu (napr. výrobný proces)
- procesy, ktoré predávajú či dodávajú produkt alebo službu zákazníkom (napr. distribúcia)

2. sekundárne procesy Tento typ procesov je zámerne určený na poskytovanie podpory primárnym procesom, a preto sú tiež označované ako podporné. Na rozdiel od primárnych neprinášajú priamo hodnotu, ale zvyčajne sa obmedzujú na funkčné oblasti organizácie. Príkladmi takýchto procesov sú riadenie ľudských zdrojov, obstarávanie zdrojov, riadenie zariadení či vývoj technológií.

3. Riadiace procesy Cieľom všetkých organizácií je dosiahnuť čo najväčšiu účinnosť a efektivitu, kde dôležitú úlohu zohráva manažment, ktorý je priamo zodpovedný za vykonávanie riadiacich procesov. Tieto procesy pracujú s primárnymi i sekundárnymi procesmi, kde sledujú či sú na správnej ceste pri plnení prevádzkových a finančných cieľov spoločnosti. Zabezpečujú tiež súlad primárnych a sekundárnych procesov v závislosti na regulačných a právnych predpisoch.

Celkový dôraz je vedený na účinnosť, efektívnosť a úlohu monitorovania, ktorá spočíva na ramenách manažmentu, ktorí sú zodpovední za vykonávanie riadiacich procesov. Tieto procesy sa pozerajú na primárne aj sekundárne procesy, hlavne na sledovanie toho, či sú na správnej ceste pri plnení prevádzkových a finančných cieľov spoločnosti. Sú tiež zavedené na zabezpečenie súladu primárneho a sekundárneho procesu s regulačnými a právnymi predpismi.

Podobne ako podporné procesy, priamo neposkytujú hodnotu, no sú pre organizáciu mimoriadne dôležité, pretože identifikujú a definujú úlohy a povinnosti BPM v rámci organizácie.

1.3.1.2 Analýza

V analytickej fáze sa uplatňujú určité metodiky. Najbežnejším a zvyčajne prvým krokom je zhromažďovanie údajov a informácií o obchodných procesoch. Tieto sú získané z aktuálne udržiavanej firemnej alebo organizačnej dokumentácie, strategických plánov a modelov procesov. Analýza poskytuje prehľad o silných a slabých stránkach obchodných procesov a prispieva k pochopeniu toho, ako ovplyvňujú celkovú výkonnosť organizácie.

Techniky analýzy procesov môžeme rozdeliť na tieto dve:

1. Kvalitatívna analýza - jej hlavným prínosom je identifikovať redundantné časti alebo vzniknuté straty v procesoch a eliminovať ich. Najčastejšie z používaných techník sú:

- **Analýza s pridanou hodnotou** - v tejto technike sa každý krok v procese klasifikuje na to, či ide o pridanú hodnotu (VA), pridanú hodnotu

v rámci biznisu (BVA) alebo nepridanú hodnotu (NVA). [12] Krok sa považuje za VA, ak prináša zákazníkovi úžitok, označovaný ochotou zákazníka zaplatiť za tento konkrétny krok. Príkladom je proces montáže produktu. Po dokončení výrobného procesu, je pred dodaním zákazníkovi konečný výrobok povinný podľa zákona podstúpiť fyzickú kontrolu nezávislou treťou stranou. To zákazníkovi neprináša žiadnu pridanú hodnotu, ale dodáva hodnotu podniku - organizácii, pretože je to požiadavka na nepretržitú prevádzku podniku. Toto je aktivita BVA. Náklady na opravu v prípade, že výrobok bol chybný z dôvodu nedbanlivosti pracovníkov, ako aj náklady na oneskorenie dodávky neprispievajú k zákazníkovi, teda ide o NVA. V konečnom dôsledku všetký činnosti, ktoré nespadajú do kategórie VA ani BVA, spadajú medzi NVA.

- **Pareto analýza** - názov získala podľa typu použitých grafov (Pareto stĺpcové grafy). Grafy znázorňujú vplyv všetkých relevantných problémov alebo problémov. Vyššie stĺpce sú tie, ktoré vyžadujú väčšiu pozornosť, čo znamená, že by mali byť uprednostňované.

2. Kvantitatívna analýza - jej techniky sa sústreďujú na čísla a štatistiky. Medzi najčastejšie patria:

- **Kvantitatívna analýza toku** - zvyčajne sa používa pri analýze požiadaviek na kapacitu, miery chybovosti na úrovni procesov a nákladov.
- **Časová analýza cyklu** - je bežnou aplikáciou analýzy toku, pri ktorej sa vypočítava priemerná dĺžka alebo čas cyklu pre celý proces (krok v procese) s cieľom posúdiť účinnosť a efektivitu.
- **Analýza fronty** - na rozdiel od analýz tokov, zohľadňuje čakaciu dobu. Je zohľadnená variabilita prevádzkových časov, oneskorenia a prepracovania času, keďže sa uznáva skutočnosť, že tieto kapacitné problémy sú nevyhnutné, a preto by mali zohrávať úlohu pri navrhovaní a rekonštrukcii procesov.
- **Simulácia procesov** - je uplatniteľná v procesoch, ktoré zahŕňajú paralelné činnosti bežiace súčasne. Riadenie sa všeobecne nevzťahuje na tieto typy aktivít. Pri simulácii procesov je proces modelovaný do simulačného modelu, ktorý je vylepšený simulačnými údajmi. Získané výstupy sa následne analyzujú. V prípade, že existuje viacero scenárov, simulačný model môže byť upravený zodpovedajúcim spôsobom a podrobený opätovnému simulovaniu, kým nie sú pokryté všetky alternatívne scenáre.

Závery získané v tehto fáze poukazujú na fakt či obchodné procesy sú alebo nie sú zosúladené s plánmi a cieľami organizácie.

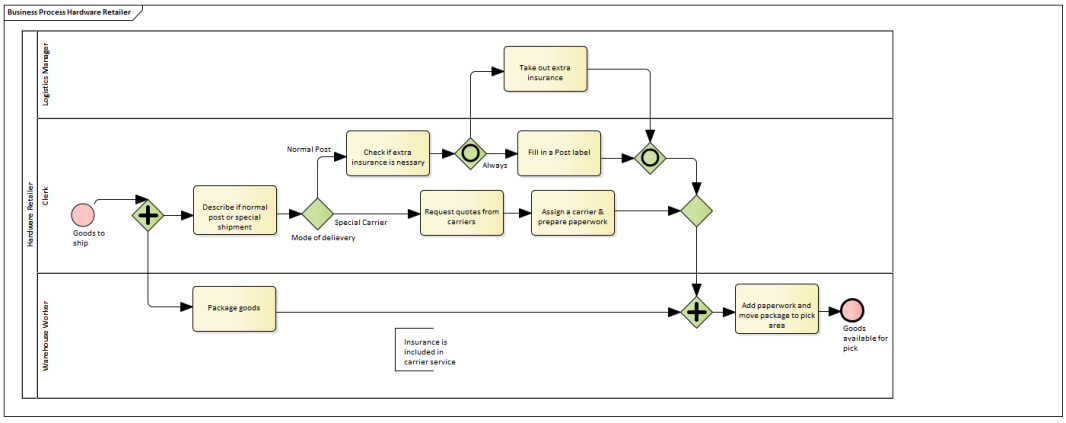
1.3.1.3 Model

V závislosti na výsledkoch analýzy môže byť potrebné navrhnúť procesy nové alebo prepracovať existujúce. Cieľom tejto fáze je teda získať návrh procesu, ktorý poskytne celkový obraz práce od začiatku až do konca a pomôže určiť či je proces dobrý tak ako je (as is) alebo má byť redizajnovaný do viac prijateľnej podoby (to be).

Fáza sa skladá z týchto častí:

- pochopenie zámeru organizácie v rámci obchodného procesu, teda toho čo chce dosiahnuť a ako použije tento proces na splnenie cieľa
- dokumentácia práce, ktorá sa má vykonať počas modelovania procesu (napr. povaha práce, čas a frekvencia výkonu práce)
- identifikácia a pochopenie environmentálnych faktorov, ktoré majú vplyv na proces, pretože pravdepodobne ovplyvnia dizajn, príp. redizajn procesov

Model obchodných procesov je aktivita zastupujúca procesy organizácie v štruktúrovanej podobe. Jej hlavným účelom je dokumentácia a analýza. Model procesov slúži aj ako stavebný kameň pre akékoľvek budúce projekty alebo procesné hodnotenia. Užitočným je aj pri vzdelávaní nových zamestnancov. Modely bývajú prezentované aj vo forme popisného príbehu, no najčastejšie pomocou diagramov či ilustrácií s popisom či ide o primárny, sekundárny alebo riadiaci proces.



Obr. 1.2: Proces zachytený pomocou BPMN notácie

V prípade, že ide o redizajn procesov, existujú dva prístupy a to zlepšovanie priebežne či celkový redizajn. Pri priebežnom zlepšovaní procesov dochádza

k úplnému prijatiu súčasných procesov. Preto je hlavným cieľom sústrediť sa na identifikáciu problémov alebo chýb, ktoré boli predtým nepovšimnuté, a hľadať ich riešenia. Tento prístup hľadá riešenia inkrementálne, pričom v rámci jedného cyklu súčasne rieši iba jeden problém. V rámci druhého prístupu na rozdiel od kontinuálneho zlepšovania procesov, dizajn sa pozerá na celú procesnú štruktúru a jeho cieľom je úplne prepracovať súčasný stav, s cieľom zlepšenia efektivity a účinnosti. Tento prístup je omnoho komplexnejší.

1.3.1.4 Realizácia

Potom ako je proces navrhnutý, je čas uviesť ho do praxe. Tento stav je dosiahnuteľný akýmkoľvek riešením BPM. Dôležitým aspektom je, že vždy existuje vzťah 1:1 medzi tým, čo modelujeme a čo implementujeme, vykonávame. Novo vytvorený či redizajnovaný proces sa stané novou äs is"verziou.

Spustiteľnú verziu modelovaného procesu, sa vytvára využitím nástrojov BPMS, či iných workflow nástrojov a SOA technológií. V závislosti na použitom nástroji je možné automatizáciu dosiahnuť tromi spôsobmi:

1. **Priame spustenie modelovaného procesu** Najnovšia verzia BPMN (2.0) je navrhnutá tak, že umožňuje priame spustenie modelu procesu, ktoré je reprezentované štrukturovaným spôsobom (napr. založenom na XML). Hlavným benefitom sú oveľa kratšie vývojové cykly, pretože nepotrebuje žiadne ďalšie preklady modelu či programovania. Tento model procesného diagramu zároveň poskytuje priamy náhľad procesu pre zainteresované strany klienta.

2. **Priame spustenie modelovaného procesu** Preklad modelu do spustiteľného procesu. Ide o modely navrhnuté vo verziách BPMN 1.2 a nižšie, ktoré neumožňujú ich spustenie. Tie musia byť preložené do špeciálnych jazykov, ktoré túto možnosť poskytujú, napr. BPEL2.

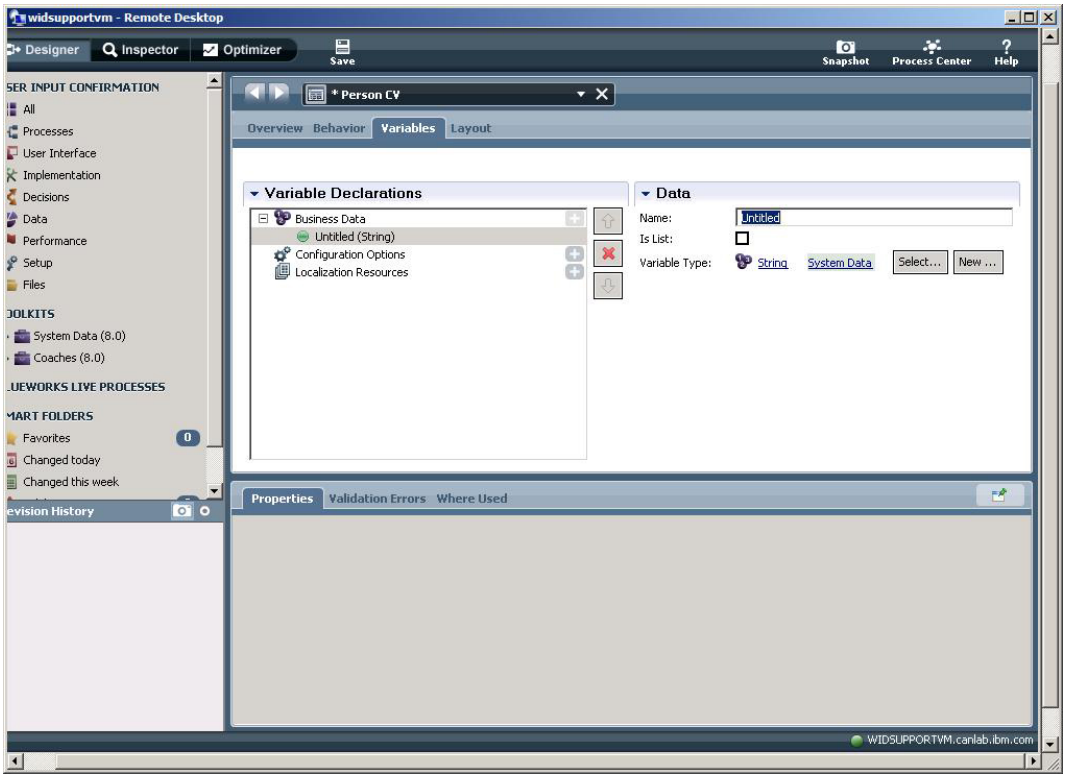
3. **Vlastná implementácia spustiteľného procesu** Pri tejto alternatíve sa proces vyvíja od začiatku na základe modelu zachyteného počas fáze modelovania. Implementácia sa zvyčajne realizuje v programovacom jazyku bežne používanom pre podnikové aplikácie ako sú Java alebo C#. Vynaložené úsilie na samotný vývoj a súvisiace činnosti softvérového inžinierstva sú pri tomto spôsobe podstatne vyššie ako pri predchádzajúcich dvoch možnostiach.

Neoddeliteľnou súčasťou implementácie procesu je aj jej riadenie, aby sa transformácia vykonala správne, a preto si vyžaduje jasnú komunikáciu so všetkými zainteresovanými stranami procesu, napr. vysvetlenie dôvodu transformácie a čo sa od nej očakáva.

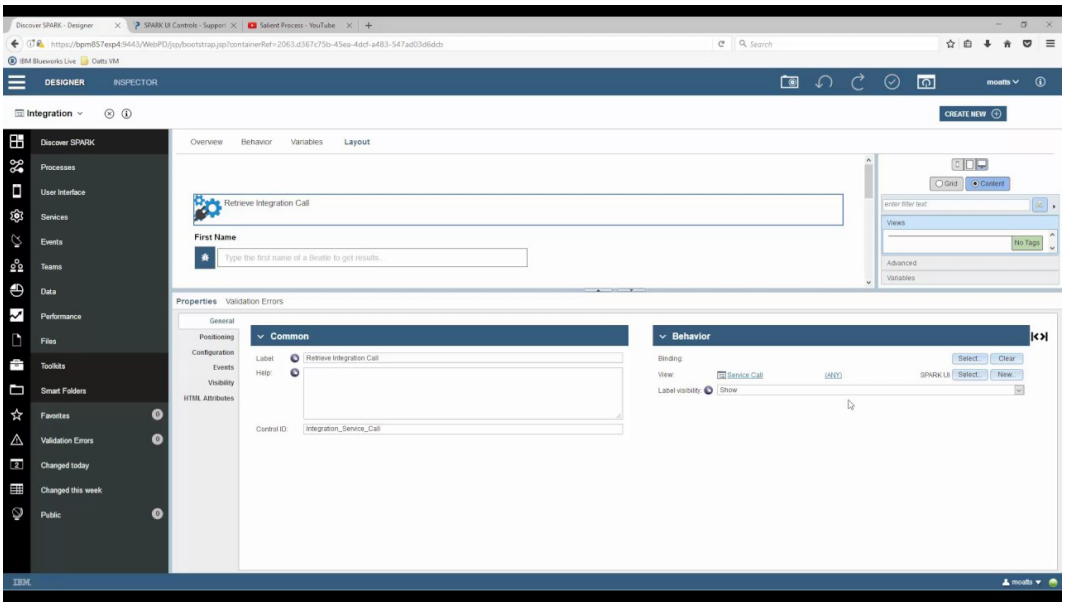
[13] píše o ďalších krokoch, ktoré je nevyhnutné vykonať a líšia sa v závislosti od zvoleného BPM nástroja. Sú to integrácia, užívateľské rozhrania a správa užívateľov.

Obchodný proces v spustiteľnej forme musí byť obohatený o technické detaily umožňujúce integráciu procesu do podnikovej architektúry. To zahŕňa vytvorenie alebo konfiguráciu adaptérov existujúcich systémov, databáz, portálových riešení, alebo ERP. Podstatná middleware technológia zohráva dôležitú úlohu v integrácii. Systém BPM môže profitovať z koexistencie technológií SOA, ako sú sprostredkovatelia správ alebo ESB.

Užívateľské rozhranie (UI) pre procesné ľudské úlohy je potrebné rozvíjať, aby sa zabezpečila interakcia medzi účastníkmi systému. UI dizajn a implementácia je buď na zákazku vytvorená (extenalizovaná) na tradičnom webe formou JSF alebo automaticky generované používateľským rozhraním technológiu zabudovanú v balíku BPM. Niektoré produkty BPM ponúkajú mnoho možností prispôsobenia pre budovanie moderných bohatých webových rozhraní integrované do podnikateľského prostredia (viď príklad nástroj na návrh používateľského rozhrania na obrázku).



Obr. 1.3: BPM uživáteľské rozhranie Proces Designera, desktopová verzia



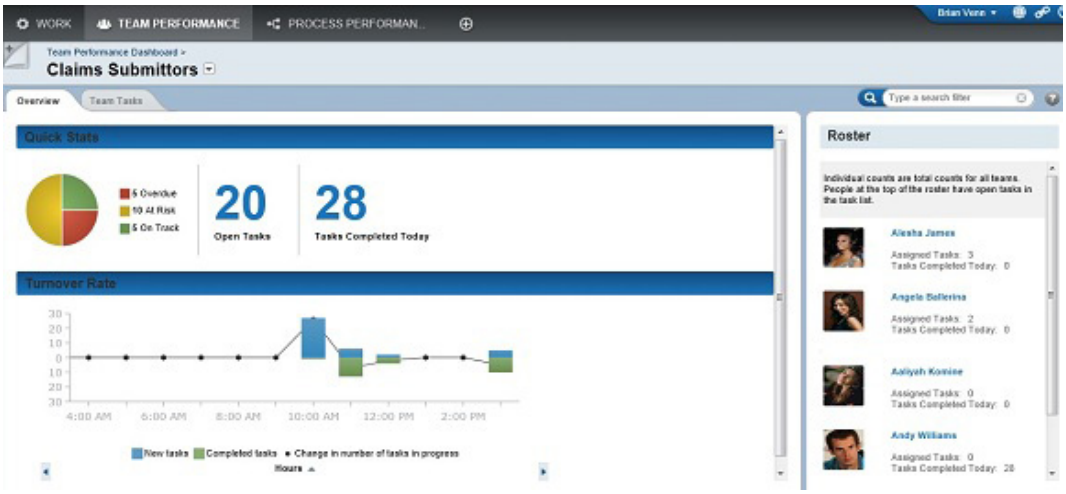
Obr. 1.4: BPM uživateľské rozhranie Proces Designera, webová verzia

Pri správe užívateľov a mapovaní definovaných úloh prístupných pre rôzne BPMS funkcie je vhodné využívať organizačné štruktúry obsiahnuté v existujúcich systémoch, ako napríklad LDAP.

1.3.1.5 Monitoring

Proces, ktorý je úspešne implementovaný a nasadený vyžaduje monitoring v podobe sledovania, merania a kontroly ukazateľov rýchlosti, efektívnosti a účinnosti. Bežným prostriedkom na meranie účinnosti a efektívnosti procesov sú ukazatele KPI. Tieto hodnoty nám dávajú potrebné informácie k tomu, aby sme zistili či je potrebné vykonať úpravy v návrhu procesu či dokonca nástrojov použitých pri jeho implementácii. Príkladom sledovania je schopnosť určiť stav objednávky zákazníka (napr. objednaný príchod, čakajúci na doručenie, zaplatenú faktúru).

Miera monitorovania závisí na tom, aké informácie chce organizácia zbierať a analyzovať alebo na tom, či má byť sledovanie vykonávané globálne v reálnom čase či ad-hoc v konexte zamerania len na určitý celok. Monitorovanie obchodných procesov sa zvyčajne vykonáva pomocou dashboardov a hlásení založených na pravidlách, a to najmä v organizáciách, ktoré majú vlastnú IT infraštruktúru, ktorú môžu využiť na iniciatívy BPM.



Obr. 1.5: Dashboard v procesnom portále IBM BPM

1.3.1.6 Optimalizácia

Optimalizácia procesu zahŕňa získavanie informácií o výkonnosti procesu z fázy modelovania alebo monitorovania. Určuje potenciálne alebo skutočné prekážky či príležitostí na úspory nákladov alebo iné zlepšenia a následne umožňuje aplikovať tieto vylepšenia v ďalšom vylepšení návrhu procesu. To znamená, že po tejto fáze sa cyklus môže znova opakovať, aby jeho výsledky viedli k neustálemu zlepšovaniu.

1.4 Použitie a výhody BPM

Procesný prístup riadenia odstraňuje všetky nevýhody tradičného funkčného riadenia. Okrem toho má niekoľko vlastných výhod, vďaka ktorým je jedným z najefektivnejších a najpoužívanejších prístupov v dnešnej dobe. Má ale taktiež aj nejaké nevýhody.

1.4.1 Výhody BPM

Prínosy BPM závisia od stanovených cieľov projektu. V prípade, že dobre rozumieme stanovenému cieľu a požiadavkam zákazníka, môžeme výrazne ovplyvniť návrh procesov a s tým spojené výsledky. BPM prináša nový pohľad na dôležitosť vykonávaných činností, stanovenia zodpovednosti za konkrétne úkony a celkovú kvalitu vykonanej práce z pohľadu podniku. Okrem toho vedie k týmovej spolupráci a zlepšeniu podnikovej kultúry. [14]

Prínosy BPM možno rozdeliť na tri oblasti:

Oblasť riadenia spoločnosti

- flexibilná reakcia na zmeny trhu alebo požiadavky zákazníka vďaka väčšej zodpovednosti a samostatnosti pracovníkov
- delegácia právomocí - vznik nezávislých procesných týmov komunikujúcich medzi sebou
- zjednotenie popisu pracovných postupov
- definovanie a meranie strategických cieľov a ich prepojenie na KPI
- riadenie životného cyklu procesov aktívnym prístupom a kolaboráciou užívateľov (sledovanie, meranie, vyhodnotenie a zlepšovanie)

Oblasť ľudských zdrojov

- zvýšenie aktivity spolupracovníkov, väčšia zodpovednosť a menší dôraz na ich kontrolu
- zjednodušené a transparentné zavedenie pracovných rolí a príslušných zodpovedností

Oblasť informačných technológií

- šanca odhalenia nedostatkov procesu už pri jeho návrhu
- centralizovaná dokumentácia všetkých procesov, dát, informačných systémov, rizík, organizačnej štruktúry
- použitie benchmarkingu (kontinuálny a systematický proces definovania a porozumenia aktuálnych princípov spoločnosti a ich zmena za účelom definovať ciele a postupy pre zlepšenie efektivity a výsledkov podniku) [15]
- priestor pre optimalizáciu a zlepšenie procesov využitím iných nástrojov

1.4.2 Nevýhody BPM

Zavedenie a používanie BPM má samozrejme aj určité nevýhody. Tie sa často mylne zamieňajú za problémy, ktoré súvisia priamo zo zavedením samotného BPM. Je to krátkodobý chaos a nadčasy zamestnancov v práci.

Naopak, vznik krátkodobého chaosu je skôr ukazateľom neefektívnosti starého systému a podnikovej kultúry. Nový systém riadenia nie je možné realizovať zo dňa na deň. Problémy pri zvýšenej aktivite a nadčasoch pracovníkov je možné kompenzovať hlavne prostredníctvom finančných odmien vo forme bonusov. [16]

Každopádne, existuje niekoľko skutočných nevýhod BPM, ktoré delíme na firmou ovplyvniteľné a tie, ktoré spoločnosť ovplyniť nedokáže.

