Jihočeská univerzita v Českých Budějovicích

Přírodovědecká fakulta

Bakalářská práce

2017 **David Mrázek**

Jihočeská univerzita v Českých Budějovicích

Přírodovědecká fakulta

Uplatnění workflow engine pro implementaci organizačních procesů

Bakalářská práce

Autor: David Mrázek

Vedoucí práce: Ing. Martin Čížek, MBA.

České Budějovice 2017

Bibliografické údaje

Mrázek D., 2017: Uplatnění workflow engine pro implementaci organizačních procesů [Applying a workflow engine to implement organizational processes. Bc. Thesis, in Czech] - 66p., Faculty of Science, University of South Bohemia, České Budějovice, Czech Republic.

Anotace

Bakalářská práce se zabývá softwarem typu workflow engine. V teoretické části je vymezena role tohoto typu softwaru a popsány související pojmy. V rešeršní části jsou srovnány Java open-source workflow engines. Stěžejní částí práce je srovnání přístupů k řízení procesů pomocí workflow engine a bez workflow engine.

Klíčová slova

Workflow, Workflow engine, Activiti, Open-source

Annotation

Bachelor thesis deals with specific type of software called workflow engine. In the theoretical part is defined role of this type of software and related concepts are described. The search section compares Java open-source workflow engines. The main part of the thesis is comparation of approaches to process control using a workflow engine and without a workflow engine.

Key words

Workflow, Workflow engine, Activiti, Open-source

Prohlášení

Prohlašuji, že svoji bakalářskou práci jsem vypracoval samostatně pouze s použitím pramenů a literatury uvedených v seznamu citované literatury.

Prohlašuji, že v souladu s § 47b zákona č. 111/1998 Sb. v platném znění souhlasím se zveřejněním své bakalářské práce, a to v nezkrácené podobě elektronickou cestou ve veřejně přístupné části databáze STAG provozované Jihočeskou univerzitou v Českých Budějovicích na jejích internetových stránkách, a to se zachováním mého autorského práva k odevzdanému textu této kvalifikační práce. Souhlasím dále s tím, aby toutéž elektronickou cestou byly souladu s uvedeným ustanovením zákona č. 111/1998 Sb. zveřejněny posudky školitele a oponentů práce i záznam o průběhu a výsledku obhajoby kvalifikační práce. Rovněž souhlasím s porovnáním textu mé kvalifikační práce s databází kvalifikačních prací Theses.cz provozovanou Národním registrem vysokoškolských kvalifikačních prací a systémem na odhalování plagiátů.

V Českých Budějovicích dne ..... podpis autora .....

Poděkování

Rád bych poděkoval vedoucímu práce Ing. Martinu Čížkovi, MBA za cenné rady a čas, který věnoval mé práci. Také děkuji mé rodině za podporu při studiu.

Obsah

1 Úvod

Téměř každá činnost v organizaci probíhá na základě definované množiny kroků s cílem dosáhnout nějakého výsledku. Tyto kroky a jejich cíl představují organizační proces.

Většina organizací se neobejde bez spolupráce s informačními technologiemi. Informační technologie jsou součástí a často nepostradatelným prvkem v rámci organizační struktury a jsou tedy zapojeny i do procesů. Mohou sloužit jako podpůrný mechanismus k provedení daného kroku procesu, nebo i celé procesy řídit. Zapojení informačních technologií do řízení procesů umocňuje současný tlak na efektivitu a automatizaci, která má za cíl snížit náklady a zlepšit chod organizace.

Většina organizací, ve kterých je požadováno automatizovat řízení procesů, má svůj informační systém. Je vhodné, aby řešení pro řízení procesů bylo zakomponováno do tohoto informačního systému. Vytvoření kompletního řešení pro řízení procesů od začátku může být nákladné. Při tlaku na nízké náklady je vhodnou alternativou open-source řešení.

Je tedy žádoucí zabývat se způsoby, kterými je možné automatizovat řízení procesů a vyhodnocovat vhodnost jejich použití, náročnost vytvoření a celkový přínos. Zvolený přístup by měl podporovat všechny zainteresované strany. Tedy business analytiky, kteří proces navrhují i vývojáře, kteří proces implementují. Je důležité zvolit vhodná kritéria pro výběr takového řešení a také zhodnotit vhodnost jeho nasazení i s ohledem na rychle se měnící prostředí organizační struktury, kdy je potřeba rychle reagovat na změny, vytvářet nová řešení a tyto řešení nasazovat.

1.1 Cíle

Cílem práce je seznámit čtenáře s problematikou workflow a workflow engine a demonstrovat použití takového nástroje v praxi z pohledu business analytika a vývojáře.

Dílčí cíle práce jsou:

- V teoretické části seznámit čtenáře s BPM, BPMN a souvisejícími pojmy.
- Vymezit roli software typu workflow engine a srovnat nejčastěji používané open-source alternativy.
- Navrhnout aplikaci využívající workflow engine Activiti, která implementuje žádosti o přidělení zdroje. Definice workflow bude konzultována se školitelem. Aplikace musí poskytovat možnost vytvořit žádost, schvalovat ji, zobrazit průběh jejího řešení a zobrazit seznamy žádostí a úkolů celkově.
- Výše uvedenou aplikaci naprogramovat a vytvořit uživatelské rozhraní s moderními webovými technologiemi.
- Zhodnotit výhody a nevýhody používání workflow engine oproti naprogramování podobné aplikace „na zelené louce“

V teoretické části práce je seznámeno s pojmy týkající se problematiky, při popisu těchto pojmů je kladen důraz na zasazení do kontextu s informačními technologiemi.

Na teoretickou část, navazuje část rešeršní, kde jsou srovnány nejčastěji používané open-source workflow engine alternativy. Součástí zadání praktické části je využít workflow engine Activiti, který je založený na Javě. Srovnání se tedy vymezuje na workflow engine z prostředí Java.

V praktické části je vytvořena referenční aplikace využívající workflow engine Activiti implementující žádost o přidělení role, ze všemi zadanými specifikacemi. Na praktickou část navazuje zhodnocení využití workflow engine, kde je řešení s použití workflow engine, srovnáno s přístupem kdy použit není.

2 Teorie problematiky

V teoretické části jsou popsány základní pojmy týkající se business procesů a business process managamentu. Prostředí business process managementu obsahuje širokou škálu pojmů. Tyto pojmy mezi sebou mají různé návaznosti, které je potřeba vysvětlit. Samotní odborníci mají na tuto problematiku odlišné pohledy a do tohoto prostředí to vnáší ještě více nejasností. V této části je tedy především vycházeno z definic autority Workflow Management Coalition, dále WfMC. WfMC je mezinárodní instituce, které se snaží sjednotit pojmy související s touto problematikou. Pojmy jsou popisovány tak, aby bylo možné ukázat asociace mezi technickým a business prostředím a poskytnout pohled vhodný pro business analytika a technicky zainteresované osoby, respektive softwarového vývojáře.

2.1 Business process

Klíčovým pojmem je Business proces, který má nesčetný počet definic, ale všechny vyjadřují podobný význam. Jedna z definic zní následovně:

„Souhrn činností, transformujících souhrn vstupů do souhrnu výstupů, přičemž tyto výstupy jsou určeny pro jiné lidi nebo procesy“ [1]. Tím je myšleno, že vstupy jsou transformovány pomocí různých prostředků na výstupy, které reprezentují výsledek, či cíl procesu.

Souhrn činností, které jsou prováděny během procesu mají mezi sebou určité návaznosti a měly by být vykonávány v určitém pořadí. Z tohoto pohledu se jako vhodná definice jeví: „*Série logicky souvisejících činností nebo úkolů, jejichž prostřednictvím, jsou-li postupně vykonány – má být předem definovaný soubor výsledků.*“ [2]

Proces bývá nasazen v rámci nějakého podniku, společnosti, či organizační struktury. Proces ve spojení s organizační strukturou je možné definovat následovně: Byznys proces je po částech uspořádaná množina procedur a aktivit, které společně realizují podnikatelský nebo strategický cíl, obvykle v kontextu organizační struktury definující funkce rolí a jejich vztahy [3]. Procedurou je zde myšlen podproces, který je obsažen v daném procesu. Aktivita je pojem vyjadřující atomickou činnost, které představuje jeden logický a dále nedělitelný krok při vykonávání

procesu [3]. Role zde vyjadřuje dovednost či soubor dovedností, které jsou přiřazovány k jednotlivým aktivitám.