Spoločnosťou ovplyvniteľné nevýhody Hlavným negatívom, ktoré môže podnik ovplyvniť, je prepustenie zamestnancov, ktorí sa v dôsledku reorganizácie spoločnosti stali nadbytočnými. Častým dôsledkom je zmenšenie oddelení o viac ako polovicu vďaka efektivite procesov. [17] Spoločnosť má ale možnosť túto mieru ovplyvniť a to nasledujúco:

- **Dočasná alebo trvalá zmena práce zamestnancov.** Práca by mala byť poskytnutá v rozsahu znalostí zamestnanca či jeho pôvodného platového ohodnotenia.
- **Obmedzenie či úplné zrušenie niektorých externých dodávok.** Ak spoločnosť využíva služby externých zdrojov, je na zvážení a analýze, či je firma schopná tieto služby vykonávať s využitím svojich interných zdrojov. Ťieto povinnosti môžu následne vykonávať zamestnanci, ktorí by boli prepustení z dôvodu nedostatku práce.
- **Dočasné pozastavenie vykonávanie niektorých pracovných činností.** Ide o krajné riešenie nadbytočných zamestnancov na pracovných pozíciách. V rámci negatívnych vplyvov to môže spôsobiť, že pracovníci sa budú obávať úplnej straty zamestnania, no na druhú stranu to môze znamenať odpočinok, po prestávke väčšie pracovné nasadenie a pre spoločnosť čas na riešenie daného problému

Spoločnosťou neovplivniteľné nevýhody Dané negatíva sú skôr filozofického charakteru, ktoré súvisia so zrýchľovaním rozvoja vedy a techniky, neustálym rozvojom firiem a tým súvisiacich problémov. [17] Tento fakt pôsobí z hľadiska vedeckotechnického rozvoja veľmi pozitívne. Podniky vytvárajú viac pridaných hodnôt za použitia know-how a celkovo je život ľudí omnoho kvalitnejší. Z historického hľadiska rast prebieha tak rýchlo, že firmy naňho nestíhajú reagovať, čo vedie k vnútornému chaosu a v dôsledku toho aj budovaniu slabej konkurencie na trhu.

1.5 Zhrnutie kapitoly

V rámci tejto kapitoly som stručne predstavil význam a princípy zavádzania BPM. Táto oblasť je špecifická a je určená najmä podnikom procesne orientovaným, čo sú prevažne bankové inštitúcie, poisťovne či automobilky. Pre zachytenie a prácu s týmito procesmi sú vytvorené nástroje, ktorým sa budem venovať v následujúcej kapitole.

Kapitola 2

Predstavenie BPMS

Celým názvom Business Process Management Suite je sada softvérových nástrojov používaných na zlepšenie podnikových procesov organizácie. Podľa spoločnosti Gartner je BPMS [18] "*hlavná aplikačná infraštruktúra na podporu projektov a programov BPM. BPMS podporuje celý životný cyklus zlepšovania procesov - od zisťovania procesov, definície a návrhu až po implementáciu, monitorovanie a analýzu a priebežnú optimalizáciu. Jeho prístup založený na modeloch umožňuje podniku a odborníkom v oblasti IT spolupracovať v celom životnom cykle viac, než je možné s inými prístupmi poskytujúcimi riešenie.*"

V porovnaní s BPM je teda podmnožinou, ktorá bližšie špecifikuje nástroje BPM, aké sú napríklad integrácia, BI alebo monitorovanie činností. Pokročilé BPMS poskytujú vývojové nástroje na tvorbu aplikácií založených na formulároch, ktoré sú často začiatkom množstva obchodných procesov. Existujú tri kľúčové typy BPMS:

1. **Monitorovacie:** ich úlohou je monitorovať každý systém podniku z dôvodu neefektívnosti procesov, a to sledovaním od začiatku do konca. Nástroj presne určuje slabosť a prekážky, kde môžu byť zákazníci frustrovaní a prerušiť transakcie a procesy.

2. **Optimalizačné:** používa podrobné mapy existujúcich procesov a snaží sa ich zefektívniť optimalizáciou určitých krokov. Samotný nástroj nemôže navrhnúť vylepšenia procesu, len ho optimalizovať, takže je taký dobrý ako samotný proces.

3. **Integračné:** alebo odborne EAI ide o kombináciu monitoringu optimalizácie. EAI sa používajú na integráciu starších systémov do nových, kde pomocou mapovaných bodov umožňuje ich vzájomnú kooperáciu. Ide o čoraz dôležitejší požiadavok, keďže systémy vyžadujú časté prepojenia medzi ERP, CRM, účtovníctom či ďalšími dátovými systémami, ktoré v sebe nesú dôležité informácie.

2.1 BPMS v súčasnosti

Podľa celosvetového prieskumu firmy Forrester Research z roku 2017 [19] iba 26% procesne orientovaných firiem využíva systémy BPMS na podporu väčšiny svojich obchodných procesov. Avšak až 69% si myslí, že tieto systémy budú podporovať väčšinu procesov svojej firmy v priebehu nasledujúcich dvoch rokov.

Trh nástrojov pre BPM sa dramaticky mení, tak ako sa postupne vyvíjali oddelené funckionality jednotlivých typov aplikácií - nástrojov pre modelovanie a analýzu, workflow systémov pre riadenie pracovného toku a dokumentov, nástrojov pre vývoj v IDE, aplikácií pre meranie výkonnosti, systémov pre definíciu a riadenie pravidiel, ale aj integračných platforiem.

Je evidentné, že stále existuje dostatok priestoru na rast, čo môže byť dôvodom, prečo sa v tejto oblasti snaží angažovať veľké množstvo spoločností a uviesť na trh vlastné riešenie, čo potvrdil aj Maureen Fleming, analytik spoločnosti IDC [20].

2.2 BPMS frameworky

Spolu s BPMS systémami sa postupne vyvinulo množstvo kritérií, ktoré musia systémy spĺňať, aby boli použiteľné. Ide o tzv. systémy druhej generácie, ktoré by mali byť schopné podporiť celý životný cyklus podnikania - od stratégie cez procesy až k IT a to v jednotnom prostredí s jednotnými konzistentými datami. Odpadá tak potreba používať niekoľko nástrojov pre fázu dizajnu procesov, jeho realizáciu a meranie výsledkov ako i v rámci analýzy a dostupnosti reálnych dát instancií procesov, ktoré je možné nativne analyzovať či simulovať nové scenáre. [21]

Existujú tri základné typy BPMS frameworkov:

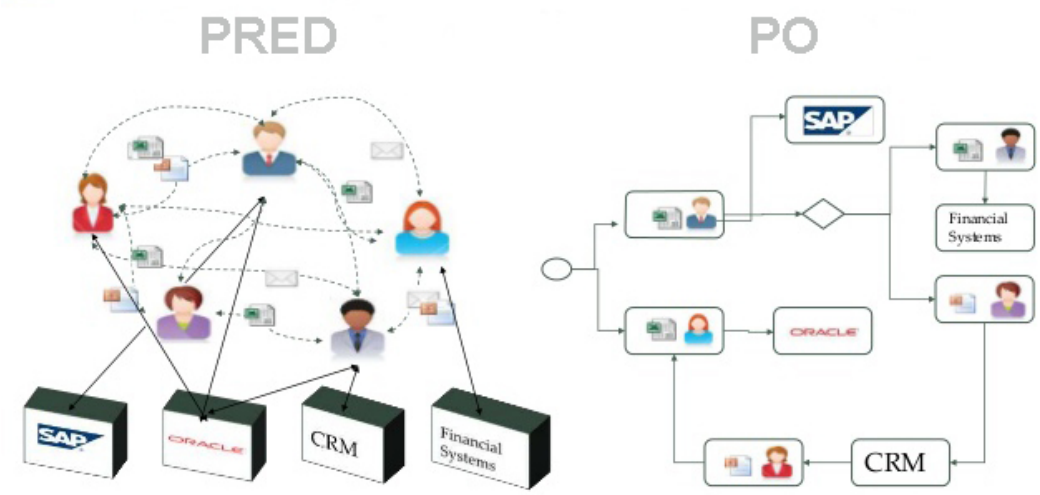
1. **Horizontálne** sa zaoberajú návrhom a vývojom obchodných procesov, zameriavajú sa na technológiu a opätovné použitie

2. **Vertikálne** sa zaoberajú špecifickým súborom koordinovaných úloh a prichádzajú s vopred vytvorenými šablónami, ktoré možno ľahko nakonfigurovať a nasadiť

3. **Full-service** ponúkajú 5 kľúčových prvkov: modelovanie a návrh obchodných procesov, definíciu obchodných pravidiel, simuláciu, testovanie

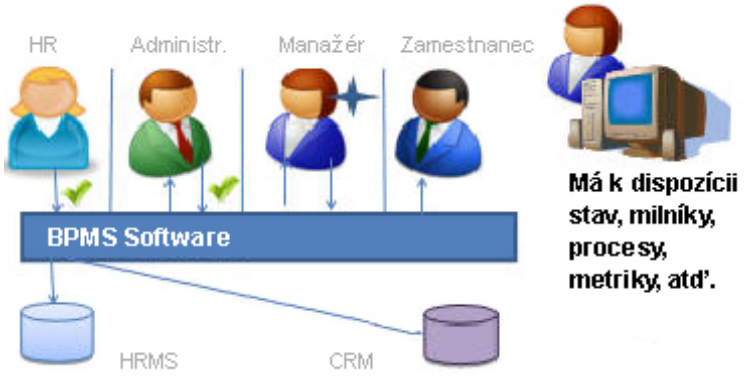
V súčasnosti sú najviac využívané BPMS systémy tretieho typu - Full service. Využitím týchto systémov pri zavádzaní BPM na základe životného cyklu 1.3.1 je možné výrazne zjednodušiť a zprehľadniť fungovanie organizácie. Ako ukážka slúži 2.1, kde je zobrazená vzájomná komunikácia zamestnancov a ich priame využívanie ďalších systémov v organizácii v stavoch pred a po

zavedení BPMS. V stave po úprave procesu využitím BPMS je na prvý pohľad vidieť pozitívny posun pri organizácii procesu a vzájomnej komunikácie jeho účastníkov.



Obr. 2.1: Stav pred a po zavedení BPMS

Nástroje BPMS vytvárajú vrstvu 2.2, ktorá používateľom poskytuje možnosť jednotne pristupovať k interným systémom či získať jasný prehľad o úlohách a stave ich požiadavkov. Manažérske pozície získavajú centralizovaný prístup k všetkým informáciám potrebným k správnemu vedeniu svojho oddelenia alebo celej spoločnosti.

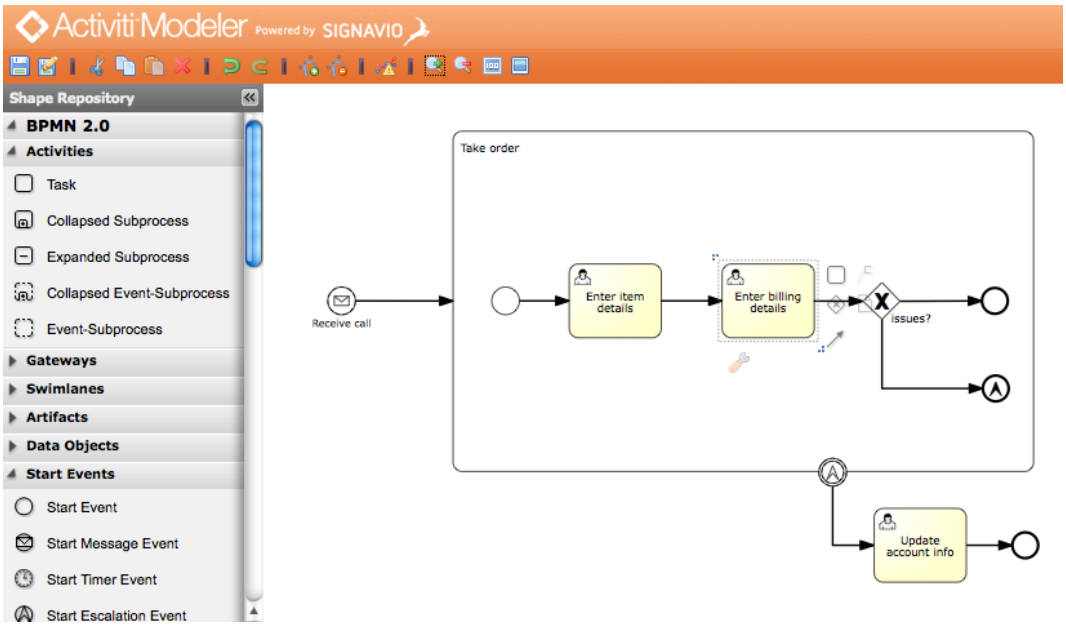


Obr. 2.2: Medzivrstva BPMS nástrojov zjednocujúca ľudské zroje a systémy spoločnosti

Takéto zlepšenie je možné dosiahnuť bezohľadu na typ použitých BPMS. Každý z nich umožňuje proces definovať a implementovať od úplného základu, bez viazanosti na preddefinované šablóny. Poskytujú tak absolútnu kontrolu nad celým procesom a jeho časťami, ktoré sú vytvárané na mieru vzhľadom na situáciu a požiadavky organizácie. I napriek tomu sa ale jednotlivé systémy odlišujú. Ich bližšej analýze a technickým parametrom sa podrobnejšie venujem v následujúcej časti.

2.2.1 Activiti

Aktiviti nástroj je produkt ECM, Alfresco, ktorý chcel vyvinúť alternatívu k jBPM pre svoje vlastné účely. Activiti je publikovaná pod Apache Licence ako open source a je vyvíjaná v jazyku Java.



Obr. 2.3: Activiti IDE pri návrhu procesov

2.2.1.1 Popis

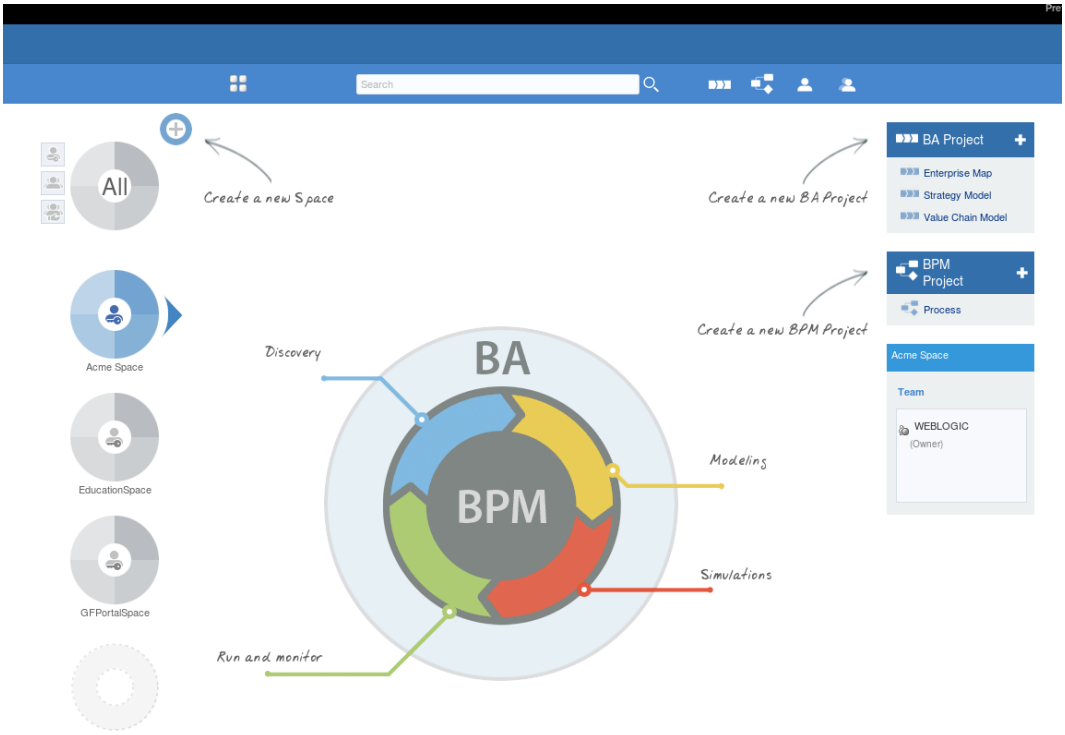
Nástroj podporuje všetky aspekty procesného modelovania v kontexte s vývojom softwaru. To zahrňuje nielen analýzu, modelovanie a optimalizáciu podnikových procesov, ale taktiež pre ne vytvorenej softwarovej podpory. Hlavným rozdielom oproti ostatným spomenutým nástrojom je ten, že logiku a funkčnosť procesov je potrebné implementovať prostredníctvom kódu podľa princípov OOP. Platforma síce poskytuje podporné knižnice či predpripravené

funkcie, no i napriek tomu je oproti ostatným nástrojom výrazne zvýšená doba vývoja. Naopak, keďže je kód písaný pre každý projekt zvlášť, dáva tak vývojárom voľnosť pri vytváraní produktov.

Veľkým nedostatkom platformy je, že neumožňuje definovať užívateľské práva a nepodporuje týmovú prácu, teda využitie danej funkcionality záleží na organizácii práce v podniku.

2.2.2 Oracle BPM

Software je tvorený viacerými modulmi a je plne kompatibilný s ostatnými produktami firmy Oracle. Najnovšou a následne aj ďalej analyzovanou verziou Oracle BPM je 12c.



Obr. 2.4: Oracle BPM Composer, webová aplikácie umožňujúca vytvárať, prispôsobovať a analyzovať obchodné procesy

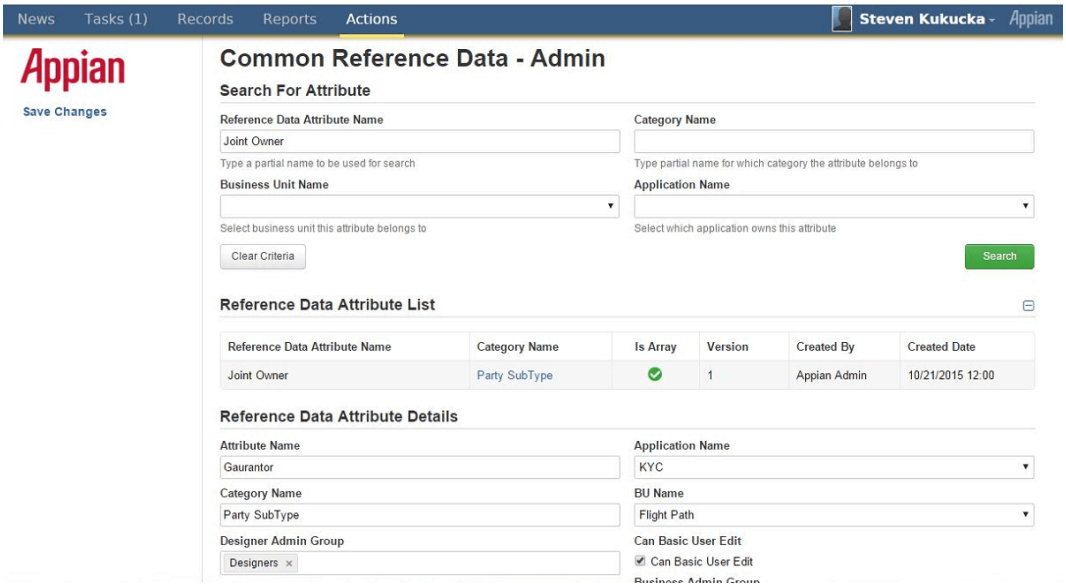
2.2.2.1 Popis

Oracle BPM disponuje silnými analytickými nástrojmi, ktoré zvyšujú prehľadnosť procesných výkonov a inteligentnejšie riadia procesy pomocou analýzy

v reálnom čase (Oracle R pre prediktívnu analýzu a Oracle Stream Explorer pre analýzu streamingu) a spracovanie udalostí (prostredníctvom Oracle Event Processing). Ponúka taktiež predinštalované integrácie do iných produktov Oracle, vrátane Oracle SOA Suite a produktov ERP spoločnosti Oracle, ako aj niekoľko adaptérov pre aplikácie tretích strán (napríklad Salesforce a SAP). Nevýhodou tejto platformy je, že aplikácie postavené na balíku Oracle BPM nie sú prenosné na Oracle Process Cloud Service. To znamená, že balík Oracle BPM je dostupný iba ako *on-premise* riešenie.

2.2.3 Appian

Tento BPMS nástroj býva označovaný ako "low-code", teda s využitím minimálneho množstva kódu, ktorý razí cestu jednoduchosti v rámci použitia. Je k dispozícii ako ön-premise"či cloudové riešenie.



Obr. 2.5: Užívateľské rozhranie spustenej úlohy v systéme Appian

2.2.3.1 Popis

Systém umožňuje rýchle vytváranie formulárov a dashboardov, je efektívny aj pri spracovaní komplexných udalostí. Zákaznícke referencie Appianu poukazujú na vyššiu mieru spokojnosti s platformou v porovnaní s ostatnými poskytovateľmi BPMS riešení. Vďačí tomu najmä veľmi dobrému užívateľskému rozhraniu. Výhodou je aj pomerne rozsiahla komunita vývojárov, ktorý sa združujú na špeciálnom Appian fóre.

Platforma umožňuje sťahovanie tzv. pluginov či widgetov, ktoré uľahčujú prácu či doplňujú dodatočnú funkcionalitu. Nevýhodou je, že tieto pluginy vývojári nemajú možnosť upravovať, takže sú odkázaný len na to, čo im tvorcovia poskytnú. To isté platí aj o GUI pluginoch, ktoré majú jasne definovaný dizajn a zmeny v podobe vlastných CSS štýlov nie sú možné.

Za ďalšiu nevýhodu sa považuje podpora vlastného programovacieho jazyka, podobného Lispu, čo vynucuje funkcionálne programovanie.

2.2.4 IBM BPM

Nástroj spoločnosti IBM, navrhnutý tak, aby umožnil vlastníkom procesov a ich používateľov priame zapojenie do zlepšovania procesov spoločnosti. Je k dispozícii ako *ön-premise* alebo cloudová varianta a je navrhnutý tak, aby podporoval mobilné zariadenia s funkciami administratívy naprieč verziami produktu.

2.2.4.1 Popis

Tak ako ostatné BPMS nástroje umožňuje organizovať, spravovať, monitorovať a rozširovať procesné artefakty, aplikácie a to všetko centralizovane v jednom nástroji, bez potreby ďalších. Poskytuje viditeľnosť celého procesu a dáva užívateľom možnosť prístupu k jeho častiam, aktivitám a dashboardom formou jednotného UI. Prostredníctvom procesného portálu používateľom poskytuje pracovné prostredie s možnosťou intenzívnej spolupráce. Pre vývojárov je veľkým prínosom možnosť paralelnej práce.

Za nedostatky je považovaná inštalácia, ktorá sa neozaobíde bez asistencie skúsených odborníkov. Všeobecným problémom je aj chýbajúca možnosť vyhľadávania častí kódu a teda i náročný refaktoring v rámci nahradzovania premenných naprieč aplikáciou.

2.3 Zhrnutie kapitoly

Všetky vyššie spomenuté platformy majú svoju komunitu vývojárov a projekty, na ktorých spĺňajú svoj účel. V následujúcej kapitole sa ale zameriam len na jeden a tým je IBM BPM, kde sú moje skúsenosti ako jedného z vývojárov najväčšie. Oproti ostatným mi ale jednoznačne vyhovuje centralizovaná forma vývoja, bez potreby využitia ďalších externých nástrojov.

Kapitola 3

IBM BPM pod lupou

Cieľom tejto kapitoly je podrobný rozbor zvoleného nástroja pre prácu s obchodnými procesmi.

Všeobecným problémom pri vývoji akýchkoľvek aplikácií býva na úvod dobrá voľba tej správnej a kvalitnej technológie. Od tohoto rozhodnutia sa odvíja rýchlosť vývoja, náročnosť údržby alebo odbornosť vývojárov. Niektoré technológie sú veľmi špecifické a preto sa tím kvalitých vývojárov hľadá veľmi ťažko a projekt sa stáva o to nákladnejším.

3.1 Úvod

Zdroj [17] definuje, že obchodné procesy obvykle existujú v mysliach a praxi zamestnancov, no je veľmi náročné ich premietnuť a zabezpečiť, aby boli pochopené i zo strany vývojárov. Implementácia, ktorá je jednou z hlavných výhod IBM BPM (IBPM), umožňuje výrazne skrátiť trvanie porozumenia prostredníctvom interaktívnych zasadnutí, kde je proces, pomocou BPMN, priamo zachytený v IBPM ako diagram. Tento model procesu, je možné priamo spustiť a umožniť tak stranám, ktoré rozumejú jeho povahe, aby potvrdili, že to, čo bolo implementované je správne. V prípade nájdenia chyby, je možné proces opraviť v rámci IBPM nástroja, až pokým nebude správne zachytený a pochopený i zo strany vývojárov. Tento interaktívny a realtime cyklus je vďaka IBPM veľmi flexibilný čo je hlavný rozdiel a benefit pri zrovnávaní s podobnými produktami na trhu.