Pro zpracování procesu je důležité nejen organizační prostředí, ale také technické prostředky daného subjektu, tedy technologie, které lze využít k provedení procesu. Ve spojení s touto myšlenkou je vhodné uvézt následující definici: Business process se skládá z aktivit, které jsou prováděny v koordinaci s organizačním a technickým prostředím. Tyto aktivity společně realizují cíl procesu [4].

2.1.1 Model procesu

Model byznys procesu je reprezentace byznys procesu v určité formě [5]. Forma, ve které je proces zapsán může být různého druhu, musí však vhodně reflektovat zamýšlený průběh procesu.

Pro modelování byznys procesu se používají různé normy a standarty, které mají určitou syntaxi, sémantiku či vizuálně definovanou podobu. Model procesu vytvořený standardizovanou metodou poslouží k pochopení procesu všem zainteresovaným stranám, které jsou s danou metodou modelování obeznámeny. Pokud je zamýšleno automatizované zpracování procesu, je vhodné vybrat metodu, která takové zpracování podporuje. Díky tomu se snadněji vytvoří můstek mezi byznys analytikem a vývojářem. Takovou metodo je například BPMN, bude vysvětleno dále.

Pojem business process model je často zaměňován s pojmem definice procesu. Pojem definice procesu je však více spojován s automatizovaným zpracováním procesu. WFMC definuje pojem následovně: Definice procesu je reprezentace procesu ve formě, která podporuje automatické zpracování. Obsahuje aktivity, jejich vztahy a kritéria indikující začátek a konec procesu, dále informace o jednotlivých aktivitách jako jsou účastníci, asociované aplikace a data [3].

2.1.2 Instance procesu

Business process instance je konkrétní případ procesu [4]. Z pohledu objektového programátora je vztah mezi modelem procesu a instancí procesu stejný jako mezi třídou a instancí dané třídy.

2.2 Workflow

Workflow je definováno jako automatizace business procesu jako celku či jen jeho části, během které jsou dokumenty, informace, úkoly, či další zdroje předávány od jednoho účastníka k jinému na základě definované množiny pravidel [3].

Právě díky automatizaci by mělo workflow vést k efektivnějšímu zpracování procesu, ze stejného důvodu však klade vysoké nároky na přesnou a jednoznačnou definici procesu [4].

Samozřejmě proces nemusí mít pouze části, které se nechají plně automatizovat, ale také části, které je nutné provést manuálně, mluvíme pak o *human interaction workflows,* tedy workflow, do kterého jsou aktivně zapojeni osoby [3] [4].

Aby mohla být vytvořena instance procesu, musí být byznys proces zapsán ve formě, která podporuje jeho automatické zpracování. Tato forma je obecně nazývaná, jak již bylo zmiňováno, definice procesu.

Automatizace workflow zajišťuje Workflow management system, česky systémy řízení workflow. Tento systém definuje, vytváří a řídí průběh workflow pomocí softwaru běžícího na jednom nebo více workflow engines, které jsou schopny interpretovat definici procesu, komunikovat s účastníky workflow a pokud je potřeba, využít další nástroje a aplikace [3].

Workflow Management system se skládá z více softwarových komponent, které udržují a interpretují definice procesu, a vytváří a spravují instance procesu a kontrolují jejich interakci s ostatními účastníky a aplikacemi. Pojem workflow management system naznačuje, že se jedná o samostatnou aplikaci, která má na starosti pouze zpracování procesů. V rámci organizační struktury je potřeba mít komplexní systém, který bude poskytovat více funkcionality než pouhou správu a zpracování procesů, proto se místo termínu workflow management system používá termín workflow component [4]. Workflow component není samostatná aplikace, ale spíše je vystavěná do aplikace a je tady součástí většího systému. Podstatnou částí pro workflow management system, či workflow component je workflow engine, který tvoří jeho jádro.