Ďalšia, nemenej dôležitá vlastnosť je integrácia. Množstvo obchodných procesov v sebe zahrňuje iba prenášanie úloh z jedného užívateľa na druhého, no majorita systémov vyžaduje integráciu či vzájomnú komunikáciu. IBPM poskytuje schopnosť vnímať externé systémy ako časti procesov. Samotná integrácia je potom realizovaná formou webových služieb alebo iných komunikačných volaní a to všetko v rámci jednej platformy, prípadne procesu.

3.2 Architektúra

K získaniu stavu 2.2 je IBM BPM prispôsobený celou svojou architekúrou, ktorej sa budem venovať v tejto časti. Každá z nižšie popísaných častí plní v tejto technológii svoj jedinečný a nezastupiteľný účel. Pre lepšiu predstavu, z akých častí sa skladá IBM BPM, slúži tento 3.1 obrázok.

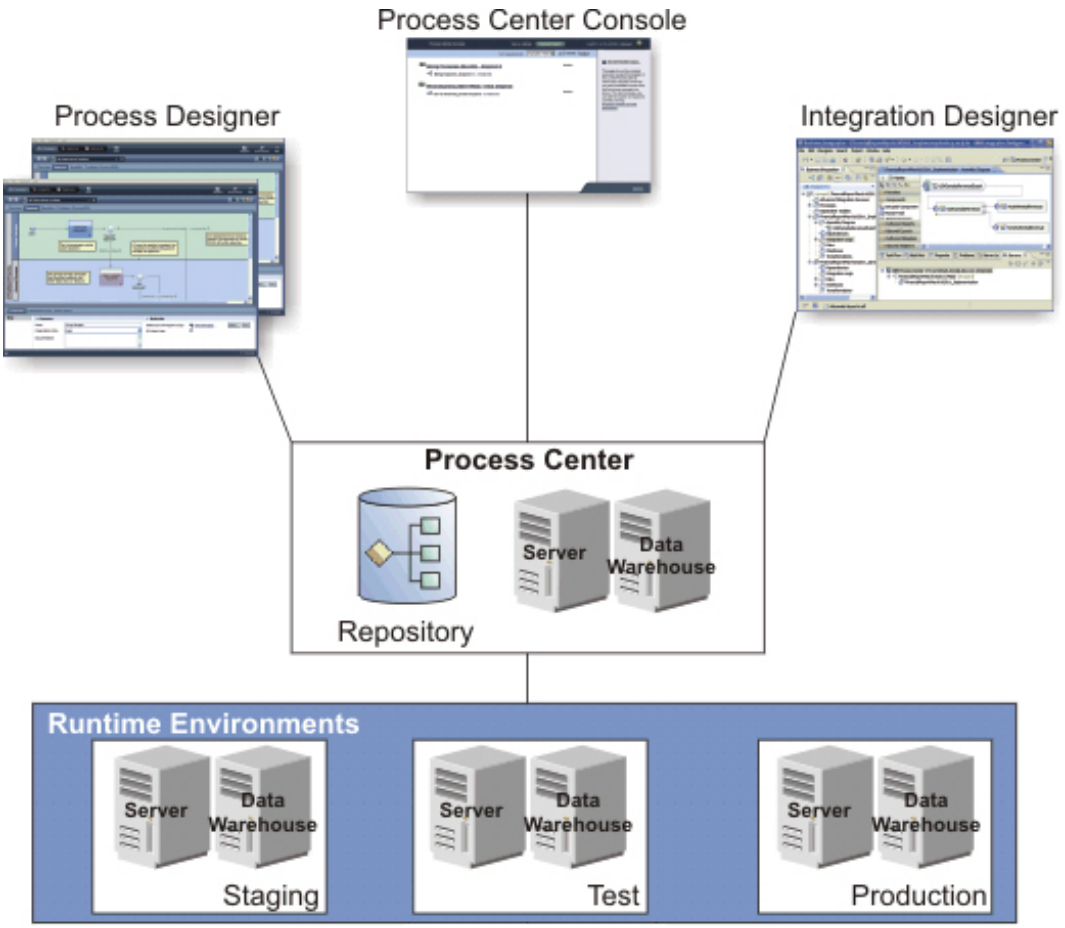
Proces Server V práci [13] je server definovaný ako motor pre spúšťanie procesov vytvorených vyvojármi a uchovávaných v Procesnom Centre. Každé prostredie (vývojárske, testovacie či produkčné) má svoj vlastný Procesný server, ktorý sa stará o príslušné procesy. Procesný server je implemenetovaný prostredníctvom IBM WebSphere Application Server (WAS), ktorý poskytuje prostredie pre beh podnikových aplikácií v jazyku Java.

Proces Center Procesné centrum je softvérová komponenta, ktorá zastrešuje účel servera. Skladá sa z dvoch hlavných komponent - server Procesného centra a Data Warehouse, ktorý slúži ako repozitár. Tieto časti umožňujú užívateľom pracovať s Procesným Designérom, spúšťať či vytvárať procesné aplikácie a zároveň uchovávať všetky data potrebné pre testovanie, monitoring či vývoj. V procesnom centre sú okrem aplikácií viditeľné aj všetky ich verzie, ktoré je možné nasadiť na nakonfigurované prostredia či exportovať ich zdrojový kód vo formáte XML, keďže je to jediný spôsob ako vyhľadávať v kóde fulltextovo. Ďalšou súčasťou PC je jeho konzola, ktorou administrátori inštalujú procesné aplikácie na iné prostredia procesných serverov. Slúži tiež k správe bežiacich instancií procesov, konfigurácii hostiteľského prostredia či správe prístupov a oprávnení všetkých užívateľov.

IBM BPM Proces Designer Používatelia v IBM Process Designer vytvárajú modely procesov, služby a ďalšie aktíva v procesných aplikáciách či toolkitoch. Proces Designer je ľahko použiteľný grafický nástroj, ktorý umožní modelovať a implementovať obchodné procesy založené na BPMN, s možnosťou ich priameho spustenia a monitorovania priamo v tomto nástroji. Následne umožňuje ich testovanie a ladenie.

V súčasnosti je k dispozícii mimo dekstopovej už aj webová verzia. Podľa aktuálnych informácií je vývoj BPM Designéra smerovaný práve týmto smerom, kde cieľom je všetko presunúť do cloudu a odstrániť závislosť na operačnom systéme Windows. Nevýhody webového designéra sú hlavne v rýchlosti a v prípade väčších projektov aj dlhej odozvy, pretože je potrebné získavanie veľkého množstva informácií a prekresľovanie jednotlivých častí UI. V rámci kompatibility, webový designér (v8.7) obsahuje niektoré prvky, ktoré v desktopovom nenájdeme, a spustenie či zobrazenie takýchto častí nie je možné, takže spätná kompatibilita je čiastočná.

Všetky vývojové artefakty vytvorené v IBM Process Designer sa spravujú Procesným Centrom a implicitne sú uchovávané v repozitáre Process Center.



Obr. 3.1: Pohľad na IBPM architektúru

Veľkym benefitom Proces Designéra je, že podporuje paralelnú prácu viacerých užívateľov v rámci jednej aplikácie. Pricíp spočíva v tom, že akonáhle užívateľ začne upravovať určitý artefakt, tak sa uzamkne a nie je možné do týchto úprav zasahovať iným užívateľom. Ostatné časti ale zostávajú stále odomknuté, ako i časti, ktoré artefakt používa ako napríklad ďalšie vnorené služby.

IBM BPM Procesny Portal Ide o webovú komponentu, ktorá umožňuje zobrazenie a spustenie všetkých procesov, v závislosti na aplikácii, prostredí či prihlásenom užívateľovi. Je to frontendová vrstva, ktorá slúži ako prístupový bod koncovým užívateľom. Okrem správy profilu užívateľa, zobrazenia dashboardov či štartovania nových procesov, zobrazuje aj task list, z ktorého užívateľ otvára príslušnú úlohu (časť procesu), kde je po jej vykreslení zobra-

zená príslušná obrazovka vytvorená v rámci PD. Tasky sú zoradené podľa deadlinov, kde tie, ktorým sa blíži doba expirácie sú zobrazené ako prvé. Navyše sú odlíšené aj farebne a je možne ich filtrovať na základe fulltextového vyhľadávania (v rámci názvu a ID) či regulárnych výrazov špecifických pre Procesný Portál.

3.2.0.1 Vývoj v IBM BPM

Vývojári majú k dispozícii dve možnosti vývoja a to formou desktopového alebo webového dizajnéra. Desktopový zatiaľ zostáva štandardom, no čoraz novšia verzia prináša stále ďalšie vylepšenia do webového, kde je evidentné, že IBM smeruje vývoj procesných aplikácií do cloudu.

Obrovskou výhodou webového dizajnéra je, že oproti desktopovému nie je závislý na operačnom systéme. Pre vývoj je k dispozícii od verzie 8.6, v súčasnosti už vo verzii 8.7, kde prináša lepšiu optimalizáciu a je doplnený o funkcionalitu z desktopového ako sú definícia procesu či služieb, ktoré sú s desktopovými kompatibilné len čiastočne. Veľkou nevýhodou je, že pri modelovaní rozsiahlych procesov jeho optimalizácie nepostačuje na plynulú prácu. Pre menšie projekty je ale plne postačujúci a pracuje sa v ňom veľmi pohodlne.

Definícia procesu Proces sa v rámci IBM BPM definuje využitím procesného diagramu a to formou BPMN notácie, čo umožňuje zachytiť akýkoľvek tok procesu. Po vytvorení diagramu, je proces možné následne spustiť a otestovať jeho chod bez náročnejšej logiky.

Reprezentácia premenných Ak je proces spustiteľný a odladený, pridáme k nemu možnosť pracovať s nejakými datami. Tie budú uchovávané v premenných, ktoré môžu byť používané aj v rámci náročnejšej rozhodovacej logiky. V prípade, že nepostačujú jednoduché datové typy ako sú String, Integer, Boolean atď. je možné v BPM Designéri vytvárať vlastné typy, reprezentované ako Business Object. Ide o štruktúry, ktoré združujú viaceré jednoduché typy, prípadne obsahujú ďalšie Business Objekty.

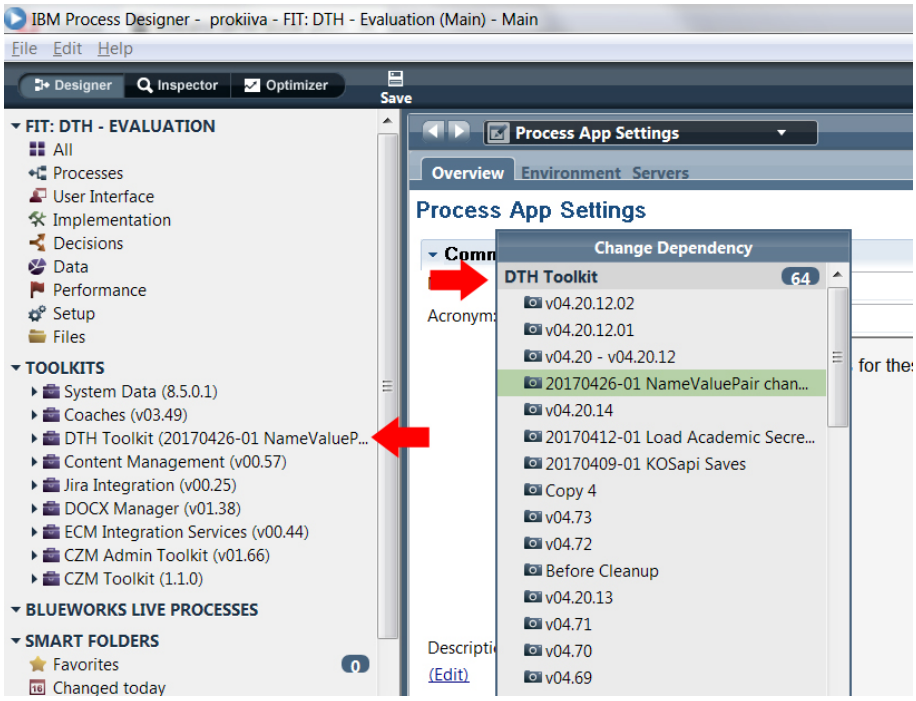
UI - Coach view Pri vytváraní jednotlivých častí procesu, je v prípade užívateľskej obrazovky potrebné zobraziť užívateľsky prívetivú informáciu - užívateľské rozhranie (UI). PD má vytváranie UI riešené formou drag and drop editora, ktorý obsahuje frontendové prvky založené na HTML, CSS a JavaScripte. Tie následne môžeme zoskupovať do znovupoužiteľných celkov nazývaných Coach Views (CV), ktoré vystupujú samostatne alebo sú taktiež súčasťou iných CV a vytvárajú tak hierarchickú štrukúru. Takéto celky sú následne používané v artefaktoch zvaných Human Service, ktoré sú priamo naviazané na časť procesu vyžadujúcu iterakciu s užívateľom.

Každý CV obsahuje svoje vlastné konfiguračné premenné, ktoré v sebe nesú vstupné data zadávané v mieste použitia (napr. v nadradenom). Ide o

akýsi vstupný parameter pri "použití funkcie". Vo vnútry CV sú ďalej predávané ako parametre ostatným prvkom či iným vnoreným CV, a vytvárajú tak vnorenú štruktúru závislostí.

Toolkit Je podľa Kolbana [22] podobný procesnej aplikácii. Toolkit môže byť považovaný za kontajner pre komponenty, napríklad tie, ktoré sú častokrát používané. Na rozdiel od procesnej aplikácie Toolkit neobsahuje nasaditeľnú - spustiteľnú aplikáciu. Namiesto toho obsah Toolkitu môže byť použitý jednou alebo viacerými procesnými aplikáciami. Do toolkitu sa umiestňujú najčastejšie služby, CV, objekty či iné komponenty používané v rámci viacerých aplikácií, keďže samotné aplikácie ich medzi sebou nezdieľajú a plnia teda účel podporného kontajnera.

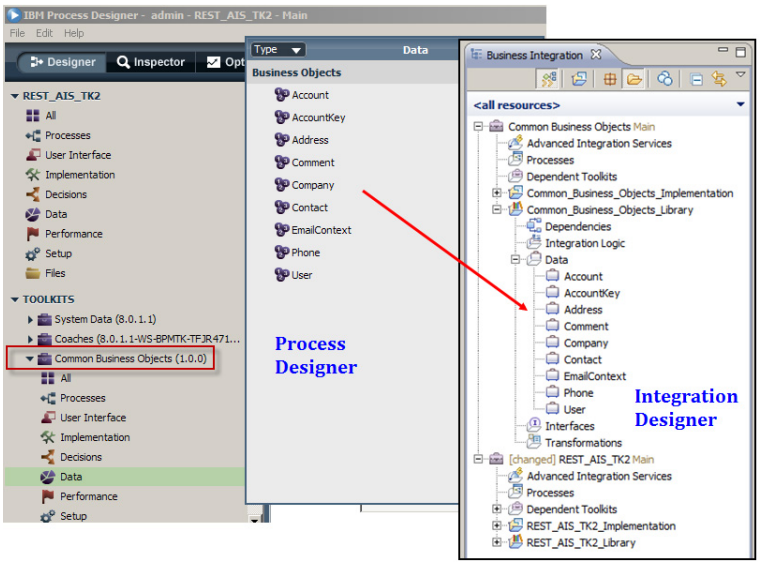
Toolkity sú, tak ako i procesné aplikácie, verzované. Dáva tak vývojárom možnosť pracovať na ich samostatne a postupne releasovať otestované funkčné celky, oddelené snapshotmi. Vytvorenie snapshotu je identické ako pri procesných aplikáciách, ktoré nesú pomenovanie a prípadnú rozšírenú informáciu v komentároch (častokrát podrobný výčet zmien). V prípade problémov v Toolkite je downgrade v rodičovskej aplikácii realizovaný zvolením nižšej verzie - snapshotu, z pomedzi všetkých nearchivovaných.



Obr. 3.2: Zobrazenie zoznamu závislostí aplikácie na danom Toolkite

Správa aplikácie Procesné centrum nám umožňuje nad každou aplikáciou samostatne vykonávať určité operácie. Je možné aplikáciu nasadiť - inštalovať, exportovať zdrojový kód v určitom formáte či spravovať snapshoty a vytvárať vetvy či kópie z prislušných aplikácií a jej verzií. Tieto akcie je možne iniciovať v Procesnom Centre 3.2. Inštalácia sa vykonáva na miesto, ktoré je zaznamenané v konfiguračnom súbore. Je možné inštalovať ktorýkoľvek viditeľný snapshot - verziu aplikácie na ktorékoľvek z nakonfigurovaných prostredí. PC ďalej dáva možnosť aplikácie archivovať, čo je IBM BPM špecifické zmazanie. Výlsledkom je, že zamedzí možnosť manipulovať s danou verziou či aplikáciou a je potrebné ju znova explicitne odarchivovať. Ide teda len o vizuálne skrytie nepotrebných častí histórie vývoja či omylov.

Ďalším nástrojom je Integračný dizajnér (ID), ktorý slúži k vytvoreniu a implementácii spojenia s inými aplikáciami a backend systémami 3.3 Jeho pomocou sa vytvárajú rozhrania ako napríklad WSDL, datové typy Business Objectov a XML mapy, ktoré slúžia k prepojeniu s externými systémami.



Obr. 3.3: Reprezentácia objektov v Procesnom a Integračnom designéri

3.3 Zhrnutie kapitoly

Aj napriek početným výhodám a jednoduchosti nie je IBM BPM ideálnym nástrojom. Jeho komplexnosť a početné možnosti ako dosiahnuť riešenie sú častou príčinou nejednotnosti implementácie, čo vedie k chybám pri vývoji alebo integrácií ďalších funkčných celkov. Dobrá znalosť a všeobecný prehľad týchto problémov môže viesť k zníženiu rizika výskytu a času potrebného k ich

identifikácii. Podrobnému rozboru a štatistikám je preto venovaná následujúca kapitola.

Kapitola 4

Najčastejšie vývojárske problémy v IBM BPM

V prípade každej platformy či jazyka je dôležitý dobrý návrh, ktorý funguje na prepoužívaní iných knižníc či častí kódu, problémy rieši dostatočne všeobecne alebo do hĺbky a už len samotnými konvenciami v návrhu smeruje vývojárov k dobre udržateľnému a prehľadnému kódu.

4.1 Úvod

Platforma IBM BPM na prvý pohľad vyzerá jednoducho, bez možností produkovať zle udržateľné či náročne rozširujúce aplikácie. No i v tomto nástroji existuje niekoľko možností ako dosiahnuť žiadúci výsledok, čo môže mať za následok nejednotnú implementáciu, či už je to návrhom procesov, služieb alebo zle štrukturovaných CV. Navonok jednoducho pôsobiaca, no zle navrhnutá komponenta v sebe môže obsahovať skrytú logiku, ktorú v lepšom prípade znefunkčníme a problém bude hneď viditeľný. Horším scenárom je zanesenie logickej chyby, ktorá nie je na prvý pohľad viditeľná a môže to mať pre uložené data deštruktívne následky.

Vývoj procesných aplikácií tiež vyžaduje iný pohľad kvôli bežiacim procesom, kde každá instancia procesu sa správa na základe posledného spustiteľného stavu procesu zachyteného pred jej spustením. To znamená, že instancia beží na základe definovaného procesu (diagramom) a jeho premennými. V prípade zmien procesu či jeho premenných, može bežiaca instancia naraziť na stav, ktorý pre ňu nie je definovaný, resp. nemá pre jeho vykonanie dostatok informácií (ako sa má zachovať, hodnota premennej, iný názov či typ premennej). Táto situácia vedie k neočakávanému správanu či dokonca k pádu aplikácie a zamedzeniu práce na danej instancii. Riešením je počkať, až všetký staré instancie dobehnú, čo môže byť veľmi časovo náročné alebo je tu možnosť vytvorenia migračných skriptov, ktoré majú za úlohu dodefinovať stavy,

v ktorých nie je zrejmé, ako sa má bežiaci proces správať.

4.2 Definície pojmov

Žiadna platforma sa nezaobíde bez problémov v rámci vývoja či testovania. Pre lepšie pochopenie vysvetlím pojmy, ktoré s touto problematikou súvisia.

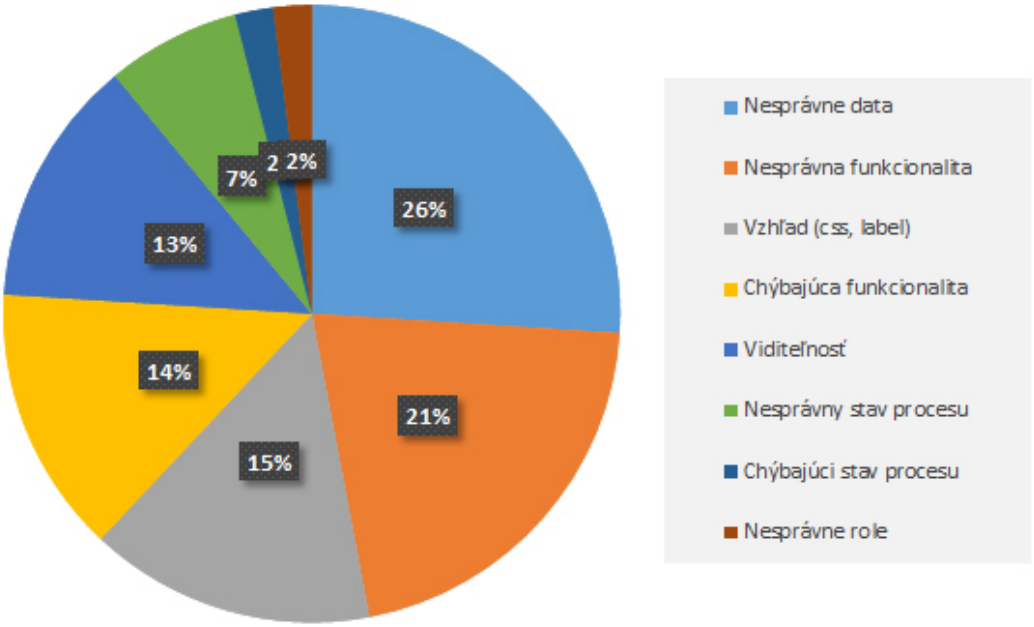
Chyba (angl. Bug) sa podľa Margaret Rouse [23] týka problému, nedostatku alebo nefunkčnosti v počítačovom programe či hardvérovom systéme. Chyba prináša neočakávané výsledky alebo spôsobuje neočakávané správanie systému. Stručne povedané, je to nejaké správanie, stav alebo výsledok, ktoré program alebo systém dosiahne, no nebol navrhnutý ho vykonať či dosiahnuť.

Zmenový požiadavok (angl. Change Request) John Spacey definuje ako [24] návrh na zmenu produktu alebo systému, ktorý často vyvoláva klient alebo iný člen tímu. Táto situácia môže nastať počas projektu ak chce klient zmeniť alebo doplniť dohodnuté výstupy.

Je dôležité poznamenať, že ani dobrý návrh a bezchybná implementácia nezabráni potenciálnemu zaneseniu chyby v budúcnosti. Príkladom takejto situácie je rozsiahly zmenový požiadavok, ktorý ma dopad na rôzne časti aplikácie. Pre lepší prehľad skutočných problémov a zmenových požiadavkov som v následujúcej časti vypracoval štatistiku najčastejšie hlásených problémov či zadávaných zmenových požiadavkov.

4.3 Štatistiky a kategorizácia

Pre účel vytvorenia všeobecného prehľadu nad množstvom a rôznorodosťou chýb či zmenových požiadavkov, som použil reálne informácie. Na ich evidenciu je využívaný trackovací systém Youtrack, ktorý je použítý na každom z niekoľkých projektov, ktoré mi slúžili ako zdoje. Informácie boli zozbierané v období fungovania projektov a to v horizonte 1 až 2 rokov. Počty chýb sú odhadované v niekoľkých tisícoch a zmenové požiadavky v stovkách. Zozbierané údaje som následne konzultoval a doplňoval o ďalšie poznatky viacerých seniorných vývojárov. Zo zozbieraných údajov som vyhodnotil, že charakter problémov sa opakuje, čo mi umožňuje ich následne kategorizovať na základe oblastí, ktorej sa priamo týkajú a výstupom sú následujúce diagramy, ktoré zobrazujú ich percentuálne zastúpenie.



Obr. 4.1: Kategorizácia chýb a ich percentuálne zastúpenie

S kategóriami na diagrame 4.1 sa spájajú následujúce problémy:

- **Nesprávne data** - ide o najčastejší typ problému, kde je spravidla potrebné debugovať väčší celok až sa postupne dopracujeme ku konkrétnemu artefaktu.
- **Nesprávna funkcionalita** - situácia kedy užívateľ dostane neočakávanú odpoveď. Napríklad: otvorenie nesprávneho dialogového okna ako reakcia na klik
- **Vzhľad (css, label)** - problém frontendu, ktorý nastáva na komponentách. Najčastejšie ide o zlé zarovnanie či dizajn určitej komponenty alebo jej časti.
- **Chýbajúca funkcionalita** - očakáva sa určitá interakcia, no tá sa nedostaví. Buď ako následok pádu predchádzajúcej funkcionality alebo chybné previazanie či doplnenie závislosti príslušného handlera. Napríklad: kliknutím sa nič nespustí, prvok sa nepridá do tabuľky, potvrdením sa neodošle email.
- **Viditeľnosť** problém týkajúci sa atribútov, nadpisov alebo aj celých komponent. Stáva sa pri prechodoch procesom, kde sa určitá časť zobrazuje dynamicky. Napríklad: pri definícii témy záverečnej práce sú

formulárové polia viditeľné a editovateľné, no pri fáze náhľadu sú stále viditeľné, no už by nemali byť editovateľné.

- **Nesprávny stav procesu** - následuje pád v prípade, že objekt neobsahuje dáta očakávané v tomto stave. Predpokladom je chybná podmienka v rámci procesu.

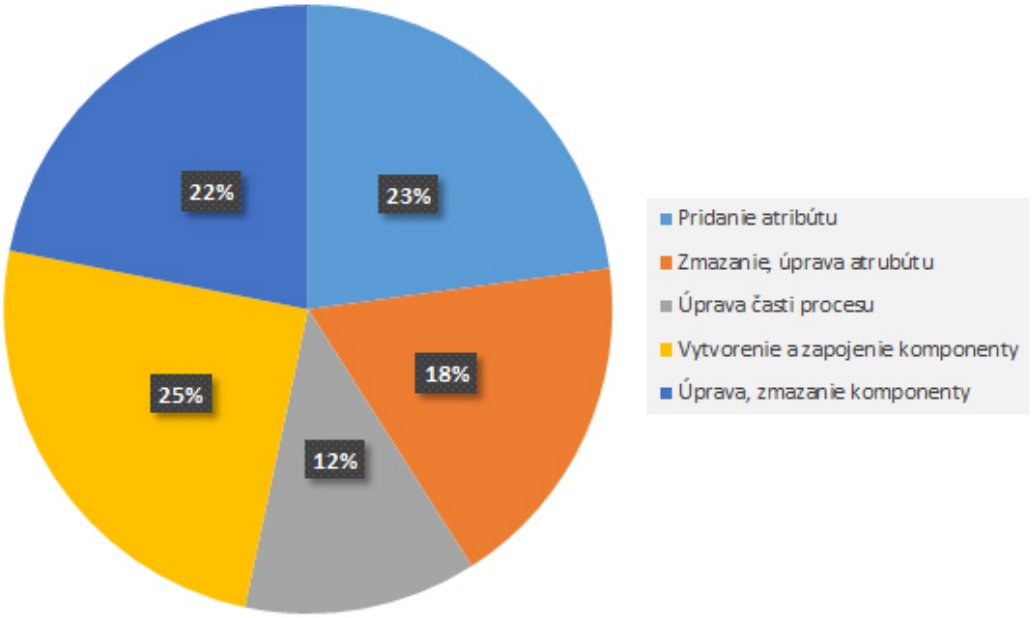
- **Chýbajúci stav procesu** - často následuje pád alebo ukončenie procesu, presunutie do stavu kompletný, v prípade prehľadného modelu ide o jednoduchší problém, ktorý nastáva len zriedka.

- **Nesprávne role** - užívateľ získa kompetenciu k spusteniu úlohy, ktorá nie je preňho určená. V niektorých prípadoch závisí od práv samotná viditeľnosť komponent, teda či je editovateľná alebo v stave zobrazenia (readonly).

Tento výčet samozrejme nie je konečný, ide o najčastejšie opakujúce sa problémy, na ktoré je možné vytvoriť všeobecnú sadu doporučení. Na projektoch sa ale občas vyskytnú, síce v menšom počte, no o to závažnejšie problémy, ktoré nie sú riešiteľné pomocou žiadnych všeobecných pravidiel a ich riziko sme schopní iba mitigovať. V prípade, že nastanú, ich hľadanie vyžaduje vysokú úroveň seniority. Ide o problémy na prvý pohľad neviditeľné, dejúce sa na pozadí. Klasickým príkladom je vyťažovanie servera, ktoré je spôsobené zle optimalizovanou službou, kde ide o neefektívne získavanie dát, slučky, či náročné operácie vyžadujúce celú kapacitu procesora.

Problémy nemusia nutne ovplyvňovať aktuálny stav či oblasť úpravy, čo môže častokrát spôsobiť zmätok, či chybnú indíciu pri hľadaní týchto problémov. V týchto situáciách sa kapacita servera vyčerpá až časom, to následne spôsobí jeho pád, no vývojár si chybne myslí, že to spôsobil svojou úpravou on. Reálnym príkladom je export aplikácie v Procesnom Centre do zip formátu, ktorý vyžaduje výrazne väčší výpočetný výkon servera ako pri exporte bez kompresie len do čistého formátu twx. Počas tejto operácie je server vyťažený takmer na maximum a stačí, aby niektorý z vývojárov uložil náročnejšiu úpravu (CV, služba) v Procesnom Designeri a server spadne.

Okrem priameho zanesenia môžu byť chyby spôsobené aj implementáciou zmenových požiadavkov. Najčastejšie zmenové požiadavky sú zobrazené na diagrame 4.2.



Obr. 4.2: Kategorizácia zmenových požiadavkov a ich percentuálne zastúpenie

S vymenovanými kategóriami sa spájajú následujúce zmenové požiadavky:

- **Pridanie atribútu** - rozšírenie časti procesu, služby alebo komponenty (formulára). Pri zlej optimalizácii nutnosť vykonávať úpravu pre každú časť. Dobrý návrh umožňuje pridanie atributu jednorázovo v rámci všetkých použití.
- **Zmazanie, úprava atribútu** - ide o veľmi častý a najnebezpečnejší úkon. Je potrebné myslieť na to, že proces a jeho instancia, ktorá beží, atribúty používa v stave pred jeho spustením.
- **Úprava časti procesu** - v prípade zlej viditeľnosti a nejednoznačnosti pomerne náročný CR. Vývojár trávi dlhý čas nad zistením všetkých závislostí a je vysoké riziko zanesenia chyby.
- **Vytvorenie a zapojenie komponenty** - častokrát používané na viacerých miestach, preto je potrebné dopredu myslieť a pomponentu čo najlepšie prispôsobiť aby bola znovupoužiteľná.
- **Úprava, zmazanie komponenty** - tak ako pri každej úprave či mazaniu artefaktu, zvýšené riziko pádu na zle odstránených závislostiach.

Je potrebné si uvedomiť, že aj zlá analýza a návrh zmenového požiadavku môže byť pôvodcom často neočakávaných chýb. Ako z definície zmenového požiadavku vyplýva, ide o zmenu v systéme a teda v prípade, že je systém zle navrhnutý, sú zmeny často náročné a riziko zanesenia chýb sa výrazne zvyšuje. Typickým zástupcom takýchto požiadavkov sú úpravy stávajúcich častí procesu či komponent, kde je potrebné dobre zmapovať miesta použitia a rozsah vplyvu naprieč celou aplikáciou. Je potrebné myslieť na to, čo v prípade danej zmeny nemusí fungovať alebo ako úpravu vykonať so zachovaním funkčnosti v každej oblasti predošlých výskytov.

Existujú ale aj požiadavky, ktoré súčasný stav upravujú len minimálne a vo veľkej časti vytvárajú niečo nové. Pri týchto požiadavkoch je zanesenie chýb závislé od seniority vývojára bez ohľadu na súčasný stav kódu a návrh zvyšku aplikácie. To znamená, že zlý stav aplikácie môže byť v podaní skúseného vývojára obohatený kódom o úrovne lepším a časom môže slúžiť aj ako inšpirácia či referenčný bod, ako by to malo vyzerať.

4.4 Pôvod vzniku chýb

Pri riešení problematiky chýb, je ale na mieste otázka, prečo sa stále opakujú tie isté chyby, na rôznych projektoch, za predpokladu, že je použitá rovnaká technológia i dodávateľ? Pôvodcom môžu byť rôzne nedostatky ako sú financie, čo má za následok zrýchlený vývoj a menší dôraz na testovanie či refaktoring kódu, čo v konečnom dôsledku vedie k zdĺhavým opravám alebo náročným nadstavbám v prípade zmenových požiadavkov.

Častým problémom sú aj prísne deadliny - milníky dodania funkčných celkov. Ak sú stanovené s malými časovými rozostupmi, pričom sú v nepomere množstvo práce a aktívnych vývojárov, stáva sa, že dodávaný obsah nie je v prijateľnom stave z dôvodu dôkladnej implementácie a slabých, občas dokonca žiadnych testov. Je preto dôležité, aby manažment mal prehľad o náročnosti riešení, bol schopný to formulovať a následne zrozumiteľne prezentovať strane klienta, aby v závere obe strany dospeli k prijateľným kompromisom.

Napokon hlavným prepokladom na zvýšenú a opakujúcu sa chybovosť je úroveň seniority vývojárov. I napriek tomu, že je dodávateľ rovnaký, ľudia v nej a ich skúsenosti sa menia. Tí ktorí dosiahli vrchol špecializácie sa buď presúvajú na riadiace funkcie a strácajú tak prehľad o drobných implmentačných postupoch jednotlivých vývojárov alebo úplne menia špecializáciu, čo má za následok doplňovanie kapacít ľuďmi s menšou úrovňou seniority.

4.5 Zhrnutie kapitoly

V tejto kapitole som definoval, na základe štatistiky, najčastejšie problémy vzniknuté chybnou implementáciou či snahou o úpravu v rámci zmenových požiadavkov. Kvantita vzniku chýb či doba ich riešenia závisia na komplexnosti

projektu a seniority vývojárov. Častým javom je potom nejednotnosť riešení čo môže viesť k ďalším, viac závažnejším problémom, ktoré nemusia byť na prvý pohľad viditeľné.

Zozbieranie týchto informácií mi umožnilo v následujúcej kapitole vytvoriť sadu overených postupov a štandardov, ktoré vedú k zvýšeniu znalostí a zlepšeniu kvality produkovaného kódu.

Kapitola 5

Alternatívne materiály

V tejto kapitole som sa pokúsil porovnať niektoré z už existujúcich materiálov, ktoré riešia podobnú problematiku v rámci optimalizácie stavu aplikácií na platforme IBPM.

Informácie, ktoré som získal dokazujú, že tieto materiály síce problematiku čiastočne riešia, no žiadne nezodpovedajú požiadavkom úplne. Ich obsah následne rozoberiem.

5.1 IBM Redbook

Tento materiál by v sebe mal, podľa názvu [25] "Performance Tuning and Best Practices", zahrňovať všetko, čo potrebujeme. V tom prípade by nebolo potrebné, aby vznikal podobý materiál, keďže kniha je tvorená samotnou IBM a ľuďmi, ktorí sa na vývoji platformy BPM priamo podieľali.

Po dôkladnom prejdení sa obsah kapitoly 3 podobá konceptu, ktorý sa formuluje v rámci tejto práce. Nedostatkom je, že tam aj tento zámer končí a ďalej sa optimalizáciou a riešením chýb zaoberá len textovo a teoreticky bez praktických ukážok, čo je podľa mňa veľkým nedostatkom. Nachádza sa tam síce množstvo užitočných rád a konfigurácií, no menej znalý vývojár si z toho prakticky nič neodnesie. Je ale určite vhodné knihu používať ako doplňujúci materiál.

5.2 Kolban's Book on IBM BPM

V súčasnosti je [22] "Kolban's Book" jediný zdroj, ktorý v sebe centralizuje všetky potrebné informácie k získaniu základných znalostí potrebných pre prácu s IBPM. Optimalizáciu a najlepšie praktiky síce nerieši, a keď tak okrajovo, dáva ale všeobecný rozhľad čo technológia IBPM umožňuje a rady ako s ňou pracovať. Po osvojení si informácií obsiahnutých v tejto knihe, je vývojár prakticky pripravený s možnosťou využitia na reálnom projekte.

Zdroj v sebe obsahuje množstvo obrazového materiálu, čo len umocňuje rýchlosť pochopenia a použitia znalostí. Knihu považujem ako inšpiráciu po grafickej stránke zobrazenia obsahu, ktorej by som sa chcel držať v kapitole pri hľadaní riešenia.

5.3 Zhrnutie kapitoly

Výsledky hľadaní teda jednoznačne potvrdzujú, že materiály s podobným obsahom, sa v podobe, ktorá by združovala informácie a jednoznačne formulovala rady ohľadom optimalizácie, zatiaľ neboli vytvorené.

Kapitola 6

Sada základných doporučení pre zlepšenie vývoja

V rámci tejto kapitoly sa pokúsim o zhromaždenie pravidiel a praktických rád, ktorými by sa mal vývojár pri produkovaní dobre udržateľného kódu držať, prípadne vzorových situácií, ktorým by sa mal naopak vyhýbať. Tieto poznatky sú získané od ľudí, ktorí ich získali dlhoročnou praxou v oblasti vývoja procesných aplikácií.

Predmetom skúmania, kde sa ďalej pokúsim pravidlá aplikovať bude Systém záverečných prác (SZP). Je to webová aplikácia vyvinutá skupinou CZM (Centrum znalostného manažmentu) patriacou pod FEL ČVUT pre účely používania FIT ČVUT. Ako názov napovedá, systém slúži k práci so záverečnými prácami a to spôsobom, že v sebe zahrňuje celý proces od jej vytvorenia - zadania do systému, rezervácie študentami až po schvaľovanie príslušníkmi vedenia fakulty, odovzdávania, oponentúry a výsledného hodnotenia.

Postupy sú členené na základe vrstiev, ktoré v sebe zapúzdrujú rôzne pohľady a funkcionalitu a sú to tieto tri:

- Procesná vrstva - definuje logiku a proces, ktorým sa aplikácia riadi a na jeho základe postupuje. Prvky sú na ňom definované formou BPMN a umožňujú modelovanie požadovaného správania na najvyššej (procesnej) úrovni.

- Frontendová vrstva - definuje komponenty, ktoré su viditeľné užívateľmi aplikácie. Nachádzajú sa v nej Coach Views (CV), ktoré sú buď súčasťou ďalších, väčších CV alebo vystupujú samostatne.

- Servisná vrstva - nesie v sebe backendovú logiku a všetko s ňou spojené. Je modelovaná taktiež formou BPMN, no komponenty sú viac orientované na spracovávanie dát, komunikáciu s databázou či inými externými zdrojmi.

Medzi najčastejšie problémy, ktoré sa vyskytujú naprieč všetkými vrstvami a je potrebné ich riešiť patria tieto:

- Redukovať v procese množstvo použitých artefaktov, ktoré v sebe nesú volania databáze. Niekoľko násobné volanie v rámci jednej systémovej služby je zbytočné a zhoršuje odozvu i rýchlosť systému. Je dôležité zvážiť, či pridanie ďalšej krabičky – volania má nejaký význam a nie je lepšie vykonať zmeny rovno v integračnej službe. Takto vytvorené služby je potrebné optimalizovať.

- Málo zdokumentovaný kód, či už ide o premenné, funkcie alebo rovno celé bloky. Budúci vývojár tak zbytočne stráca čas, ktorý potrebuje k pochopeniu kódu či zdĺhavému zisťovaniu účelu už implementovanej funkcionality. Tento problém by ale vôbec nemal nastávať pretože publikácia s názvom Clean Code [26] jasne hovorí o tom, "*že kód, ktorý potrebuje byť dokumentovaný, je zlý kód. Všetky premenné či názvy funkcií majú teda presne pomenúvať to čo vykonávajú, prípadne akú oblasť ovplyvňujú.*"

- Pred tvorením novej časti, či už je to premenná, funkcia alebo rovno služba je potrebné najprv dôkladne analyzovať, aby nevznikali zbytočné duplicity, ktoré taktiež sťažujú orientáciu, prípadne optimalizáciu. Preto je potrebné zjednotiť pomenovania a poradie premenných.

- Naopak pri odstraňovaní existujúcich častí či už sú to premenné, funkcie až zložité časti kódu, platí dôležitosť analýzy dvojnásobne. Niekedy odstránenie navonok nepotrebnej premennej môže mať obrovský dopad na funkcionalitu aplikácie. Z praxe viem, že logické chyby sa hľadajú najhoršie, keďže k ich nájdeniu a odstráneniu je potrebné kódu a jeho štruktúre dobre rozumieť, čo je v prípade človeka, ktorému nie je daný kód vlastný, veľmi časovo náročný problém. Najlepšou radou je nepremenovávať ani nemazať nič pri čom si nie som istý účelom. Problém sa odporúča poznačiť a riešiť až v prípade väčších zmien v štruktúre, tzv. refaktoringu.

- Zhlukovanie kódu už bolo spomenuté v časti o duplicitách a opätovných volaniach rovnakých služieb s inými parametrami. V rámci tejto problematiky dávam do pozornosti zjednotenie v rámci inicializácie, kde sa často stáva, že objekt či premenná sa inicializuje až v časti, kde sa používa - volá. Ak je ale premenná volaná niekde v kóde pred, nastáva problém, kde siahame na miesto, ktoré nie je definované, čo spôsobí chybu až pád aplikácie. Najlepším riešením je inicializácie centralizovať v rámci jedného inicializačného objektu, ktorý voláme vždy ako prvý, pred použitím akejkoľvek inej funkcionality.

V následujúcej časti je pre každý problém definovaná sekcia s pravidlom, ktoré ho pomáha riešiť. Má vždy jednotnú štruktúru začinájúcu úvodom, v

ktorom je najprv všeobecné popísaná vzniknutá situácia. Po nej následuje definícia a názorná ukážka problému na ktorú naviažem náčrtom jeho riešenia. Pokračujem aplikáciou pravidla, aby čitateľ získal predstavu ako pravidlo použiť. V závere každého pravidla zhodnotím jeho prínos a priblížim jeho využitie vo všeobecnosti.

6.1 Procesná vrstva

Optimalizácia v rámci procesnej vrstvy spočívala v úplnom pochopení fungovania celkovej aplikácie, jednotlivých stavov či vetvení, pretože každá, zdanlivo drobná, zmena na tejto úrovni má značný vplyv na funkcionalitu celej aplikácie. Celá logika je definovaná na najvyššej úrovni definície procesu, ktorá je popísaná v časti 3.2.0.1. Užívateľ sa vplyvom chyby môže dostať do stavu, ktorý preňho nie je korektne definovaný (aplikácia napríklad očakáva dáta, ktoré daný objekt neobsahuje) a teda spôsobí nechcený pád, v lepšom prípade zlú reakciu.

Chyby tejto vrstvy sú v rámci štatistík 4.3 zastúpené v kategóriách Chýbajúca funkcionalita, Nesprávny stav procesu, Chýbajúci stav procesu a Nesprávne role.

6.1.1 Logovanie a odchytávanie chybových stavov

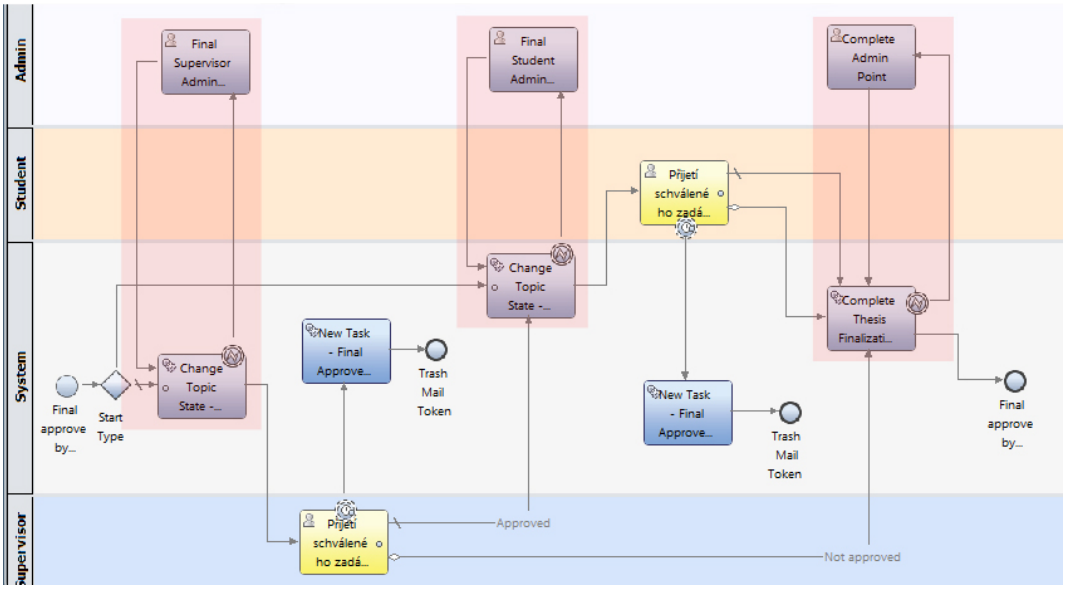
Logy sú jediným zdrojom informácií, ktorými vývojár získava prehľad o celkovej činnosti užívateľov v aplikácií. Je to zdroj, ktorý poskytuje informácie o udalosti vykonávanej pred pádom či neočakávaným výstupom. Preto by sa k tejto činnosti malo pristupovať zodpovedne a každý celok či komponentu, pri ktorej hrozí riziko pádu alebo dôkladne ošetriť a zmapovať zavolaním logu a pridaním odchytávania výnimky.

6.1.1.1 Problém

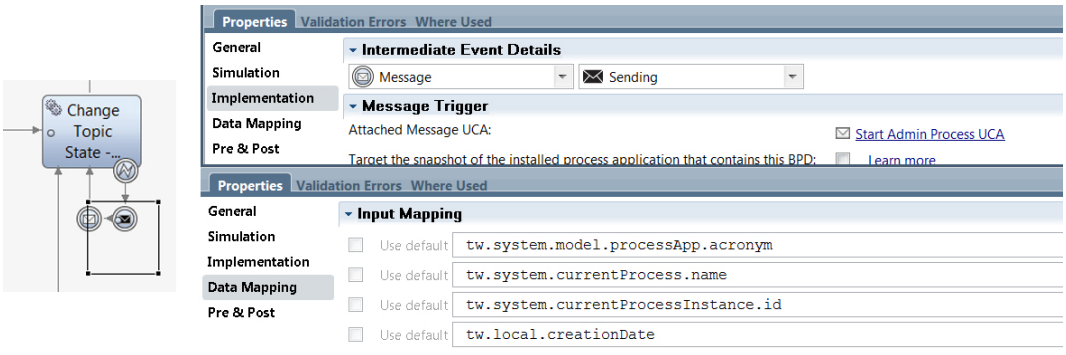
Na obrázku 6.1 na prvý pohľad vidíme neprahľadnú štruktúru a rozmiestnenie procesov. Značne k tom prispieva i “error a exception handling“, ktorý je realizovaný odvedením procesu do časti, na obrázku 6.1 pomenovanej, “Admin Point“.

6.1.1.2 Riešenie

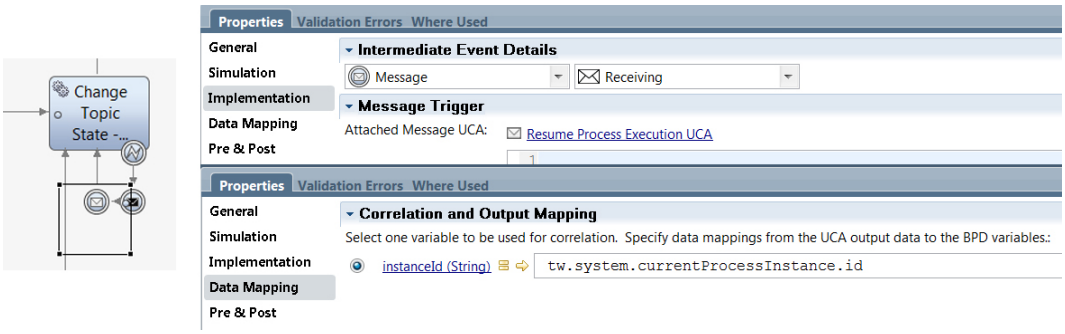
V prvom rade zadefinujem používaný pojem Undercover Agent (UCA), čo sú podľa Kolbana [22] komponenty v IBPM, ktoré naslúchajú na udalosti založené na plánovanom čase. Sú takisto zodpovedné za spúšťanie procesných inštancií, resp. reakciu na prichádzajúce požiadavky, prostredníctvom správ odosielaných inou časťou či procesom samotným. Spracovávanie chýb sa odporúča riešiť cez komponenty nazývané “Intermediate Eventy“ zapojením tak



Obr. 6.1: Nadbytočné komponenty procesu s názvom “Admin Point“



Obr. 6.2: Pri chybovom stave sa odošle správa (formou UCA) na administrátora, kde sa mu vytvorí v procesnom portále nová úloha



Obr. 6.3: Po spracovaní, náhľade chyby a stlačení tlačítka sa odošle správa, na ktorú reaguje príslušný event

ako vidíte na obrázku 6.2. Takéto zapojenie sa vo vývojárskom slangu nazýva "koloběžka". Ďalej sa odporúča vytvorenie samostatnej aplikácie s názvom napríklad "Admin Application", obsahujúca proces, ktorý sa daným chybovým eventom napojenom na komponente spúšťa. Teda UCA odoslaným aplikáciou, v ktorej chybový stav nastal. Štart procesu je vlastne reakcia na prichádzajúce správy typu chyba. Použitie a konfigurácia vstupných premenných je zrejmá z následujúcich obrázkov 6.2 6.3.