2.2.1 Workflow engine

Workflow engine poskytuje běhové prostředí pro instance procesů založené na procesní definici [3]. Poskytnutí běhového prostředí zahrnuje tyto funkce:

- Interpretaci procesní definice
- Vytvoření instance procesu
- Řízení průběhu procesu
- Navigaci mezi aktivitami
- Kontrolu a správu procesu

Interpretací procesní definice je myšlena schopnost zpracovávat proces, který je definován v určité formě. Dokáže tedy tuto definici zpracovat, vytvářet instanci takového procesu a automatizovaně zpracovávat. Dále řídit průběh tohoto proces. Workflow engine si lze představit jako stavový automat, kde jednotlivé stavy odpovídají dílčím úkolům a přechody mezi nimi jsou realizovány na základě definovaných pravidel. Právě přechody mezi jednotlivými aktivitami, je zásadní funkcí pro řízení průběhu procesu. Workflow engine zajišťuje, aby aktivity byly prováděny v požadovaném pořadí.

Nezbytnou funkcionalitou také je kontrola a správa procesu, tím je myšleno získávání informací o procesu, které se dají použít pro jeho kontrolu a správu. Z pohledu životního cyklu procesu je tento krok vhodný pro vyhodnocení výkonosti procesu a jeho případnou optimalizaci.

Součástí workflow engine je také komunikace s dalšími částmi aplikace při integraci do informačního systému jako workflow komponenty.

2.3 Business Process Management

Pro Business Process Management (dále jen BPM) existuje mnoho rozličných definic. Co všechno BPM zahrnuje a co vlastně je, řeší spousty článků a autorit v tomto oboru, ve snaze najít jednu společnou definici.

Workflow Management Coalition (WfMC) je mezinárodní instituce, která se snaží sjednotit pojmy týkající se BPM. WfMC ve spojení s dalšími experty a autoritami vytvořili následující

definici. Business Process Management je disciplína, zahrnující modelování, řízení, kontrolu, realizaci a optimalizaci byznys procesů, tedy procesů, které podporují cíle společnosti, její systémy, zaměstnance, zákazníky a partnery v rámci společnosti i mimo ni [3].

Výše zmíněná definice by se tedy dala parafrázovat ve zkráceně formě jako, business process management zahrnuje pojmy, metody a techniky sloužící k návrhu, správě, zavedení a analýze business procesů.

Důležitým aspektem BPM je propojení s informačními technologiemi. Na BPM se dá nahlížet jako na dva odlišné aspekty, jako na manažerskou disciplínu, ale také jako na softwarové inženýrství [6]. Procesy se dají zpracovávat pomocí software. Tento software se obecně nazývá Business process management system. Busines process management system je obecný softwarový systém, který zpracovává jasně definované reprezentace procesu a koordinuje jejich průběh [4].

Běh procesu a koordinace jeho průběhu je pouze jedním z kroků životního cyklu procesu. Pokud chceme softwarově obsloužit i návrh, modelování, monitorování, optimalizaci, musíme použít více nástrojů. Společnosti, které poskytují celkové řešení pro BPM tento balík software nazývají Business process management suite.

2.3.1 Životní cyklus BPM

Abychom získali celkové řešení pro BPM, je potřeba dokázat obsloužit všechny kroky životního cyklu BPM. Skládá se z fází, které spolu vzájemně souvisí. Těmito fázemi jsou: návrh, modelování, zavedení, monitorování, optimalizace [5]. Na obrázku (Obr. 1) je vidět, že fáze jsou organizovány do cyklické struktury, která ukazuje jejich logické návaznosti. Tyto návaznosti nejsou, ale zcela striktní. Návrhové a vývojářské aktivity mohou být prováděny během každé z těchto fází. Vývojové přístupy zahrnující souběžné činnosti v několika fázích nejsou neobvyklé. Životní cyklus procesu zahrnuje tyto fáze:

- **Návrh**

V této fázi je definován business process, to zahrnuje znázornění průběhu procesu se všemi souvisejícími detaily, jako jsou účastníci procesu, zdroje, upozornění a oznámení, eskalace. Dále jsou identifikovány důležité aktivity, promyšleny případné organizační změny.