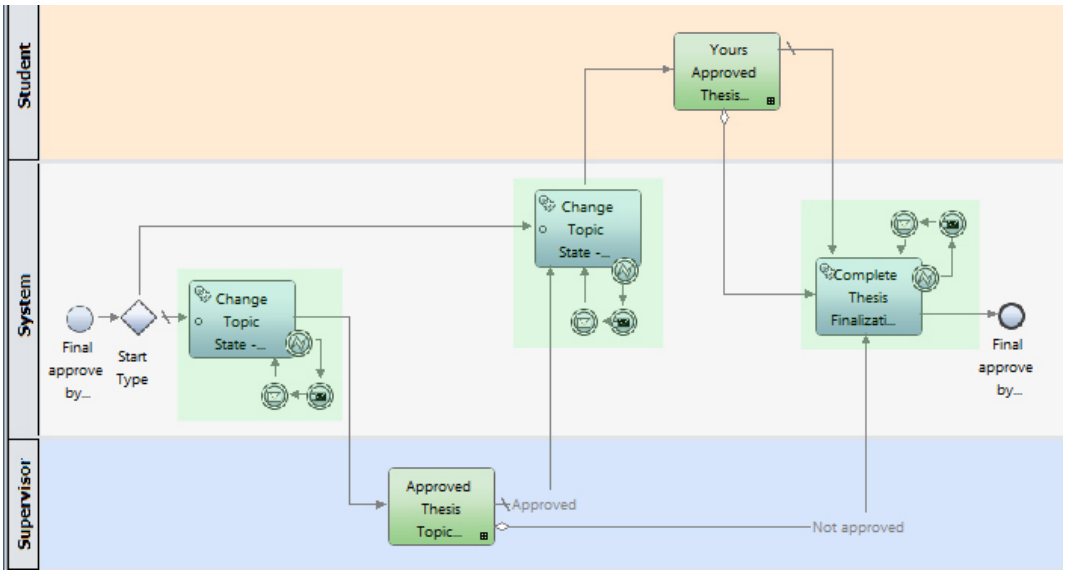
V administrátorskej aplikácii je možné vytvoriť pre správu prakticky akúkoľvek užívateľsky prívetivú aplikáciu. Postačí ale jednoduchý Coach, ktorý zobrazuje informácie o chybe a nejaký element (tlačítko), ktorým užívateľ presunie proces zo stavu "pozastavený" (nie je možné s ním pracovať) späť do stavu "aktívny" (pôvodný stav, pred tým ako nastala chyba) v prípade, že je všetko v poriadku. Prístup k takémuto tasku má spravidla formou procesného portálu úžívateľ s aministrátorskými právami.

6.1.1.3 Prínos pravidla

Na obrázku 6.4 je na prvý pohľad evidentné, že hlavným prínosom je zvýšenie prehľadnosti procesu. Ďalším zlepšením je aj odstránenie, teraz už nadbytočnej administrátorskej swimline, ktorú v sebe už obsahuje aplikácia, na ktorú je príslušné UCA odosielané. V rámci hlavného procesu ju nemusíme vytvárať a o všetko sa už externá eplikácia postará za nás.

6.1.1.4 Zhrnutie vo všeobecnosti

Takýto systém spracovania je vhodný a aplikovateľný na akúkoľvek aplikáciu. Stačí len vytvoriť všetky potrebné celky a naviazať na časť procesu, kde sa chyba môže vyskytovať. Takáto funkcionalita často nebýva súčasťou aplikácie ale je používaná formou toolkitu.



Obr. 6.4: Výsledná komponenta po aplikácií systému odchytávanie výnimiek formou "koloběžky"

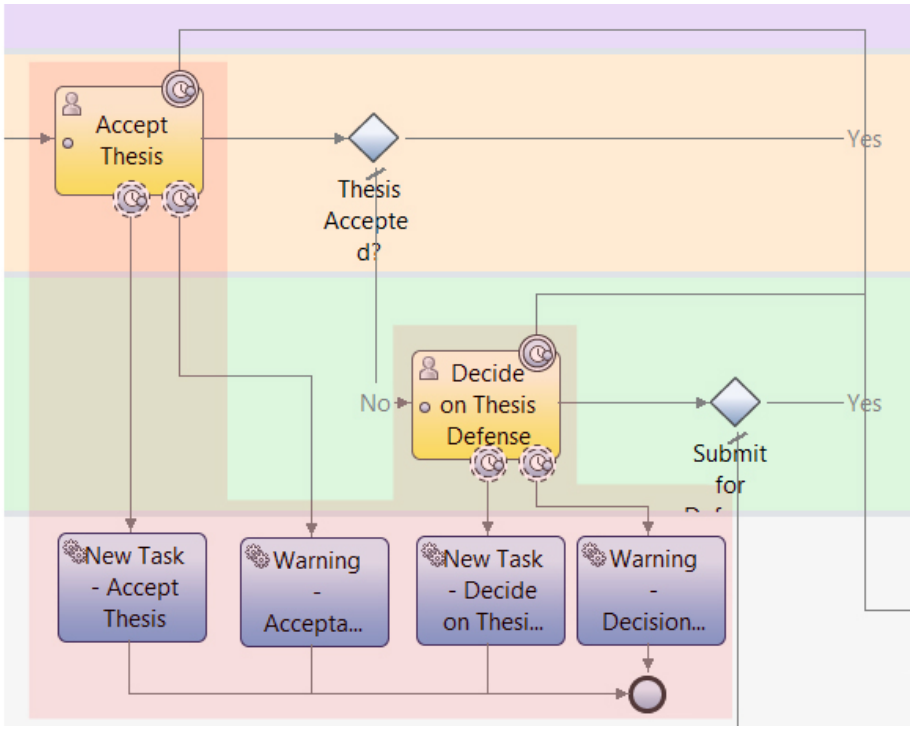
6.1.2 Členenie logických celkov do subprocesov

Jedným z dôležitých úkonov, ktoré prispievajú k lepšiemu prehľadu v procesoch je zoskupovanie súvisiacich častí do subprocesov. Pri takýchto úpravách procesu je dôležité si uvedomiť, ktoré časti sú si príbuzné a majú medzi sebou súvis.

Ak sa toto kritérium nedodrží, tak vývojár môže mylne a zdĺhavo, napokon neúspešne, hľadať konkrétnu časť či funkčný celok procesu, čo zbytočne navyšuje čas potrebný k rozvoju a teda tento problém uberá na účinku samotnej optimalizácie z pohľadu rozšíriteľnosti. Dôležitosť riešenia tohoto problému taktiež potvrdzujú šatistiky predchádzajúcej kapitoly 4.3, kde predstavujú podstatnú časť zo všetkých problémov ako sú *nesprávny a chýbajúci stav procesu.*

6.1.2.1 Problém

Problém vzniknutý v tejto časti je zrejmý z obrázka 6.5, kde vidíme zbytočnosť zobrazenia funkcionality a volaní služieb. Je to názorný príklad, ako by to nemalo vyzerať a na konci i vidieť ako výrazne sa zlepší prehľadnosť procesu aplikáciou pravidla členenia do subprocesov, viď obrázok 6.7.



Obr. 6.5: Pôvodný stav procesu a jeho zbytočne viditeľné časti

6.1.2.2 Riešenie

Vytvorením podprocesu, do ktorého danú funckionalitu 6.6 zapúzdrime a správame sa k nej ako k samostatnému "mini procesu", čo je vidieť na obrázku 6.7

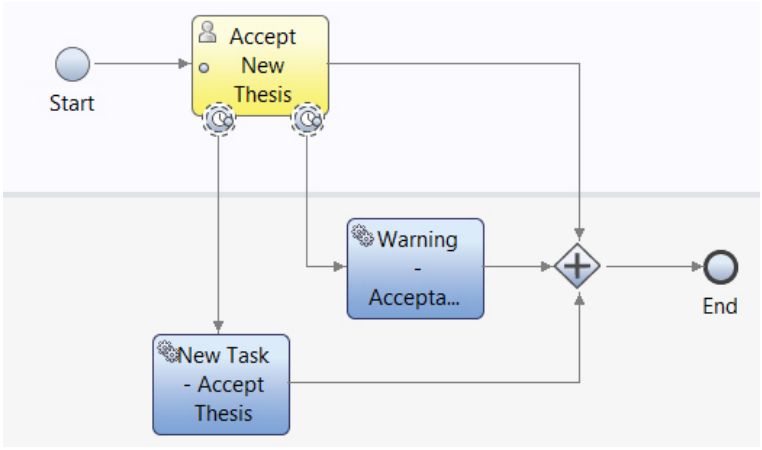
6.1.2.3 Prínos pravidla

Ak je princíp aplikovaný na menšie logické celky, ktoré nie sú nevyhnutne potrebné k pochopeniu hlavného toku procesu, ich "skrytím" a zoskúpením pod komponentu subprocesu výrazne zvyšujeme jeho prehľadnosť a čitateľnosť.

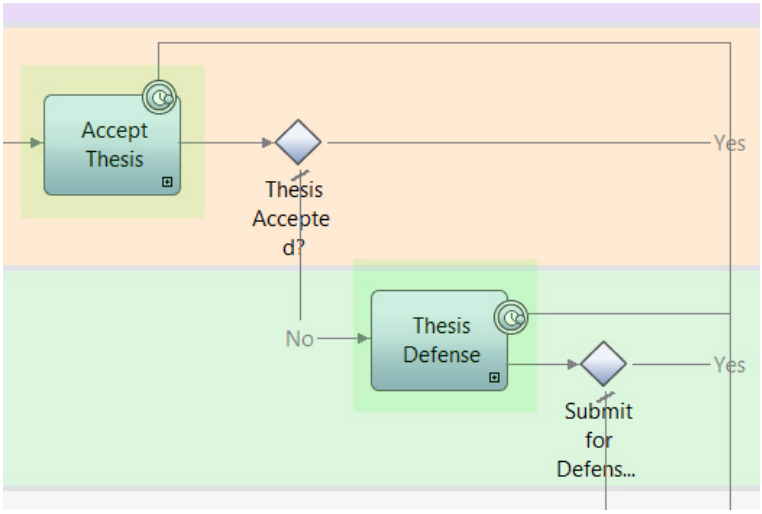
V prípade potreby získania detailnejšej informácie o fungovaní danej časti je možné subproces jednoducho otvoriť a pracovať tak ako v prípade jeho viditeľnosti v rámci hlavného procesu.

6.1.2.4 Zhrnutie vo všeobecnosti

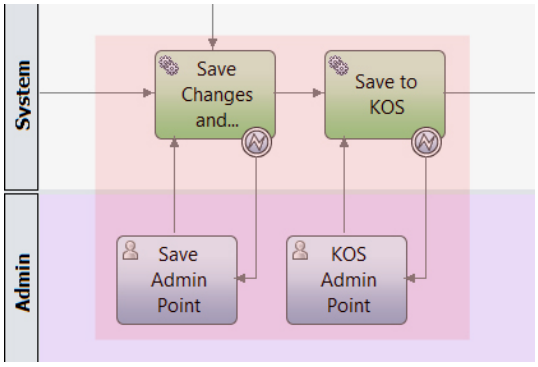
Zmena aplikovateľná na akékoľvek logické celky procesu, ktoré priamo nesúvisia s jeho celkovým vnímaním a zároveň sú medzi sebou logicky previazané. Ich význam je zahrnutý formou súhrnného pomenovania komponenty subprocesu, takže v prípade potreby sú jednoducho dohľadateľné.



Obr. 6.6: Časť procesu presunutá pod samostatný subproces



Obr. 6.7: Výsledný stav zjednodušeného procesu



Obr. 6.8: Dve servisné komponenty za sebou a navyše bez “koloběžky“

6.1.3 Servisné volania

Ide o bežne využívanú komponentu, ak potrebujeme získať alebo spracovať dáta v rámci procesu.

6.1.3.1 Problém

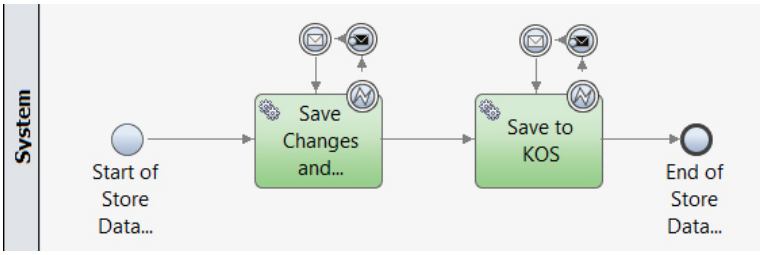
Častý problém je, že týchto volaní je zbytočne mnoho a týmto znižujú prehľad a orientáciu v rámci celého procesu. 6.8

6.1.3.2 Riešenie

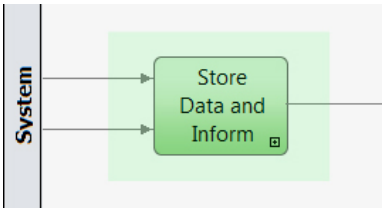
Pri riešení si položíme otázku, či je možné nahradzovať alebo iba zoskupovať. V prípade, že komponenty riešia príbuzné veci a ich využitie sa v iných procesoch neopakuje, je na nás ku ktorej z možností sa prikloníme. Spravidla sa vytorí jedna väčšia vrstva a proces sa zjednoduší. V opačnom prípade, ak komponenty riešia dve nezávislé veci je preferovaná možnosť zoskupovania pod subproces. Ďalej ale nastáva otázka, či je možné naviazať error handling rovno na celú komponentu subprocesu. V tomto prípade nie, pretože model jasne vyžaduje odchytávanie chybových stavov na konkrétnej komponente 6.9, na ktorej nastal a teda naviazaním o "vrstvu vyššie"by nebolo hneď jasné v ktorej z dvoch servisných komponent chyba nastala.

6.1.3.3 Aplikácia pravidla

Výsledkom je znova súhrnne pomenovaný subproces, ktorý v sebe zoskupuje ukladanie dát a notifikáciu.6.10



Obr. 6.9: Výsledná podoba servisných komponent presunutá do samostatného subprocesu



Obr. 6.10: Finálna podoba časti procesu formou subprocesnej komponenty

6.1.3.4 Zhrnutie vo všeobecnosti

Otázka sa môže klásť pri akejkoľvek situácii, ktorá obsahuje viacero servisných komponent, no až odpovede na nich nám naznačia do akej miery je možné túto metódu použiť a či vôbec.

6.1.4 Konvencie premenných a konštánt

Na prvý pohľad banálna záležitosť vnímaná ako samozrejmosť, no prax ukazuje ako často sa táto konvencia nedodržiava. Tento úkon nemá žiaden vplyv na výkon či rýchlosť aplikácie, no značne ovplyvňuje efektivitu vývoja či znižuje riziko zanesenia chyby a to v prípade, že je potrebné hľadať niektorú z premenných, meniť jej názov alebo použitie, a jej výskyt je skrytý medzi ďalšími objektami či dvadsiatkou iných premenných.

6.1.4.1 Problém

V tejto oblasti je riešených hneď niekoľko problémov. Sú to nezoradené zoznamy premenných v rámci objektov, CV alebo služieb. V aplikácii Záverečné práce sa nezoradený zoznam premenných 6.11 nachádzal prakticky všade, kde sa dajú definovať premenné a teda celkové dohľadávanie bolo časovo náročné. Ďalším problémom sú ich názvy, ktoré nie sú jednotné. Napokon je rozporu-

plné ich použitie a nutnosť ich zavádzania, kde je možná náhrada konštantami a teda zjednodušenie ich zoznamu.

6.1.4.2 Riešenie

Riešenie pozostáva z niekoľkých krokov. Najprv by malo logicky následovať ich premenovávanie, na základe konvencií. Táto akcia sa ale u bežiacich procesov striktne nedoporučuje, pretože to môže ovplyvniť všetky súčasne bežiace instancie. Je teda potrebné najprv všetky staré procesy nechať dobehnúť. Ak sa chceme vyhnúť tomuto problému je potrebné si každé zavedenie nového atribútu dobre rozmyslieť a pomenovať podľa toho, čo naozaj predstavuje. Robert C. Martin v knihe Clean Code píše, že [26] *"Každé pomenovanie atribútu si vyžaduje pozornosť ako by sme vymýšľali meno pre svoje dieťa."* Ak je takáto zmena predsalen nevyhnutná, bežnou praxou pri písaní zložených názvov premenných alebo fráz je princíp Camel Case, kde každé slovo alebo skratka v strede frázy začína veľkým písmenom, bez medzier alebo interpunkcie. Ďalším, pomerne jednoduchým, krokom je zoradiť ich. Bežným radiacim kritériom je abeceda, kde premenné sú občas pomenované i prefixami na základe čoho je možné i radenie do skupín podľa príslušnosti, a je teda na mieste zvážiť využitie ďalších objektov združujúcich skupiny atribútov.

Mnohokrát sa stáva, že atribút pomenúva hodnotu, ktorá je nemenná a použitá na niekoľkých miestach. Pri tejto situácii je na zváženie, či je atribut na tomto mieste vôbec potrebný a umiesniť ho medzi konštanty. V IBPM je možné konštanty definovať viacerými spôsobmi. Často používaným je definícia v záložke *Behavior* v rámci daného CV 6.12. Odporúčaný postup je ale umiestnenie atribútov do externého JavaScript súboru, kde je poskytovaný formou get funkcie 6.13.

6.1.4.3 Aplikácia pravidla

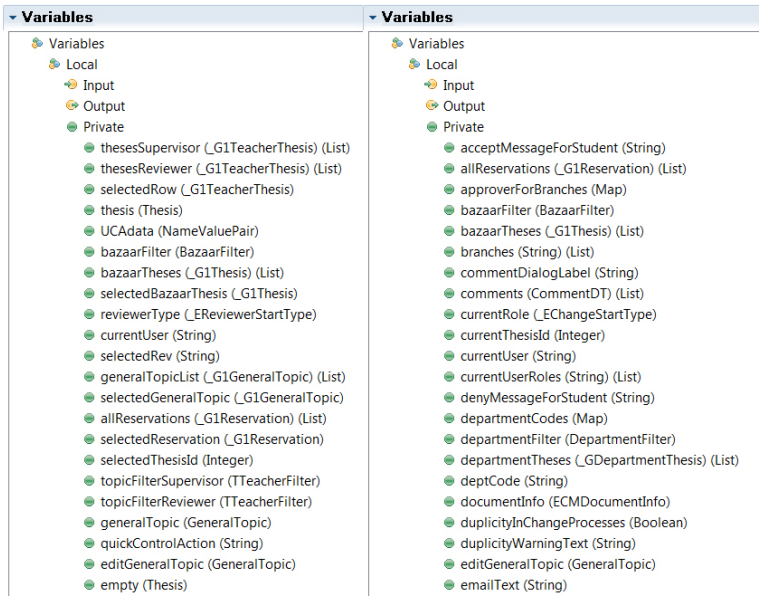
Dobre pomenovaný a zoradený zoznam 6.11 dáva lepší prehľad o tom, či daný atribut existuje a aký plní účel. Znižuje sa tak šanca zavedenia duplicitných či nadbytočných atribútov.

6.1.4.4 Zhrnutie vo všeobecnosti

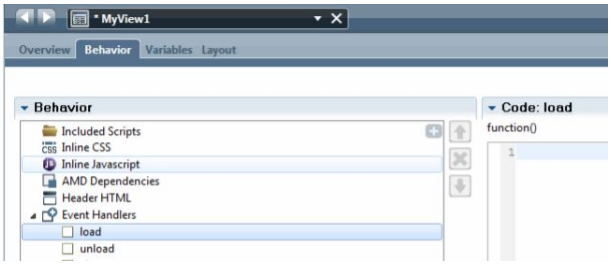
Do problémového stavu, riešenom v tejto časti by sa vyvíjaná aplikácia nikdy nemala dostať, preto je nevyhnutné si tento návyk osvojiť a striktne dodržiavať už od začiatku pred spustením prvej instancie procesu na produkčnom prostredí.

6.1.5 Zhrnutie optimalizácií

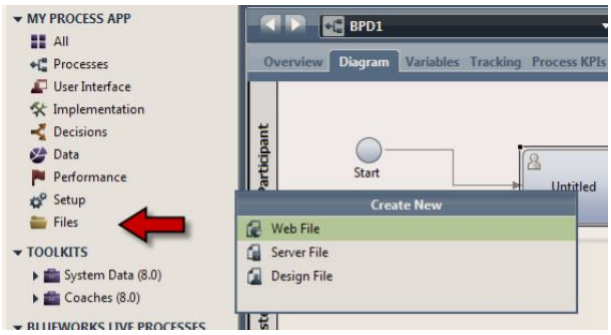
Optimalizácie v rámci procesnej vrstvy výrazne zpríjemňujú prácu pre vývojárov a to tým, že znižujú čas pri hľadaní chýb alebo implementácií rozšírení



Obr. 6.11: Zoznam premenných pred a po aplikácií kritérií



Obr. 6.12: Miesto v rámci CV, kde je možné písať vlastný JavaScript kód



Obr. 6.13: Pridávanie externého JavaScript súboru

naprieč procesom. Zlepšujú tak aplikáciu po stránke vývoja a rozšíriteľnosti. Pri zmenových požiadavkoch podľa prieskumov nepokrývajú veľké množstvo prípadov (Úprava časti procesu), no je to z dôvodu, že sa odporúča navrhnutý proces meniť minimálne. Každá výrazná zmena totiž prináša množstvo problémov spojených s migráciou instancií. Ak takýto požiadavok predsa len nastane, vývojár sa rýchlo zorientuje, prípadne i klient jednoduchšie pochopí problematiku či možnosť realizácie požiadavku. Sprehľadnením procesov je dosiahnuté zníženie chybovosti v rámci nesprávnych či dokonca chýbajúcich stavov procesu.

6.2 Frontendová vrstva

Optimalizácie na úrovni fontendovej vrstvy v sebe nesú úpravy v rozložení jednotlivých častí obrazoviek, kde sú volané alebo použité. Obrazoky sú zložené z menších častí, zvaných CV, ktoré boli predstavené v časti 3.2.0.1. Táto vrstva má teda na svedomí zdĺhavé (príp. viacnásobné) vykresľovanie jednotlivých komponent či dlho trvajúce prechody medzi obrazovkami.

6.2.1 Členenie na Coach Views

Ďalším neodmysliteľným krokom k dosiahnutiu dobre udržiavaného a prehľadného kódu je členenie jednotlivých frontendových častí do znovupoužiteľných Coach Views. Princíp spočíva najprv v indentifikácii komponenty, ktorá je používaná na viacerých miestach. Pre tento účel sa odporúča vytvorenie komplexného pohľadu formou diagramu kde sa dajú opakované miesta jednoduchšie vyhľadávať. Takto identifikované celky vytvorím ako znovupoužiteľné CV, ktoré v sebe budú zapúzdrovať všetkú potrebnú funkcionalitu a reagovať na zmeny na základe vstupných parametrov ako je napríklad viditeľnosť tlačidiel či volanie iných služieb v závislosti na tom, pod akou rolou je daný CV používaný. Tento princíp v sebe nesie zmysel samotného DI, definovanom v sekcii o navrhových vzoroch. Ja si v CV definujem parametre, ktoré potrebujem a ten, kto ho potrebuje použiť musí zabezpečiť aby boli korektne predané. Vytvorený CV zapojím tak, že parametre vstupujúce do pôvodných komponent predám cez binding a konfiguračné nastavenia novovzniknutému CV.

Aplikácia záverečných prác spomínané členenie na CV obsahuje len zriedkakedy čo značne komplikuje vykonávanie akejkoľvek dodatočnej zmeny v aplikácii bez zvýšeného rizika zanesenia chýb.

Proces aplikácie tejto zmeny bol pre mňa, ako človeka bez hlbších implementačných znalostí v rámci tohto projektu, pomerne komplikovaný. Nemal som dostatočné znalosti o fungovaní aplikácie ako celku, či vzájomných závislostí jednotlivých častí. Preto som prechodom všetkými modulmi vytvoril screenshoty obrazoviek, predstava akéhosi diagramu, a takéto miesta označil. Pre lepšiu predstavu je diagram dostupný v prílohe D. Tieto obrazce som zoradzoval tak ako po sebe následovali v aplikácii, následne zoskupoval podľa

Human Services a napokon ohraničil a označil ako jeden modul. Na takomto komplexnom pohľade som postupne nachádzal podobné či rovnaké časti, pridával im tagy, a podľa toho vytvoril Coach view, ktorým som mohol všetky tieto časti nahradiť. CVs som podľa výskytov triedil do kategórii TK alebo vlastný.