- **Modelování**

Druhou fází je modelování byznys procesu. V této fázi je byznys proces plně specifikován. Průběh procesu je formalizován za pomocí nějaké notace a je vytvořena definice procesu. Dále jsou definovány proměnné procesu a identifikovány služby, které mohou být použity k provedení aktivity. Tato část je stěžejní pro nasazení procesu do prostředí workflow engine, použitá modelovací notace by měla podporovat automatizované zpracování.

- **Nasazení**

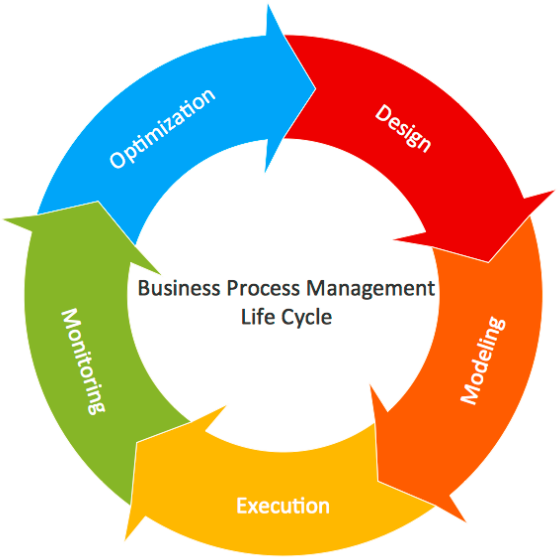
Vymodelovaný business process je nasazen. Tato fáze může zahrnovat organizační změny. Samozřejmě, že zamýšlíme nasazení za pomocí BPMS, kde jednotlivé instance procesů bude obsluhovat workflow engine.

- **Monitorování**

V této fází je potřeba definovat metriky, podle kterých bude hodnocena efektivita procesu. Dále je proces monitorován a jsou tedy sbírány statistické údaje, které slouží k vyhodnocení efektivity procesu vůči zvoleným metrikám.

- **Optimalizace**

Poslední fází je optimalizace, která na základě výsledků monitorování, změně požadavku či nových poznatcích identifikuje potřebné změny, nedostatky a problémy. Na základě této analýzy jsou navržená možná řešení a vylepšení. Cyklus pak přechází opět do fáze návrhu, kde jsou získané poznatky aplikovány a celý se opakuje.



Obrázek 1 Životní cyklus BPM

2.4 Business Process Modeling Notation

BPMN je jeden z řady modelovacích jazyků a notací sloužící pro vytvoření procesního modelu. Standart je vyvíjen Object Management Group, která stojí i za jazykem UML. Cílem této části není popsat celý standart, ale seznámit s principy.

Cílem BPMN je poskytnout notaci, která bude jednoduše pochopitelná pro všechny zainteresované strany, kterých se daný proces týká, počínaje analytiky, kteří proces navrhují, dále vývojáři, jenž jsou zodpovědní za implementaci technochnologií, které daný proces vykonávají či s ním souvisí, konče lidmi, kteří procesy řídí a monitorují [4].

BPMN se tedy snaží obsáhnout kompletní škálu abstrakce procesní analýzy až po technickou implementaci. Současná verze BPMN je 2.0 a zásadně se liší od verzí 1.x. První verze BPMN vyšla v roce 2004. Verze 1.x byla široce využívanou modelovací notací, která se soustředila pouze na grafickou vizualizaci procesu. Aby model tohoto procesu mohl běžet v prostředí workflow/process enginu, bylo zapotřebí převést tento model do jazyka, který umožňoval spuštění procesu v tomto prostředí do tzv. execution language, jako WS-BPEL (První jazyk, který byl používán pro běh procesů v workflow engine) [4].

BPMN 2.0 rozšiřuje notaci právě o tuto sémantiku, která umožňuje procesům, aby byly zpracovávány v BPMN 2.0 kompatibilní engine, díky tomu se z BPMN stal společný vyměnitelný formát, který může být vyměnitelný nejen mezi grafickými editory, ale také mezi workflow engines, které jsou BPMN 2.0 kompatibilní. Díky zmíněným vlastnostem BPMN sbližuje byznys a IT.

2.4.1 Elementy BPMN

Cílem této části je popsat principy BPMN prvků a demonstrovat jejich použití. Při popisu standartu vycházím z oficiální dokumentace BPMN a z dokumentací Activiti a jBPM [7; 8; 9]. Popisovány nejsou všechny elementy, ale pouze část elementů z třídy Common Executable, jedná se o podtřídu BPMN, která se zabývá prvky, jež mohou běžet v prostředí workflow engine. Jednotlivými prvky BPMN se dále zabývám při porovnávání způsobu, jakým s touto notací pracují srovnávané engine.

BPMN má čtyři základní kategorie elementů.

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

2.4.1.1 Flow objects

Tokové objekty (Flow objects) jsou hlavními stavebními prvky BPMN. Do této skupiny patří events (údálost), activities (aktivitiy) a getaway (brány). Aktivity představují činnosti vykonávané během procesu. Brány se používají pro reprezentaci slučování a větvení procesu v závislosti na definovaných podmínkách. Událost je stav, který může nastat v průběhu procesu a ovlivnit tak jeho průběh.

Events (Události)

Událost je stav, který může nastat v průběhu procesu a ovlivnit tak jeho průběh. V BPMN jsou události vizualizovány kruhem. Události se dají rozdělit na Catching a Throwing**.**

Za catching event je označována událost, do které když se dostane průběh procesu, tak zde bude čekat na nějakou spoušť. Typ spouště je určen pomocí ikony, co se vizuální stránky týče, nebo je definován v XML. Catching events jsou vizualizovány pomocí nevyplněné ikony.

Throwing event je událost, do které když se průběh procesu dostane, tak je spoušť okamžitě spuštěna. Throwing events jsou vizualizovány ikonou, která je vyplněna černě.

Dál se události dají kategorizovat také jako:

- Start
- End
- Intermediate

Způsob vizualizace v BPMN je vidět na obrázku (Obr. 1).

Obrázek 2 Typy událostí

Start event indikuje, kde proces začíná. Typ start event definuje, jakým způsobem proces začíná. To například může být na základě nějaké zprávy, časového intervalu a dalších. Základním type start event je none, což znamená, že typ spouště není nijak specifikován, tedy že proces začne nespecifikovaným způsobem.

XML reprezentace none start event vypadá následovně:

```
<startEvent id="request" activiti:initiator="initiator" />
```

Další kategorií událostí jsou End Events. Tento typ event značí konec procesu, či subprocesu. End Events jsou vždy throwing.

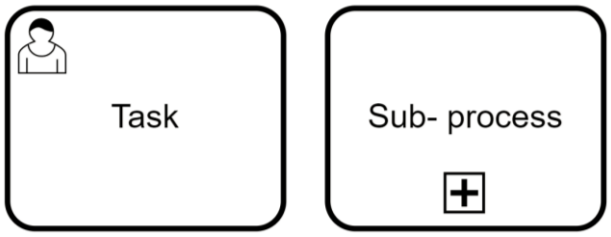
Posledním typem událostí jsou Intermediate Events. Intermediate Events reprezentují události, které mohou nastat během provádění procesu. Intermediate events se používají ke

zpoždění průběhu procesu [4]. Zpožděním je myšleno potřeba počkat na nějakou událost, které musí nastat, aby mohl proces pokračovat. Typickým příkladem je čekání na signál. Signal Intermediate Event je jak throwing tak i catching.

Intermediate Events mají svoji podtřídu Boundary Events. Boundary Events jsou catching events, které jsou vždy připojené k nějaké aktivitě. Když aktivita běží, Boundary Event s určitým typem spouště naslouchá a když nastane definovaná událost, aktivita je přerušena a proces pokračuje dále od této události.

Aktivities (Aktivitiy)

Aktivity jsou jednotky práce, které představují činnosti odehrávající se uvnitř procesu. Aktivity se dále rozdělují na Atomické aktivity a podprocesy. Atomické aktivity jsou dále nedělitelné úkony, nazývají se také úlohy (Task). Podproces (Sub-process) je proces, který je definovaný uvnitř rodičovského procesu. Základní vizualizace těchto BPMN prvků je na obrázcích níže (Obr. 3 a Obr. 4).