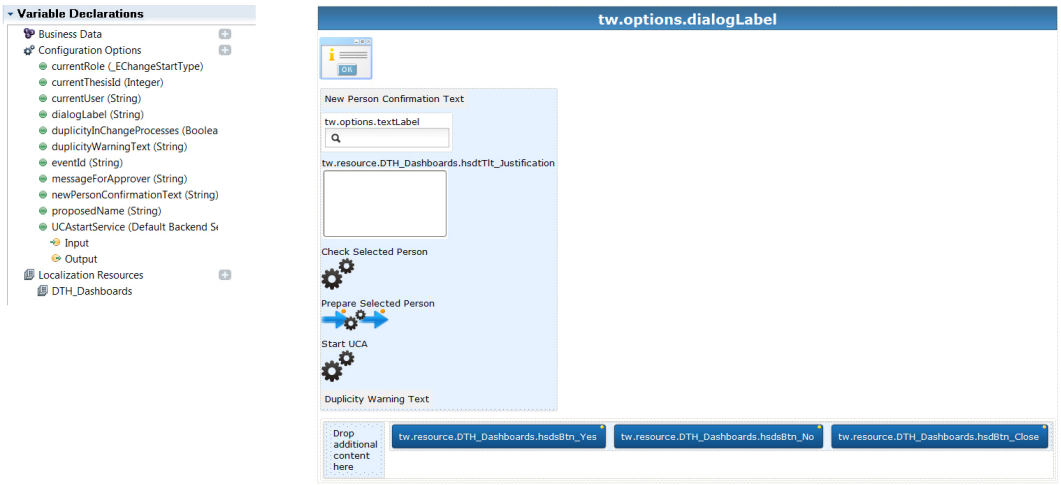
TK CVs boli také, ktoré mali výskyt i mimo vlastnú aplikáciu. Tieto typy som definoval, ako z označenia vyplýva, v Tookitoch ktoré boli všobecne použité vo všetkých moduloch. Bolo to z dôvodu, že takéto komponenty boli využívané vo viacerých moduloch, preto by bolo potrebné ich definovať pre každý modul zvlášť. Takto postačilo iba volanie prostredníctvom Toolkitu, ktorý bol pre všetky moduly spoločný. V prípade, že šlo o vlastné, a teda ich výskyt iba v rámci príslušného modulu, bolo postačujúce definovanie iba v tom danom module.

6.2.1.1 Problém

Kód dialogového okna, ktoré vidíte na obrázku 6.14 sa v rámci aplikácie nachádzal tri krát. V prípade zmien, či rozvrhnutia prvkov bolo potrebné upravovať každú časť zvlášť, čo zvyšovalo šancu zanesenia chyby pri úpravách.

6.2.1.2 Riešenie

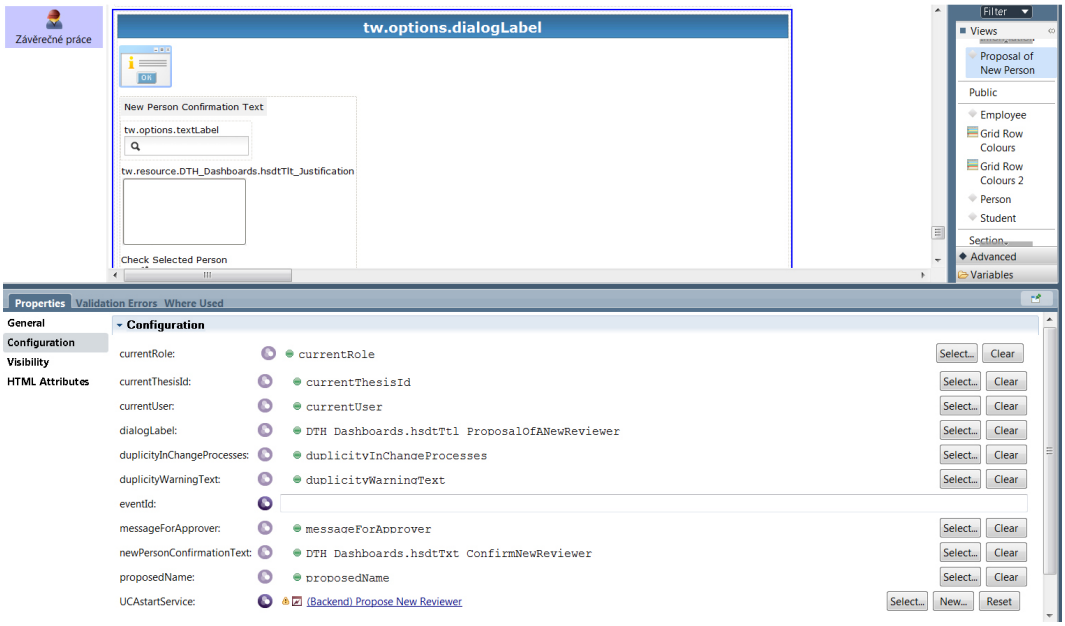
Vytvorené Coach View, ktoré vidíte na obrázku v sebe nesie všetky časti pôvodného okna a navyše je parametrizované tak, že sa dá použiť všeobecne pre všetky tri prípady.



Obr. 6.14: CV konštruované na základe niekoľkonásobne používanej časti

6.2.1.3 Aplikácia pravidla

Na následujúcom obrázku 6.15 je viditeľné použitie znovupoužiteľného CV, ktoré sa síce na Human Servise nachádza stále tri krát, no vždy sú mu predávané iné parametre v závislosti na tom, pre akú rolu sa má vykresliť. Teda v tomto prípade ide o premennú `currentRole`, ktorá v sebe nesie príslušnú rolu, na základe ktorej sa v CV zobrazujú príslušné dáta a tlačidlá. Za povšimnutie stojí i parameter `UCAstartService`, ktorého vstupom je rovno služba, ktorá zabezpečuje otvorenie korektného okna či odoslanie dát na spracovanie službou pripravenou pre danú rolu. Všetky parametre sú predávané v záložke "configurations" formou ako je demonštrované na obrázku 6.15.



Obr. 6.15: Zapojenie a predanie vstupných parametrov vytvoreného CV

6.2.1.4 Zhrnutie vo všeobecnosti

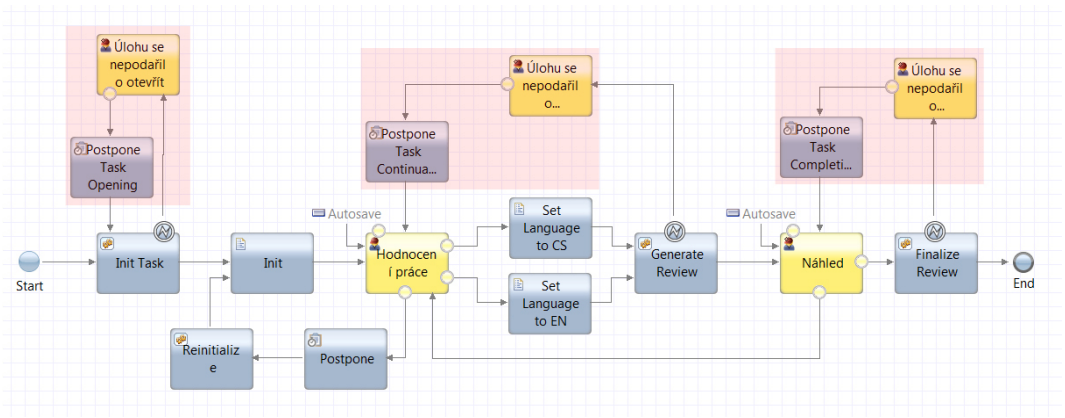
Táto operácia slúži len ako názorný príklad vytvorenia znovupoužiteľného CV. Existuje nekonečne mnoho možností kde sa dá tento princíp aplikovať, no podstatné je uvedomiť si, že takáto možnosť exisuje a je potrebné ju používať, čim sa nám výrazne zjednodušuje rozšíriteľnosť či udržateľnosť jednotlivých komponent aplikácie.

6.2.2 Členenie kódu do Toolkitov

Táto rada môže byť chápaná ako nadstavba na predchádzajúcu optimalizáciu. A teda ak máme screeny rozdelené do menších Coach Views, je dobrým zvykom, v prípade použitia naprieč viacerými aplikáciami, takýmto častiam vytvoriť vlastný Toolkit. Je dôležité si uvedomiť, že tak ako sú znovupoužiteľné Coach Views, to isté platí i pre služby, objekty či iné komponenty IBM BPM.

6.2.2.1 Problém

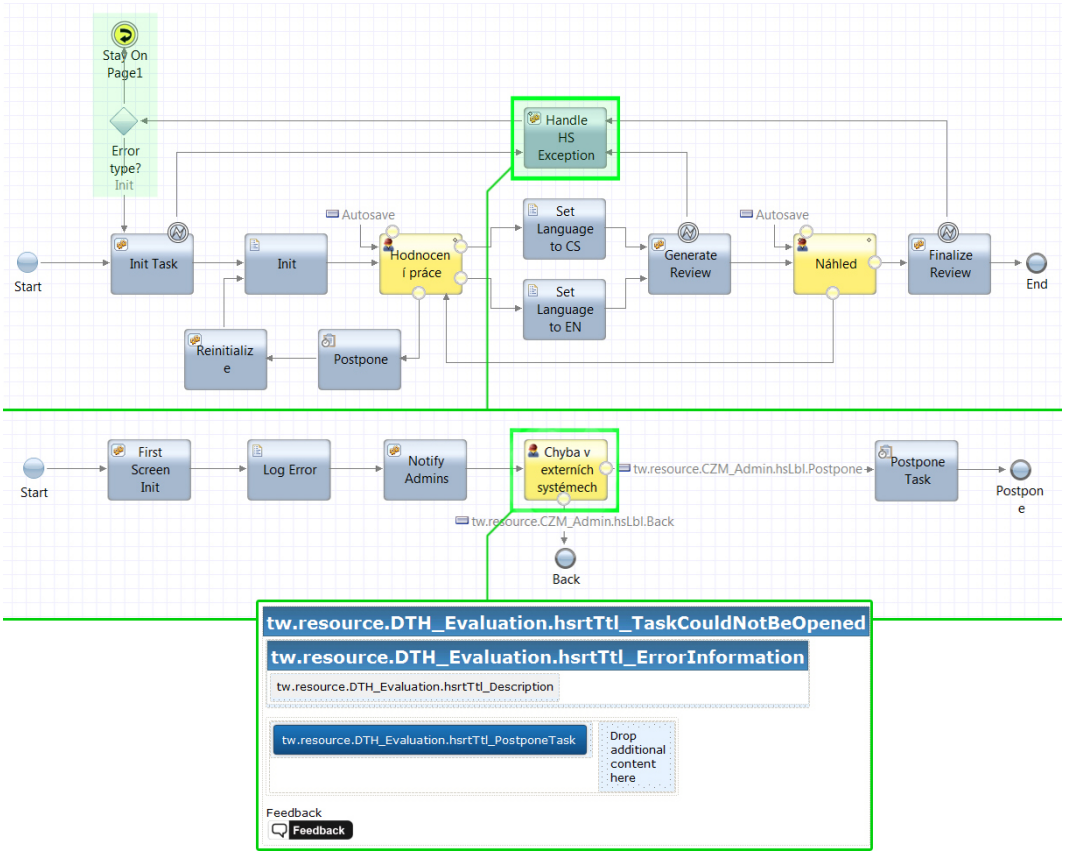
Ako demonštratívny príklad bol v aplikácii záverečné práce nájdený celok 6.16, ktorý bol opätovne implementovaný 14 krát. Je pochopiteľné, že vykonať akúkoľvek zmenu, naprieč všetkými jeho použitiami vo všetkych aplikáciach je časovo náročné s vysokou šancou zanesenia chyby.



Obr. 6.16: Flow zapojených Coaches obsahujúcich chybové hlášky

6.2.2.2 Riešenie

Implementácia univerzálnej Human Service obsahujúcej Coach View, ktorá je parametrizovaná a zapúzdruje všetku logiku je v tomto prípade žiadúca. Je vhodné taktiež aplikovať pravidlo, že ak je používaná naprieč viacerými aplikáciami, jej implementáciu vložíme do Toolkitu. Jej zapojenie 6.17 je demonštrované na následujúcom obrázku.



Obr. 6.17: Centralizované zapojenie Human Servisy nesúcej funkcionalitu pôvodného Coacha, služba z vnútra, náhľad Coach View

6.2.2.3 Aplikácia pravidla

Rozdelením podobných komponent do CV a niektorých i do TK sa programová časť aplikácie stala prehľadná, omnoho ľahšie rozšíriteľná a hlavne znovupoužiteľná. V prípade zmien v týchto komponentách nie je potrebné zmeny aplikovať niekoľkonásobne naprieč všetkými použitiami a teda zbytočne zvyšovať riziko chyby.

6.2.2.4 Zhrnutie vo všeobecnosti

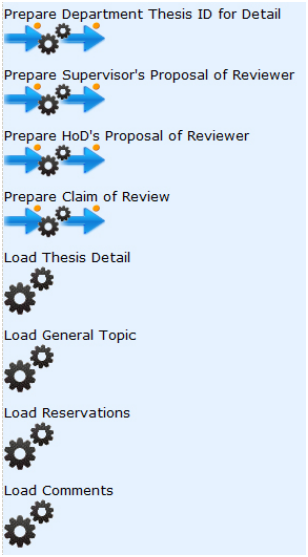
Vo všeobecnosti platí, že vždy je lepšie každú komponentu vytvárať ako CV, pretože nikdy nevieme, kedy sa nám v budúcnosti môže hodiť. Ak nebude zodpovedať naším požiadavkom, tak sa jednoducho vytvorí nová a nič sa tým nestratí.

6.2.3 Volanie AJAXových služieb

Pri začiatkoch vývoja aplikácie ZP technológia neumožňovala iné typy volaní, takže v prípade handlingu rôznych požiadaviek bolo potrebné vytvoriť väčšie množstvo naslúchajúcich "Intermediate eventov".

6.2.3.1 Problém

Pri spúšťaní jednotlivých CV, BPM potrebuje inicializovať všetky udalosti, ktoré sa môžu niekedy počas behu daného CV volať. Teda v tomto prípade so spustením obrazovky inicializuje osem slotov 6.18, ktoré sú pripravené reagovať na nejakú požiadavku. Tieto operácie sú drahé z pohľadu pamäte, ktorá je zbytočne zaťažovaná a nie je vôbec isté, či inicializované udalosti budú vôbec zavolané.



Obr. 6.18: Súbor inicializovaných udalostí

6.2.3.2 Riešenie

Elegantné riešenie prišlo až časom a to implementáciou funkcionality zvanej script flow, kde je vložená komponenta v sebe schopná niesť, resp. zapuzdrovať súbor udalostí, ktoré sú vykonávané ďalej na základe určitých pravidiel, pričom v pamäti vystupuje ako jediná udalosť a teda zaberie jediný slot.

6.2.3.3 Aplikácia pravidla

Aplikácie navrhnutého postupu výrazne znižuje pamäťové nároky aplikácie, čo má taktiež značný vplyv na výkon a odozvu.

6.2.3.4 Zhrnutie vo všeobecnosti

Spomínaná metóda je aplikovateľná len v prípade, že v danej aplikácií je používaný Coach TK verzie 6.5 a vyššej, pretože od tejto verzie tam existuje príslušná komponenta Script Flow.

6.2.4 Zhrnutie optimalizácií

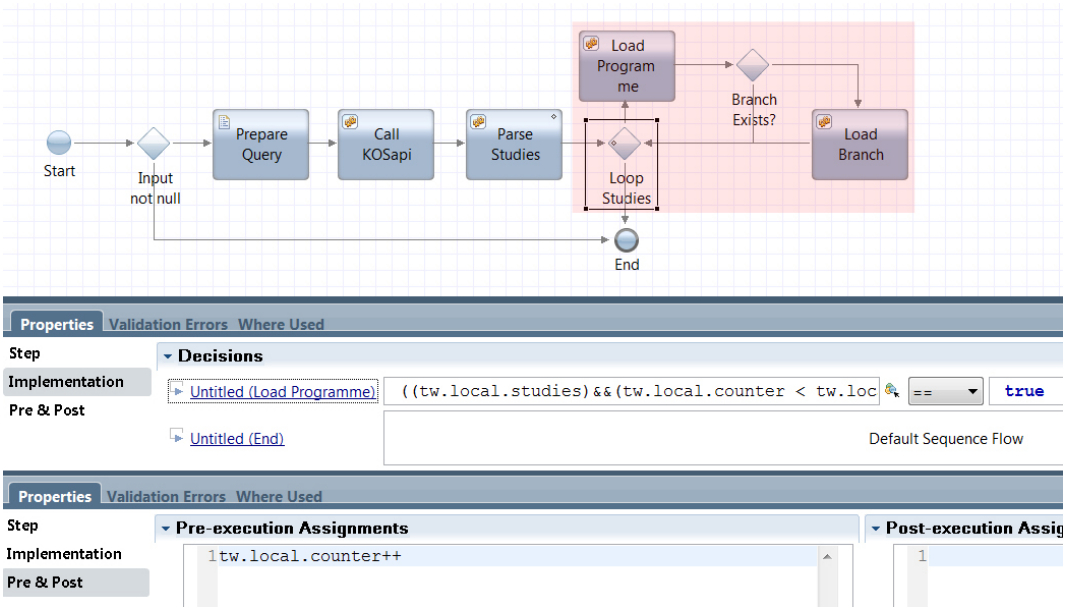
Zavedením pravidiel na frontendovej vrstve znižujeme riziko vzniku chýb po stránkach viditeľnosti, vzhľadu a funkcionality. Tieto kategórie tvoria podľa štatistík 4.3 približne jednu tretinu zaznamenaných problémov. Najväčší prínos majú optimalizácie v rámci zmenových požiadavkov, pretože viac ako dve tretiny 4.2 zaznamenaných tvoria úpravy v častiach, ktoré sú viditeľné užívateľom. Týka sa to kategórií Pridávania, zmazania a úprav atribútov ale aj Vytvorenie, mazanie a úpravy komponent. Po stránke optimalizácií zvyšujú rýchlosť a odozvu aplikácie, jej použiteľnosť a rozšíriteľnosť.

6.3 Servisná vrstva

Optimalizácie tejto časti považujem za najdôležitejšie pretože ide o rýchlosť, odozvu a celkovú flexibilitu pri práci s aplikáciou, ktorú odcenia nie len užívatelia, testeri ale častokrát i samotní vývojári. Ide o backendovú časť, kde bolo cieľom zlepšenie služieb a to najmä odstránením nadbytočných volaní a requestov do databáze, či zlepšiť ich škálovateľnosť z hľadiska znovupoužiteľnosti. Backend môže byť volaný už pri samotnom návrhu procesu 3.2.0.1 alebo ako AJAXový požiadavok v prípade frontendu 3.2.0.1. Postupy a úpravy na tejto vrstve som zdokumentoval, aby boli uchopiteľné a použiteľné pri ktorejkoľvek aplikácii v IBPM.

6.3.1 Viacnásobné volania služby

Väčšina novovytvorených jednoduchých služieb slúži iba k vráteniu záznamu na základe vstupnej hodnoty. V prípade, že je potrebné získať hodnôt viac nastáva problém. Rýchlym riešením, bez potreby upravovať službu je vytvorenie cyklu formou gatewaye, ako môžete vidieť na obrázku 6.19



Obr. 6.19: Zapojenie, v ktorom je v cykle niekoľkokrát volaná rovnaká služba

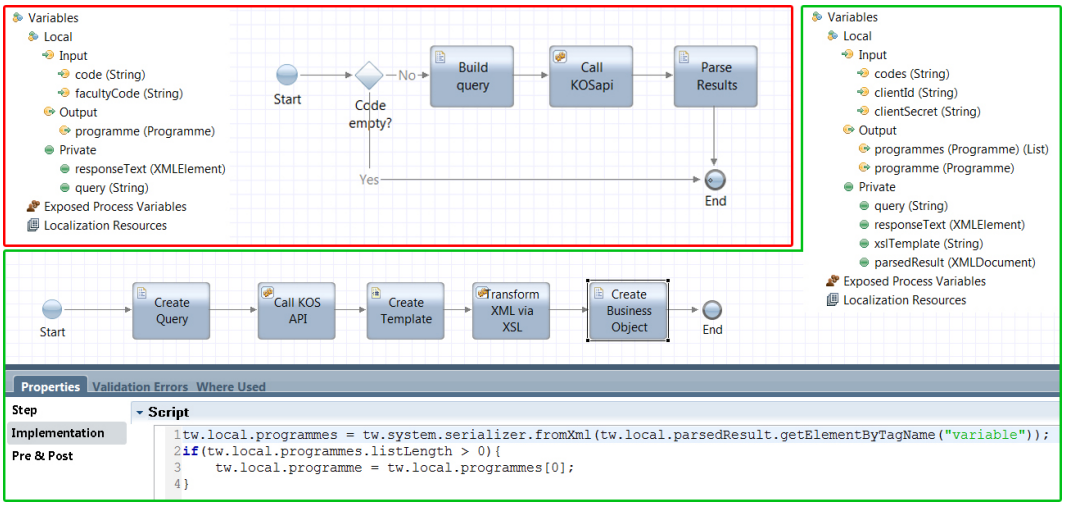
6.3.1.1 Problém

Táto transformácia je v prípade veľkého množstva iterácií mimoriadne neefektívna, pretože každý prechod cyklom znamená, že zlužba je znovu volaná, len s iným parametrom a spravidla je inkrementovaný hneď v "post" záložke komponenty rozcestníka. Hovorovo ide o "for" cyklus implementovaný v BPMN. Súbor takýchto operácií dosť výrazne znižuje odozvu aplikácie a spomaľuje chod systému.

6.3.1.2 Riešenie

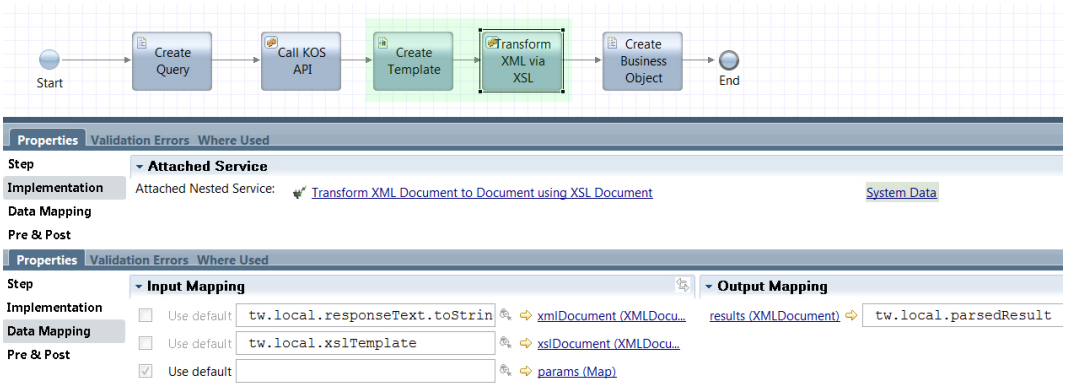
Pri riešení takéhoto problému je nevyhnutné službu poupraviť a optimalizovať tak, že bude schopná na vstupe spracovať i kolekciu, príp. reťazec - string, a teda zoznam uložený v jednom zo vstupných parametrov 6.20. Znamená to, že službu zavolám iba raz, ale dostanem kompletný výsledok ako za bežných okolností pri uplynutí všetkých iterácií cyklu. Vrátenú kolekciu dát spracujem úplne rovnako a používam tak ako doteraz. Pre zaujímavosť pripomeniem, že táto služba je implementovaná podľa návrhového vzoru Adaptér, ktorý som predstavil v kapitole o frameworkoch. Vzor je aplikovaný formou spracovávania vsupu, v prvej skriptovej komponente, volanie externej služby a v ďalších komponentách spracovávanie dát poskytnutých službou. V prípade, že sa zmení externá slúžba na inú, v našom prípade KOS API, servisnú komponentu na-

hradíme a všetky zmeny vykonáme úpravou ostatných komponent, ale vždy len v rámci tejto služby, pretože systém s ňou počíta a je odstienený od jej vnútornej implementácie.