Obrázek 3 Základní vizualizace aktivit

Jednotlivé úlohy jsou dále specifikovány dle jejich účelu. Základní jsou:

- **Service Task** – Typ úlohy, která je implementována částí softwaru či webovou službu, která se postará o vykonání aktivity.
- **User Task –** Typ úlohy, která zahrnuje interakci uživatele**.**
- **Script task –** Úloha, která používá skriptovací jazyk k provedení jednoduché funkcionality.

Na obrázku (Obr. 2) je vizualizace aktivity typu *User Task* a jeho XML reprezentace.

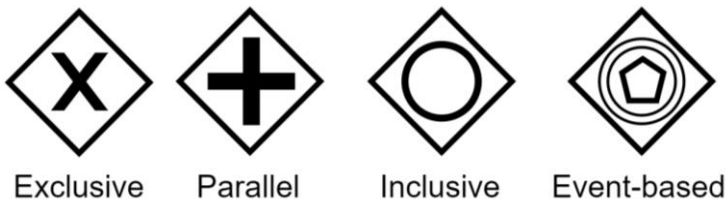
```
<userTask id="userTask" name="User task" />
```

Obrázek 4 User task

Getaway(Brány)

Brány jsou užívány k řízení průběhu procesu, k větvení jeho průběhu a zároveň slučování. Existuje několik typů bran. Z nich základní jsou tyto:

- **Exclusive –** Používá se k modelování rozhodnutí v rámci procesu, na základě podmínky proces pokračuje do dané větve
- **Parallel –** Paralelní brána se používá k modelování konkurence v procesu, respektive běh více větví procesu zároveň. Umožňuje rozdělit průběh do více větví a zároveň tyto větve opět spojit.
- **Inclusive –** Kombinace Exclusive a Parallel. Rozhodnutí probíhá na základě podmínky, ale pokud více větví splňuje definované podmínky, může proces probíhat paralelně více větvemi, stejně jako u paralelní brány.
- **Event-based –** Tento typ brány dokáže provést rozhodnutí na základě události. Průběh procesu se v této bráně zastaví a čeká, dokud nenastane jedna z událostí na základě, které může rozhodnout o pokračování do dané větve procesu.



Obrázek 5 Základní typy bran

2.4.1.2 Connecting Objects

Tento typ objektu navzájem spojuje flow objects.

- **Sequence flows –** Spojení mezi dvěma elementy procesu.

Obrázek 6 Spojení mezi elementy

2.4.1.3 Swimlanes

Tento typ objektu slouží k organizování a kategorizaci činností. Nemá vliv na průběh procesu jako takový.

- **Pool –** Hlavní organizační jednotka
- **Lane –** Slouží k další kategorizaci a organizaci v rámci hlavní organizační jednotky.

2.4.1.4 Artifacts

Do této kategorie spadají popisné objekty, které umožňují přidání dalších informací do modelu procesu, díky čemuž je model přehlednější.

3 Java open-source workflow engines

Jedním z dílčích cílů bakalářské práce je srovnat nejčastěji používané open-source alternativy s workflow engine Activiti, který je použitý k vytvoření referenční aplikace v praktické části práce. V této části bakalářské práce provádím rešerši na základě volně dostupných informací k daným workflow engines. Vybrané alternativy musí být z prostředí Javy a splňovat následující kritéria.

Srovnání workflow engines nemá sloužit k zjištění jaký engine je lepší, ale spíše k informování o stěžejních vlastnostech, podle kterých má být workflow engine vybírán.

3.1 Kritéria výběru

Výběr nejčastěji používané alternativy k workflow engine Activiti byl vybírán na základě těchto kritérií:

- Open-source licence
- Založené na Javě
- Lze je přímo integrovat do projektu
- Dokumentace
    - dostatek informací pro jeho použití
    - Javadoc
    - Průvodci
    - Tutoriály
    - Příklady
- Podpora BPMN 2.0
- Aktivita vývoje
    - Měřeno dle dat dostupných z openhub.net
- Velikost a aktivita komunity
    - Měřitelná fóry a aktivitou v nich.