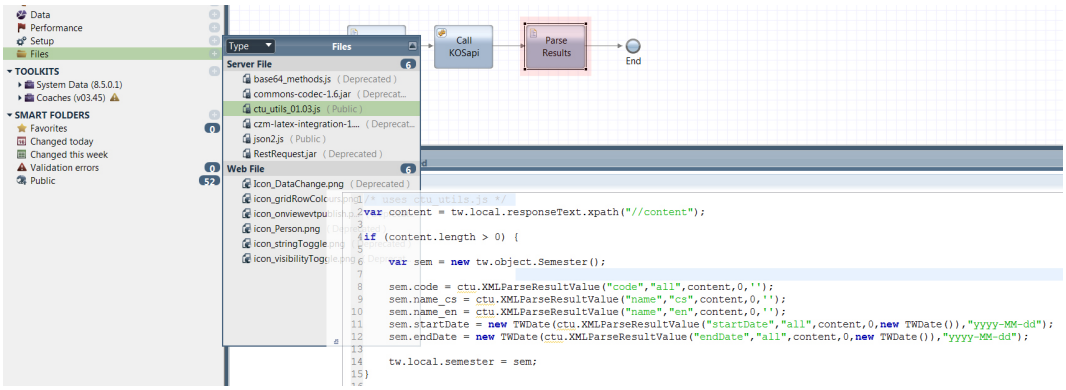


Obr. 6.20: Obsah a premenné pôvodnej a novej služby

Zobrazenie korektného vnútra služby môže pôsobiť mätúco, netreba v tom hľadať zložitosť. Prvá "server side script" komponent z obrázka 6.20 sa rozšírila o spracovávanie rôznych vstupov, kde pred tým spracovávala iba jeden prvok, teraz ich v stringu parsuje a prijíma niekoľko. Na základe nich potom konštruuje query, ktorý volá v druhej, servisnej komponente, ktorá ostala nemenná. Teraz nastáva najväčšia zmena a to v spôsobe získavania dát. Obsah pôvodného parsovania je viditeľný na obrázku 6.22, kde sú používané JS funkcie z iného súboru. Pre neznalých je táto operácia v poriadku, je ale dobre dávať prednosť predpripraveným funkciám, ktoré nám poskytujú Toolkity ako je v našom prípade TK s názvom "System Data". Ten v sebe nesie transformáciu XSL šablóny do business objektu, ktorý je ďalej používaný samotným BPM. Operácie cez externý Javascript majú sčasti vplyv na pamäť, no podstatnejšie je, že skompilovaný kód, ktorý danú transformáciu vykonáva sa môže kedykoľvek za behu z pamäte uvoľniť. Je preto zbytočné siahať pre niečo, čo už máme dávno k dispozícii, a bezdôvodne tak zvyšovať riziko zanesenia chyby. Spomenuté operácie sa týkajú vyznačených komponent, viď obrázok 6.21. Vzor XSL šablóny je zverejnený v prílohe C.



Obr. 6.21: Zapojenie využívajúce funkcionalitu System Data TK



Obr. 6.22: Pôvodné zapojenie využíva funkcie z externého súboru

6.3.1.3 Aplikácia pravidla

Táto zmena priniesla ďalšiu, dobre optimalizovanú službu, ktorá výrazne znížila dobu odozvy a spracovávania informácií. Táto skutočnosť vyplýva z meraní a zvýšenia rýchlostí v rámci všetkých častí, kde je použitá.

6.3.1.4 Zhrnutie vo všeobecnosti

Demonštrované úpravy sa javia ako samozrejmosť, no je potrebné na to myslieť a nenechať sa uniesť jednoduchosťou a pocitom, že niečo konečne funguje. V prípade služieb to platí obzvlášť, keďže ide o základný stavebný pilier kaž-

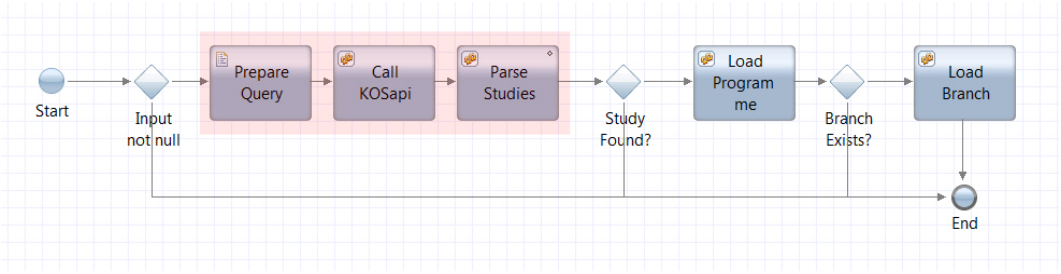
dej aplikácie. Ak sú služby navrhnuté a optimalizované zle, chyba sa dedí a nabaľuje, keďže sú väčšinou používané opakovane.

6.3.2 Znovupoužiteľnosť služieb

Pri vývoji je dôležité implementovať funkcie či služby tak, aby ich bolo možné znovu používať aj v inom kontexte. Zabezpečiť to ale nie je jednoduché a to najmä v prípadoch, kedy nie je vopred zrejmé na čo všetko bude daná služba používaná. Vo všeobecnosti platí pravidlo, že čím je rozsah pôsobenia služby väčší a teda čím viac problémov sa snaží riešiť, tým klesá jej znovupoužiteľnosť. Rozhodnutie ktorým smerom sa má uberať je teda vždy v rukách vývojára.

6.3.2.1 Problém

V aplikácii ZP sa častokrát vyskytovali služby, ktoré riešili množstvo podobných problémov, no práve kvôli tomu bolo vytvorených i množstvo veľmi podobných a len v malej časti líšiacich sa služieb. Názorný príklad je demonštrovaný na obrázku 6.23, kde v jednej službe získavame študijné informácie o študentovi. Služba je zbytočne komplikovaná a vykonáva nadbytočné, časovo náročné operácie.

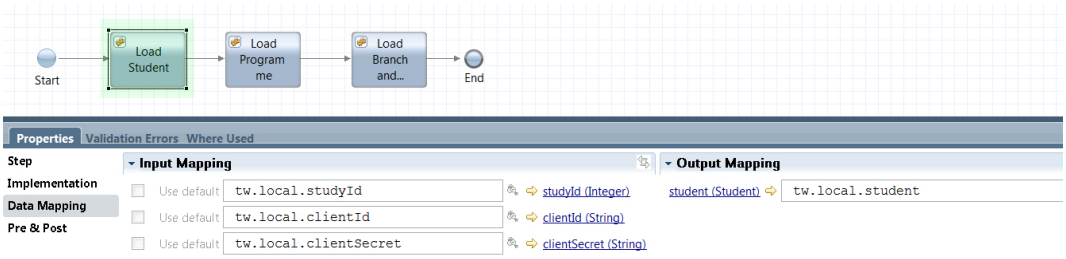


Obr. 6.23: Chybné zapojenie komponent v obsahu služby

6.3.2.2 Riešenie

Pri riešení takéhoto problému je dôležité uvedomiť si, či novovzniknutá služba bude mať zmysel, prípadne čo konkrétne bude riešiť. To nám dá predstavu, ako by mala vyzerať z pohľadu parametrov a obsahu. Z architektonického pohľadu je zrejmé, že práca so študentami a teda získavanie k nemu patriacich informácií je pomerne frekventovaný úkon a teda je to vhodný adept na samostatnú službu, ktorá sa môže prípadne využívať v rámci iných, zložitejších služieb. Znova pripomeniem princíp DI, ktorý uplatníme pri definovaní služby, ktorá plní určitú funkcionalitu. Ak ju potrebujem, službu zavolám a zabezpe-

čím aby pracovala s korektnými vstupnými dátami. Mapovanie vstupných a výstupných parametrov takejto služby je zachytené na obrázku 6.24.



Obr. 6.24: Správne zapojenie komponent v obsahu služby s náhľadom mapovacích parametrov

6.3.2.3 Aplikácia pravidla

Po úpravách získavame prehľadnú službu obsahujúcu ďalšie tri, ktoré už nie sú prepojené cez rozcestníky – vetvenia, ako je viditeľné na obrázku 6.24, a rozličné iné volania. Všetko potrebné je v nich zapúzdrené a teda ten, kto ich používa nemusí riešiť prípady "čo ak". Služba by mala fungovať ako samostaný celok, ktorý na základe vstupu poskytuje adekvátny výstup. V prípade, že sa chyba predsa len vyskytne a služba má na výstupe nesprávne dáta 4.3, vývojár presne vie, kde s hľadaním chyby začať a postupným zmenšovaním rádiusu dojsť až k identifikácii problému.

6.3.2.4 Zhrnutie vo všeobecnosti

Zmenou získame lepší prehľad pri skladaní komplexnejších služieb, kde je hneď jasné a väčšinou z názvu komponenty vyplývajúce čo daná služba poskytuje. Táto problematika je úzko spätá i s rozhodnutím, kedy je lepšie v službe jedným volaním vrátiť rozsiahly objekt, ktorý sa následne parsuje a rozkladá na menšie časti v rámci scriptov, ako pre každú časť volať tú istú službu len s rozdielnymi parametrami.

6.3.3 Uvoľňovanie premenných na konci služby

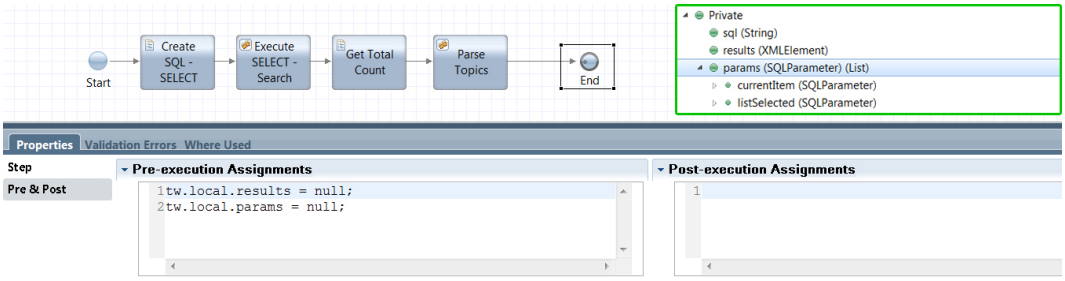
Práca s veľkými objektami je vo väčších aplikáciách štandardom. V prípade IBM BPM ale nastáva pri službách menší problém, ktorý navonok nie je viditeľný.

6.3.3.1 Problém

Ide o uvoľňovanie pamäte, konkrétne premenných, aby sa zbytočne neukladali a nezaťažovali tak BPM databázu. V prípade menších projektov je táto skutočnosť zanedbateľná, ale pri projektoch väčších rozmerov, ako i SZP, sa výskyty takéhoto typu opakuju častejšie a preto čas odozvy zbytočne narastá.

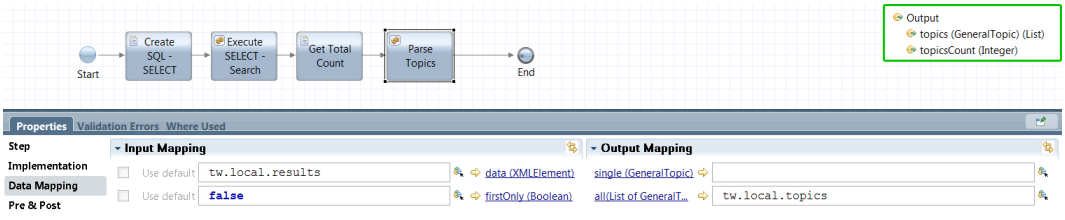
6.3.3.2 Riešenie

Rišenie je v princípe jednoduché. Stačí na konci služby "vynullovať" všetky parametre, prípadne celý objekt postupne, premennú po premennej tak ako to býva u inicializácií. Táto operácia sa odporúča vykonávať pri záložke "Pre-execution" v "End" komponente, viď obrázok 6.25.



Obr. 6.25: Ukážka "nullovania" objektov a ich štruktúra

Dáta, ktoré sa mapovali do služby ako parameter boli spracované, predané do výstupnej premennej 6.26 a preto "uvoľnenie pamäte" nijak neovplyvní návratové hodnoty služby.



Obr. 6.26: Presun dát (mapovanie) na výstupnú premennú

6.3.3.3 Aplikácia pravidiel

Tak ako bolo spomenuté, táto operácia prináša výrazné zlepšenie doby odozvy samotnej aplikácie, čo je pre užívateľov a ich pohodlnosť pri práci so systémom kľúčové.

6.3.3.4 Zhrnutie vo všeobecnosti

Údaje z Human Service nie sú uvoľňované až do okamihu, kedy služba dosiahne konečný bod (endpoint). V článku [25] sa píše, že ak je služba implementovaná, bez predpokladu dosiahnutia takéhoto bodu, ako je napríklad samostatná stránka či presmerovanie, pamäť nebude uvoľnená až po dobu timeoutu, čo je spravidla až 2 hodiny. Preto je potrebné vždy v situáciách spĺňujúcich daný popis tento stav ošetrovať.

6.3.4 Zhrnutie optimalizácií

Po aplikácií pravidiel naprieč servisnou vrstvou je možné dosiahnuť viditeľný posun v rýchlosti prechodov medzi obrazovkami, načítaní dát či všeobecné zlepšenie odozvy pri spracovávaní požiadavkov. Optimalizácia dáva lepšiu možnosť služby testovať postupne, po menších logických celkoch a teda zabrániť problémom s nesprávnymi datami či chýbajúcej funkcionalite. V rámci zmenových požiadavkov dávajú pravidlá, tak ako pri frontendovej vrstve, možnosť lepšie rozširovať a efektívne optimalizovať služby na základe požiadavkov.

6.4 Zhrnutie kapitoly

V tejto kapitole som formuloval pravidlá, ktorých použitie som následne demonštroval na praktickom príklade systému Záverečných prác. Do akej miery sú pravidlá prínosné v rámci systému Záverečných prác, je zhrnuté v následujúcej kapitole. V závere musím konštatovať, že aplikáciou pravidiel na akýkoľvek zvolený systém postavený na technológii IBPM, je veľmi pravdepodobné dosiahnutie zlepšenia niektorých kvalitatívnych parametrov a to z pohľadu vývoja, využiteľnosti a údržby (celá dimenzia procesu životného cyklu), infraštruktúry (dimenzia vlastností) a použiteľnosti (dimenzia kvalitatívnych charakteristík) viď 7.2. Zlepšenie ostatných vlastnosti ako je funckionalita, obsah či spoľahlivosť aplikáciou pravidiel nie je zaručená. Tieto fakty sú v následujúcej kapitole podložené výsledkami testov vykonávaných na aplikácií Záverečné práce.

Kapitola 7

Overenie pravidiel

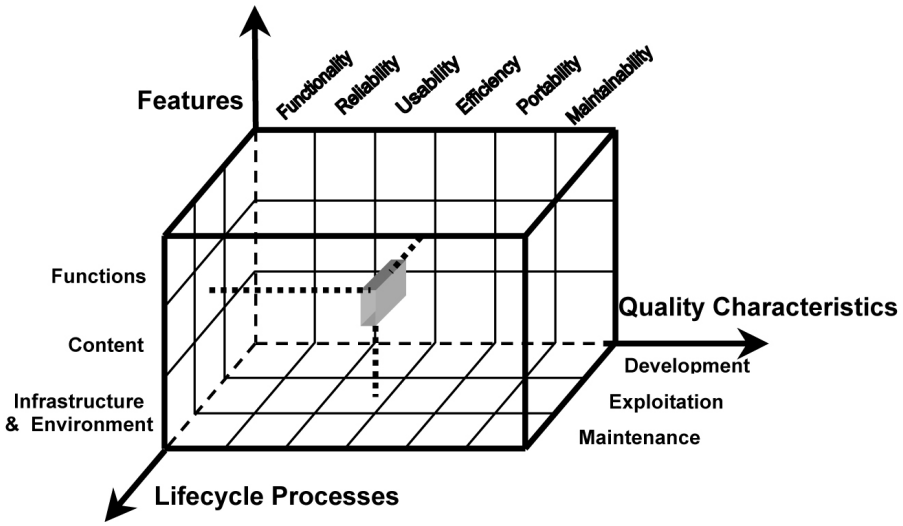
Predchádzajúca kapitola definovala sadu pravidiel, ktoré môžu viesť k zlepšeniu stavu aplikácie postavenej na platforme IBPM ako po stránke odozvy a rýchlosti zo strany užívateľov, tak rozšíriteľnosti a menšej chybovosti z pohľadu vývoja čo pozítívne vplýva na šetrenie finančných zdrojov zákazníka a zníženie doby zaškoľovania nových vývojárov. V tejto kapitole sa budem venovať podrobným testom a meraniu kvalitatívnych parametrov k získaniu lepšieho pohľadu na to, aké výrazné zlepšenie poskytuje použitie definovaných pravidiel.

7.1 Meranie kvalitatívnych parametrov

Ako vizuálnu pomôcku, ktorá dobre znázorňuje úroveň kvalitatívnych parametrov, použim trojdimenzionálny model kvality 7.4. Každná dimenzia znázorňuje určitú skupinu parametrov a to v rámci kvality, vlastností a životného cyklu procesu. Mriežka reprezentuje jednotlivé parametre, ktoré by mala, v ideálnom prípade, aplikácia obsahovať. Objektom v mriežke je samotný produkt - aplikácia, ktorého veľkosť a umiestnenie v priestore znázorňujú pokrytie daných parametrov.

Na následujúcom obrázku je formou modelu vyobrazený stav aplikácie pred použitím pravidiel definovaných v kapitole 4.

Na modeli v rámci dimenzie kvality možno vidieť jedine pokrytie parametra použiteľnosti. To znamená, že i napriek slabej spoľahlivosti či efektivite v rámci súsediacich parametrov mriežky, aplikácia spĺňa účel, pre ktorý bola vytvorená a práca s ním je pre užívateľov prijateľná. Zo strany dimenzie životného cyklu procesu je verzia pred použitím pravidiel plne postačujúca po stránke využitia, no naopak vývoj či údržba sú zastúpené len minimálne, čo znamená, že súčasný stav je v rámci podpory a oprav chýb veľmi komplikovane udržiavaný a akákoľvek implementácia dodatočnej funkcionality by bola veľmi časovo i finančne náročná.



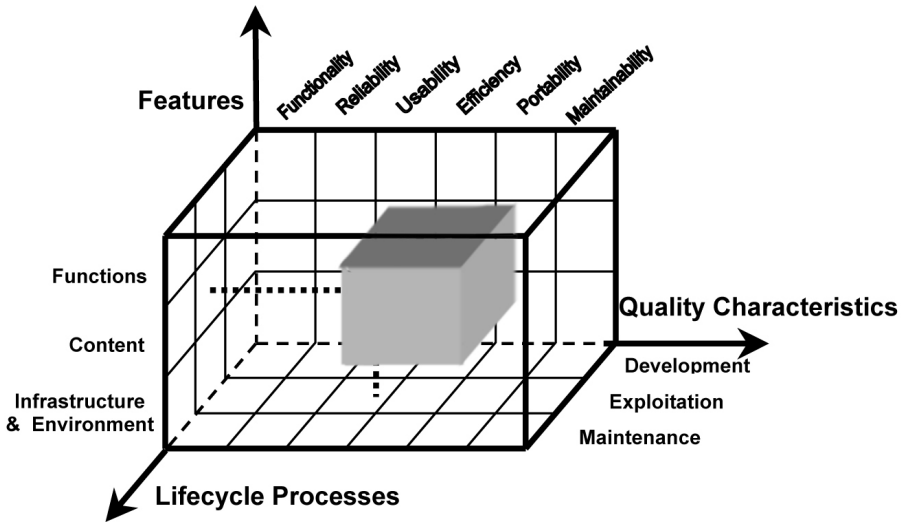
Obr. 7.1: Stav aplikácie vyobrazený na trojdimenzionálnom modele pred použitím pravidiel

Tabuľka 7.1: Známky zlepšení meraných parametrov aplikácie

Životný cyklus procesu		**Kvalita**	
Vývoj	4	Funkcionalita	1
Využiteľnosť	4	Spoľahlivosť	1
Údržba	4	Použiteľnosť	3
Vlastnosti		Efektivita	4
Funkcie	3	Prenosnosť	2
Obsah	2	Udržiavateľnosť	1
Infraštruktúra a prostredie	0		

Ideálnym stavom je, ak aplikácia pokrýva zodpovedne každý parameter a teda vizualizácia objektu kompletne vypĺňa celú mriežku, bez potreby jednotlivé parametre na ose preskupovať. Tento stav je náročné, niekedy až nemožné, dosiahnuť, najmä ak ide o komplexnejšie systémy.

Model kocky je konštruovaný na základe číselných známok zaznamenaných v tabuľke 7.5. Parametre ohodnotené známkou 0 až 1 neboli predmetom zlepšenia alebo nedošlo k žiadnemu zlepšeniu. Známka 2 až 3 znamená mierne zlepšenie a v modely vizuálne zaberajú malú až väčšinovú časť dielika príslušného parametra na ose. Ide napríklad o atribúty použiteľnosti a prenosnosti v rámci dimenzie kvality. V prípade, že by bol zlepšený aj atribut funkcionality, model sa upraví výmenou parametrov na ose, teda v našom prípade spoľahlivosti, aby výsledný objekt vyzeral súvislo, bez väčšej medzery.



Obr. 7.2: Vizualizácia stavu po aplikácii pravidiel

Aplikáciou pravidiel došlo k najväčším zlepšeniam najmä po stránke životného cyklu procesu a teda využiteľnost, údržba a vývoj. Týmto parametrom sú teda priradené známky 4, čo znamená viac ako 100% zlepšenie, ktoré je podložené analytickými testami v následujúcich sekciách. Tieto parametre boli dosiahnuté vďaka zlepšeniu štuktúry komponent a ich znovupoužiteľnosti, kde v súčasnom stave je správa či opravy častí aplikácie rýchlejšia a jednoduchšia. Akúkoľvek funkcionalitu v rámci rozšíriteľnosti je teraz možné jednoducho navrhnúť a efektívne implementovať so zníženou šancou zanesenia chýb. Rovnako vysoké zlepšenie došlo aj v ďalšom atribúte hodnotenom známkou 4. Je to efektivita a výsledky tohoto zlepšenia sú podložené testami rýchlosti v sekcii 7.2.1.

Pri pohľade na model po úpravách 7.2 je na prvý pohľad viditeľné, že je stále čo zlepšovať. Nedostatky zostávajú v rámi dimenzie kvality po stránke spoľahlivosti a dostupnosti, ktoré neboli predmetom optimalizácií a sú spôsobené najmä občasnými výpadkami serverov. Zlepšeníe týchto problémov by spočívalo v zmene infraštruktúry, ktorá je taktiež slabým miestom viditeľným na modele. Tieto úpravy už ale nie sú predmetom doporučneých postupov na platforme BPM.

7.2 Testy

V závere, po aplikácii pravidiel bol systém Záverečných prác podrobený testom, aby sa ukázalo, ktoré časti sa podarilo zlepšiť, a ktoré naopak zhoršiť.

Vykonávané testy boli dvojakého charakteru a to užívateľké a vývojárske. Užívateľský typ testov bol zameraný na rýchlosť a odozvu systému pri každodennej práci.

Druhý, vývojársky, typ testov riešil zlepšenie stavu aplikácie z pohľadu vývoja. Rozšíriteľnosť sa z časových dôvodov testovala analyticky.

7.2.1 Rýchlosť

Rýchlostné testy priniesli, podľa očakávaní, dobré výsledky a teda optimalizáciou bolo dosiahnuté výrazné zlepšenie rýchlosti. Testy boli vykonávané meraním časov vykreslenia jednotlivých obrazoviek, pri nasadení starej a novej verzie systému. Prechod aplikáciou bol náhodný, merania boli vykonávané nezávislo na postupnosti procesu.

Časové úspory sú zaznamenané v následujúcich tabuľkách, hodnoty sú uvedené v sekundách. Z časových dôvodov, keďže aplikácia je pomerne komplexná a obsahuje množstvo prechodov ako je možné vyčítať z tabuliek, som počet meraní stanovil na tri. Výsledná hodnota zobrazená v tabuľke je získaná ako priemer z týchto troch meraní.

Tabuľka 7.2: Vytvorenie rezervácie a schvaľovania Záverečných prác

Špecifiká pohľadu		**Rýchlosť zobrazenia**	
Rola	**Názov pohľadu**	**Pred**	**Po**
Vedúci	Vytvorenie rámcovej TZP	6.257	4.134
	Vytvorenie rezervácie a schvaľovanie ZP	4.311	3.802
	Schválenie rezervácie študenta	2.154	2.158
Študent	Prejavenie záujmu o obor	2.076	1.891
	Záväzná voľba rámcovej témy	2.053	1.785
	Schválenie konkrétneho zadania ZP	2.034	1.788
	Schválenie úprav v zadaní ZP	2.194	2.050
	Finálne prijatie zadania ZP	2.039	1.881
Vedúci ZP	Vytvorenie konkrétneho zadania ZP	3.854	3.547
	Prepracovanie zadania ZP	2.652	2.044
	Finálne prijatie zadania ZP	2.084	1.921
Kat. schval.	Schválenie zadania ZP	1.982	1.995

Tabuľka 7.3: Priradenie oponenta Záverečných prác

Špecifiká pohľadu		**Rýchlosť zobrazenia**	
Rola	**Názov pohľadu**	**Pred**	**Po**
Budúci op. ZP	Prejavenie záujmu o oponentúru ZP v burze voľných oponentúr	5.371	3.225
Kat. schval.	Navrhnutie oponenta	3.974	3.024
	Schválenie oponenta	2.024	1.961
Vedúci ZP	Vytvorenie rámcovej TZP	3.812	2.912
	Navrhnutie oponenta	3.533	3.342
Oponent ZP	Vyjadrenie súhlasu s oponentúrou	2.045	2.037

Tabuľka 7.4: Vloženie a odovzdanie vypracovanej ZP

Špecifiká pohľadu		**Rýchlosť zobrazenia**	
Rola	**Názov pohľadu**	**Pred**	**Po**
Študent	Vloženíe a odovzdanie vypracovanej ZP vo formáte PDF	9.214	7.807
	Rozhodnutie o eskalácii neakceptovanej ZP	2.234	2.241
Vedúci ZP	Akceptácia vypracovanej ZP	1.894	1.731

Tabuľka 7.5: Vytvorenie a odovzdanie posudkov ZP

Špecifiká pohľadu		**Rýchlosť zobrazenia**	
Rola	**Názov pohľadu**	**Pred**	**Po**
Vedúci ZP	Zoznámenie sa s obsahom ZP	3.478	3.379
	Vytvorenie a odoslanie hodnotenia vedúceho ZP	8.254	6.035
Oponent ZP	Zoznámenie sa s obsahom ZP	4.479	4.269
	Vytvorenie a odoslanie posudku oponenta ZP	8.046	6.227

Tabuľka 7.6: Zmenové procesy v ZP

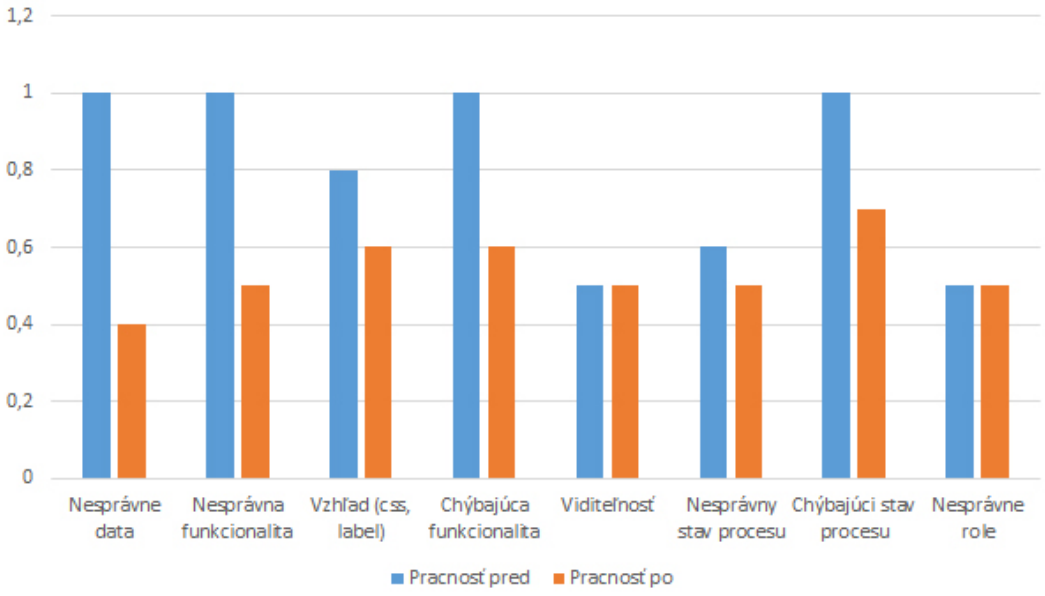
Špecifiká pohľadu		**Rýchlosť zobrazenia**	
Rola	**Názov pohľadu**	**Pred**	**Po**
Vedúci ZP	Návrh zmeny zadania ZP	3.053	3.072
	Špecifikácia zmeny zadania ZP	4.922	4.016
	Navrhnutie nového vedúceho	6.972	4.995
Kat. schvaľovateľ	Schválenie zmeny zadania ZP	2.043	2.055
	Navrhnutie nového vedúceho	5.726	4.039
	Schválenie vedoucího ZP	2.094	1.798
	Navrhnutie nového oponenta	6.102	5.392
	Schválenie oponenta	2.174	2.034
	Schválenie odzadania ZP	2.052	1.915
Študent	Schválenie zmeny zadania ZP	4.632	3.742
	Schválenie úprav v zadaní ZP	4.522	4.030
	Navrhnutie nového vedúceho	6.134	3.992
	Rozhodnutie o eskalácii návrhu na zmenu	2.421	2.326
	Žiadosť o odzadanie ZP	2.263	1.994
Fak. schvaľovateľ	Schválenie zmeny zadania ZP	5.886	5.131
Vedúci ZP	Prepracovanie návrhu zmeny zadania ZP	2.072	2.081
	Schválenie zmeny	3.011	3.053
	Navrhnutie nového oponenta	6.037	5.308
Nový vedúci ZP	Vyjadrenie súhlasu s vedením ZP	2.104	1.817
	Vyjadrenie súhlasu s oponentúrou	2.092	1.915
Oponent ZP	Navrhnutie nového oponenta	6.075	5.367

Výsledky ukazujú výrazné časové zlepšenie v stave po zavedení optimalizácií. Táto skutočnosť nie je ale splnená pri všetkých obrazovkách. Napríklad pri obrazovkách "Schválenie rezervácie študenta", "Schválenie zadania ZP", "Prepracovanie návrhu zmeny zadania ZP"nastal, síce zanedbateľný, no preukázaný nárast doby načítania obrazovky.

Táto anomália mohla byť spôsobená implementáciou, ktorá na úkor menšieho zhoršenia jedného merateľného parametra po stránke rýchlosti a odozvy, naopak zlepšila stav kódu, ktorý nie je na prvý pohľad viditeľný, a teda výrazne zlepšila udržateľnosť. Takéto situácie sú častým predmetom úvah, či výsledná zmena stojí za zhoršenie iného parametra a o koľko.

7.2.2 Porovnanie zlepšenia času opravy chýb

Cieľom týchto testov bolo získať informácie o množstve a dobe riešenia chýb vzniknutých nasadením optimalizovanej verzie systému. Charakter chýb súčasného stavu je rovnaký, no výrazne sa zmenila doba ich opravy čo zobrazuje následujúci graf 7.3.



Obr. 7.3: Graf porovnania doby opravy chýb v stave pred a po aplikácii pravidiel

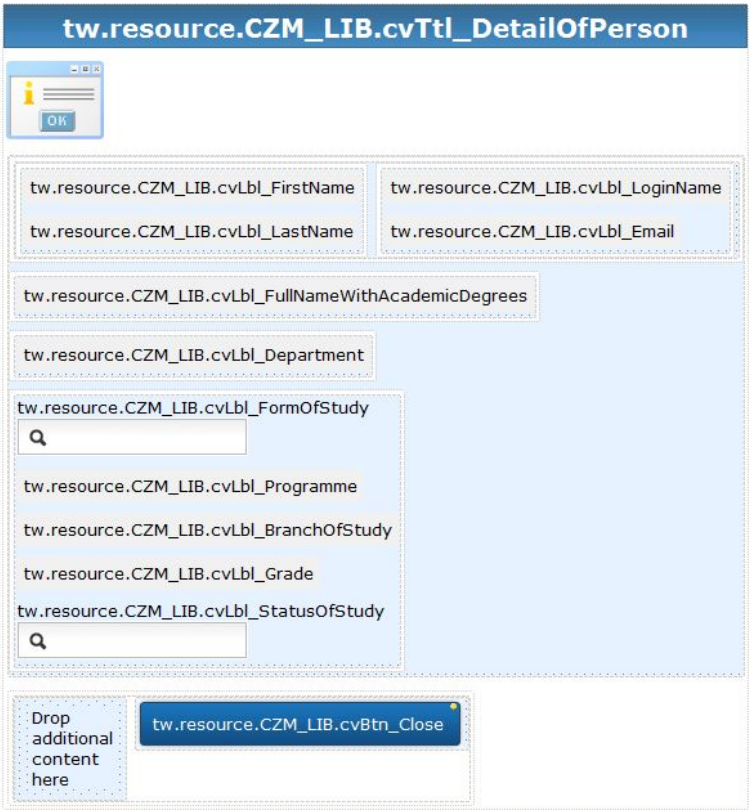
Testy boli vykonávané dvoma testermi, ktorí na základe pripravených scenárov vykonávali operácie naprieč celým procesom od vytvorenia zadania práce, cez schvaľovací proces až po akceptáciu, nahranie zo strany študenta, oponentúru a napokon vyhodnotenie. Počas testov bolo síce zaznamenaných väčšie množstvo bugov, no pri porovnaní doby opravy so stavom pred a po, je vidieť výrazný posun vpred.

Najčastejším typom chýb boli nesprávne hodnoty, kde vidieť najväčší prínos, či chýbajúce atribúty vo formulároch. Vďaka spoločným a znovupoužívaným komponentám zanikli problémy pri nachádzaní všetkých miest výskytu a teda bolo postačujúce problém riešiť centrálne, len v rámci danej komponenty.

7.2.3 Rozšíriteľnosť

Testy v tejto oblasti súviseli s náročnosťou potenciálnych rozšírení pri stave pred a po. V súčasnosti je aplikácia ZP v úplnom stave, bez novo zadaných nových požiadavkov, ktoré je potrebné implementovať, takže testy prebiehali analyticky.

Príklad menej náročného testu rozšíriteľnosti bolo pridávanie formulárového poľa Select, do CV Person. V rámci tejto zmeny je potrebné imeplementovať logiku, ktorá selectbox plní požadovanými hodnotami, pridať filter, ktorý túto množinu upravuje a pridať držiteľa vybranej hodnoty.



Obr. 7.4: Zobrazenie obsahu CV, ktorý bol v aplikácii použitý na 31 miestach

Takáto zmena bola v stave pred problematická z dôvodu nutnosti pridávania atribútu a celej jeho logiky opakovane do niekoľkých rovnakých CV naprieč modulmi. Súčasný stav tento CV používa na viacerých miestach, jeho konfigurácia sa rieši dynamicky a teda zavádzanie nového atribútu je riešené len v rámci toho daného CV. Vo výsledku je testovanie jednoduché, pretože buď atribút funguje všade alebo nikde a nie je potrebné vyhľadávať všetky výskyty. Týmto bola výrazné znížená chybovosť zavádzania novej funkciona-

lity a taktiež doba vývoja, ktorá v závislosti na počte miest výskytov lineárne narastala.

7.3 Zhrnutie kapitoly

V závere možno konštatovať, že testy priniesli pozítívne výsledky a tie potvrdili prínos definovaných pravidiel a postupov.

Pri zmenách v rámci implementácie som sa zameral na optimalizáciu procesného flow aplikácie Záverečné práce a teda sprehľadnenia a zjednodušenia možnosti dodatočného rozvoja aplikácie. Upravoval som procesy na úroveň jednoduchého pochopenia a zjednodušoval zbytočne zložité časti, čím sa znížila doba vývoja. V prípade optimalizácií na frontendovej časti, jednak z hľadiska rozloženia prvkov (komponent) tak i nadbytočného volania AJAXových funkcií pri vykresľovaní jednotlivých častí, bola zlepšená odozva aplikácie a rýchlosť načítania či spracovávania požiadavkov servera. Pri zmenách štruktúry som na základe starých komponent vytváral nové, lepšie štruktúrované, a tie následne členil do Toolkitov. Vo výsledku som teda dosiahol stav, kde je aplikácia dobre škálovateľná a jednoducho rozšíriteľná.

Napokon je tu stále skutočnosť, že zavedenie pravidiel nemusí mať jednoznačný vplyv na rýchlosť systému, najmä ak ide o menej komplexné aplikácie, no ich dodržiavanie zlepšuje udržateľnosť a orientáciu v kóde, čo je obrovským prínosom pre všetkých vývojárov a v konečnom dôsledku i klienta, kde sa mu týmto znižujú celkové finančné náklady na dodatočný vývoj či podporu systému.

Záver

Skutočnosť či vytvorené vzory a rady sú prínosom i vo všeobecnosti, je závislá najmä na schopnosti a motivácii vývojárov poskytnuté prostriedky použiť. Tak ako každá rada, môže byť akceptovaná, dobre aplikovaná a časom môže priniesť značný úžitok.

Musím ale konštatovať, že v prípade systému Záverečných prác, bola aplikácia pravidiel veľmi prínosná a pomocou nej sa stanovené ciele podarilo naplniť. Jej aplikáciou sa podarilo dosiahnuť, že súčasný stav je dobre škálovateľný a rozložený na nezávislé časti, ktoré sú jednoducho rozšíriteľné. Optimalizácia mala za následok aj zlepšený chod a odozvu systému z pohľadu užívateľov o čom svedčia i výsledky testov.

Osobný prínos

Praktické skúsenosti v akejkoľvek sfére v IT sú neodmysliteľnou súčasťou procesu vývoja každého vývojára. Teória je podstatná, no bez praxe je to vždy len stav hypotézy a subjektívneho názoru. V rámci tejto práce som si vyskúšal optimalizáciu už vydaného releasu aplikácie, ktorá sa vyvíjala za úplne iných podmienok a nástrojov aké sú dostupné dnes. Získal som skúsenosti ako správne manažovať svoj čas, keďže súčasne som pôsobil na ďalšom projekte postavenom na technológii IBM BPM. Pri práci v komunite vývojárov som mal možnosť nazrieť do častí, ktoré som doposiaľ neobjavil a teda moje skúsenosti v tejto oblasti rástli omnoho rýchlejšie.

Pohľad ďalej

V rámci ďalšieho rozvoja je cieľom vytvoriť vzorovú aplikáciu, ktorá začínajúcim vývojárom v IBM BPM poskytne komplexný prehľad možností tejto technológie a to formou sady tutoriálov a vzorových implementácií. Cieľom je centralizácia materiálov na výučbu či zdokonaľovanie sa v oblasti vývoja procesných aplikácií v IBM BPM a poskytnúť tak miesto odkiaľ je možné čerpať všetky potrebné informácie jednoducho a rýchlo.

Literatúra

[1] Nepal, M.: Business Process Management – Overview. [online], 2018, [cit. 13. 4. 2018]. Dostupné z: https://kissflow.com/bpm/business-process-management-overview

[2] Palmer, N.: What is BPM. [online], 2014, [cit. 15. 4. 2018]. Dostupné z: https://bpm.com/what-is-bpm

[3] Messler, Z.: La différence entre Worflow management et BPM. [online], 2015, [cit. 15. 4. 2018]. Dostupné z: https://www.appian.com/blog-fr/workflow-vs-bpm-roadtrip

[4] Reichert, M.; Schiebel, E.: La différence entre Worflow management et BPM. [online], 2012, [cit. 15. 4. 2018]. Dostupné z: http://dbis.eprints.uni-ulm.de/803/1/DA_Wohlhaupter_12.pdf

[5] Reichert, P. D. M.: Research in Business Process Management: A bibliometric analysis. [online], 2012, [cit. 7. 5. 2018]. Dostupné z: http://dbis.eprints.uni-ulm.de/803/1/DA_Wohlhaupter_12.pdf

[6] Taylor, J.: The History of Project Management: Gantt and the Early 20th Century. [online], 2011, [cit. 13. 4. 2018]. Dostupné z: https://www.brighthubpm.com/methods-strategies/11654-gantt-and-the-early-20th-century-in-pm-history

[7] Rodriguez, M.: BPM: The Intelligent Assembly Line. [online], 2009, [cit. 13. 4. 2018]. Dostupné z: http://www.kmworld.com/Articles/White-Paper/Article/BPM-The-Intelligent-Assembly-Line-60097.aspx

[8] Rouse, M.: Business process management. [online], 2011, [cit. 13. 4. 2018]. Dostupné z: https://searchcio.techtarget.com/definition/business-process-management

[9] Vercruysse, J.: The 5 basic elements of the BPM Life cycle. [online], 2017, [cit. 13. 4. 2018]. Dostupné z: https://www.effic.be/en/5-basic-elements-of-the-bpm-life-cycle

[10] Arsanjani, A.; Bharade, N.; Borgenstrand, M.: Business Process Management Design Guide Using IBM Business Process Manager. [online], 2015, [cit. 15. 4. 2018]. Dostupné z: http://www.redbooks.ibm.com/redbooks/pdfs/sg248282.pdf

[11] Rummler, G.: Contributions and Conflicting Ideologies in the Field of Performance. [online], 2010, [cit. 13. 4. 2018]. Dostupné z: https://pdfs.semanticscholar.org/e14b/285dc85a64a5764eeb05031d34c5e3c0cbfd.pdf

[12] Martin: Business Process Management Life Cycle. [online], 2017, [cit. 13. 4. 2018]. Dostupné z: https://www.cleverism.com/business-process-management-life-cycle

[13] Fibichr, J.: Leveraging Toolkits in BPM Application Development. [online], 2013, [cit. 13. 4. 2018]. Dostupné z: https://dip.felk.cvut.cz/browse/details.php?f=F8&d=K102&y=2013&a=fibicjar&t=dipl

[14] Grasseová, M.; Dubec, R.; Horák, R.: Procesní řízení. In *Procesní řízení ve veřejném sektoru: teoretická východiska a praktické příklady*, 2008, s. 177–185.

[15] Košturiak, J.: Benchmarking. [online], 2017, [cit. 13. 4. 2018]. Dostupné z: https://www.ipaslovakia.sk/sk/ipa-slovnik/benchmarking

[16] Shyshkina, H.: Nástroje pro modelování a optimalizaci podnikových procesů. [online], 2016, [cit. 15. 4. 2018]. Dostupné z: https://is.muni.cz/th/374639/fi_m/Diplomova_prace.pdf

[17] Šmída, F.: Zavádění a rozvoj procesního řízení ve firmě. In *Management v informační společnosti*, 2007, s. 152–158.

[18] Gartner: Business Process Management Suites (BPMSs). [online], 2016, [cit. 24. 4. 2018]. Dostupné z: https://is.muni.cz/th/374639/fi_m/Diplomova_prace.pdf

[19] Company, F. R.: Top 10 best BPM software products. [online], 2017, [cit. 13. 4. 2018]. Dostupné z: http://www.enterpriseappstoday.com/management-software/top-10-best-bpm-software-products.html

[20] Fleming, M.: Guest Blog: Industry Analyst Maureen Fleming Shares Her Thoughts on BPM Trends. [online], 2014, [cit. 13. 4. 2018]. Dostupné z: https://blogs.opentext.com/guest-blog-industry-analyst-maureen-fleming-shares-her-thoughts-on-bpm-trends

[21] Bursík, F.: Nástroje BPM první a druhé generace. [online], 2007, [cit. 13. 4. 2018]. Dostupné z: http://bpm-tema.blogspot.cz/2007/05/nstroje-bpm-prvn-druh-generace.html

[22] Kolban, N.: Kolban's book on IBM Business Process Management. [online], 2014, [cit. 8. 5. 2017]. Dostupné z: http://neilkolban.com/ibm/wp-content/uploads/2014/12/Kolbans-IBPM-Book-2014-12.pdf

[23] Rouse, M.: Bug. [online], 2007, [cit. 13. 4. 2018]. Dostupné z: https://searchsoftwarequality.techtarget.com/definition/bug

[24] Spacey, J.: What is a Change Request? [online], 2017, [cit. 13. 4. 2018]. Dostupné z: https://simplicable.com/new/change-request-definition

[25] Collins, M.; Duan, Z. H.; Fried, A.; aj.: IBM Business Process Manager V8.5 Performance Tuning and Best Practices. [online], 2015, [cit. 8. 5. 2017]. Dostupné z: http://www.redbooks.ibm.com/redbooks/pdfs/sg248216.pdf

[26] Martin, R. C.: Function Names Should Say What They Do. In *Clean Code*, 2009, s. 297–298.

Dodatok **A**

Zoznam použitých skratiek

AJAX Asynchronous Javascript and XML

API Application Programming Interface

BI Business Inteligence

BPM Business Process Manager

BPMN Business Process Manager Notation

BPEL2 Business Process Execution Language Version 2.0

CV Coach View

CZM Centrum znalostního managementu

Debugovať testovanie, ladenie počítačového programu

EAI Enterprise Application Integration

HS Human Service

IBM International Business Machine

IBPM IBM BPM

IDE Integrated Development Environment

JSF JavaServer Faces

KPI Key Performance Indicators

LDAP Lightweight Directory Access Protocol

MVC Model View Controller

MVCS Model View Controller Service

OS Operačný systém

OOP Obiektovo-orientované programovanie

TK Toolkit

UCA Undercover Agent

Dodatok **B**

Slovník pojmov

Backend časť webovej aplikácie, ktorá obplyvňuje funkcionalitu a nie je viditeľná bežným užívateľom

Bug chyba

Business Intelgence označenie pre informačné technológie, aplikácie a metódy na zber, normalizáciu, analýzu, prezentáciu a interpretáciu obchodných dát o vývoji v organizácii

Cache pomocná pamäť, ktorá si ukladá dočasne dáta, aby skrátilo čakaciu dobu, často oproti serveru

Flow Tok, Napr. procesný flow, procesný tok, flow dát

Frontend časť webovej aplikácie vidieľná bežným návštevníkom

LDAP protokol pre ukladanie a prístup k datam na adresárovom serveri

Learning curve grafické vyobrazenie plynulosti postupu pri učení sa

Logy zátnamy zo servera o činnosti aplikácie

Middleware je počítačový softvér, ktorý prepája softvérové komponenty alebo osoby a ich aplikácie

On Premise riešenie v podobe aplikácií inštalovaných na firemných serveroch

Plugin doplnkový modul inej aplikácie a rozširuje tak jej funkčnosť

Query otázka, spravidla príkaz oproti databáze

Refaktoring proces vykonávania zmien v softvérovom systéme za účelom vylepšovania jeho vnútornej štruktúry

Release nasadenie aplikácie do produkčného prostredia, napr. 1. produkčný release

Select formulárový prvok

Selectbox miesto výberu prvku vo formulárovom poli Select

String datový typ, textový reťazec

Timeout uplynutie, vypršanie určitého časového limitu

Widget ovládací prvok, zariadenie prispôsobené k čo najjednoduchšiemu ovládaniu iného zariadania

Dodatok C

XLS šablóna vstupu transformačnej služby

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="2.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template name="format-date-time">
            <xsl:param name="date" />
            <xsl:value-of select="substring($date, 1, 4)" />
            <xsl:text>/</xsl:text>
            <xsl:value-of select="substring($date, 6, 2)" />
            <xsl:text>/</xsl:text>
            <xsl:value-of select="substring($date, 9, 2)" />
            <xsl:text> </xsl:text>
            <xsl:text>00</xsl:text>
            <xsl:text>:</xsl:text>
            <xsl:text>00</xsl:text>
            <xsl:text>:</xsl:text>
            <xsl:text>00</xsl:text>
            <xsl:text>.0 UTC</xsl:text>
        </xsl:template>
      <xsl:template match="entry">
        <variable type="Semester">
            <code type="String"><xsl:value-of select="content/code" /></code>
            <name_cs type="String"><xsl:value-of select="content/name[@lang='cs']"
                 /></name_cs>
            <name_en type="String"><xsl:value-of select="content/name[@lang='en']"
                 /></name_en>
            <xsl:choose>
                <xsl:when test="content/startDate!=''">
                    <startDate type="Date">
                        <xsl:call-template name="format-date-time">
                            <xsl:with-param name="date" select="content/startDate"
                                 />
                        </xsl:call-template>
                    </startDate>
                </xsl:when>
                <xsl:otherwise></xsl:otherwise>
            </xsl:choose>
            <xsl:choose>
                <xsl:when test="content/endDate!=''">
                    <endDate type="Date">
                        <xsl:call-template name="format-date-time">
                            <xsl:with-param name="date" select="content/endDate" /
                                 >
                        </xsl:call-template>
                    </endDate>
                </xsl:when>
                <xsl:otherwise></xsl:otherwise>
            </xsl:choose>
            </variable>
    </xsl:template>
</xsl:stylesheet>
```

Dodatok D

Diagram CVs troch optimalizovaných modulov

Obr. D.1: Náhľad obsahujúci všetky komponenty príslušných obrazoviek v rámci všetkých optimalizovaných aplikácií

Dodatok **E**

Obsah priloženého CD

- `attachments` ..... zdrojové súbory z príloh
  - `cv_diagram.jpg` ..... Diagram CV aplikácií
  - `transform_template.xsl` ..... Transformačná XSL šablona
- `images` ..... adresár s obrázkami
- `mybibliographyfile.bib` ..... súbor citovaných zdrojov
- `prokiiva_DP_2018.pdf` ..... text práce ve formáte PDF
- `prokiiva_DP_2018.tex` ..... zdrojová forma práce vo formáte LaTeX
- `readme.txt` ..... stručný popis obsahu CD