Tyto kritéria splňuje jBPM a ten je dále porovnán s Activiti. Tyto dva však nejsou jediní v této kategorii. Kritéria částečně splňuje Eclipse Startdust, ale od roku 2016 se dále nevyvíjí. Dalším adeptem k srovnání byl Camunda BPM, jeho jádro však tvoří Activiti, které je již do výběru zahrnuto. Vhodným workflow engine ke srovnání se zdál být řešení Bonita BPM, to však muselo být vyřazeno kvůli malé dokumentaci a neaktivní komunitě.

Engines jsou vybrány tak, aby oba nabízely podobné možnosti. Tradiční workflow engines jsou stavěny především pro netechnické uživatele. Oba srovnávané engines mají dvojí pohled, míří jak na bussiness uživatele, tak na vývojáře, ve snaze propojit tyto dva světy.

3.2 Activiti

Activiti je workflow a Busines Process Management Platform. Její vývoj začal v roce 2010 a je financovaný Alfrescem, ale jedná se o nezávislý open-source projekt, distibuovaný pod Apache license [6]. Projekt založil Tom Baeyens a Joram Barrez, kteří se podíleli i na vývoji jBPM. Actviti je tedy založeno na zkušenostech, které získali při vývoji jBPM, ale není založené na společném kódu.

3.2.1 Activiti workflow engine

Jádro Activiti tvoří BPMN 2.0 workflow engine, který poskytuje běhové prostředí pro instance procesů. Ve své podstatě je activiti engine stavový automat.

Součástí activiti engine je abstrakce *process virtual machine*, která transformuje BPMN 2.0 definici procesu do podoby stavového automatu. Díky process virtual machine, respektive, díky abstrakci, kterou poskytuje, umožňuje engine podporovat další jazyky pro definici procesu.

Activiti může běžet v jakékoliv Java aplikaci, či na serveru, podporuje také škálovatelnost engine a jeho nasazení v clustreru, či cloudu.

3.3 jBPM

JBPM je Business Process Management Suite. Jedná se o zcela open-source platformu, která je vyvíjená jBoss komunitou a vývoj sponzoruje Red Hat. jBPM je distribuováno pod Apache

Software Apache License [9]. JBPM je součástí projektu Drools, jedná se o Business Logic integration Platform. Tato platforma poskytuje společné prostředí pro workflow, zpracování událostí, automatické plánování a byznys pravidla [10]. Díky integraci s dalšími součástmi projektu Drools lze rozšířit možnosti jBPM.

3.3.1 jBPM engine

Srdcem jBPM je workflow engine napsaný v Javě [9]. Autoři ho popisují jako "light-weight", jelikož může běžet v jakémkoliv zařízení, které podporuje Java Runtime Enviroment a také zdůrazňují jeho spolehlivost a stabilitu.

Může být vestavěný jako součást jakékoliv Java aplikace, nebo může běžet jako samostatná služba na serveru.

Podobně jako Activiti je implementovaný jako obecný stavový automat a může být tedy rozšířen o podporu více modelovacích jazyků.

3.4 Srovnání workflow engines

V této části porovnávám workflow engine Activiti a jBPM.

3.4.1 Metodika srovnání

Při srovnání engine čerpám především z dokumentace k daným engines. Při nejasnostech v dokumentaci jsou některé aspekty srovnání testovány přímo nasazením engine. Dále bylo čerpáno ze související literatury k jednotlivým engines.

3.4.2 Kritéria srovnání

Při srovnání se snažím především ukázat, jaké je spojení mezi vývojářem a bussines analytikem. Z tohoto pohledu je jedním z hlavních kritérií implementace modelovacího jazyka, tedy jaké možnosti má analytik při modelování procesu, respektive jaké BPMN prvky je možné implementovat. Dále se zaměřuji na technologie, se kterými nativně spolupracují a další vlastnosti, které napomáhají k jeho použití. Z pohledu vývojáře je relevantním kritériem

náročnost na integraci engine do projektu. S tím souvisí i zhodnocení úrovně dokumentace a dalších informací, které jsou k jednotlivým engines dostupné.

3.4.3 Implementace modelovacího jazyka

Předmětem srovnání v této části je implementace jazyka pro modelování procesů.

Oba engines implementují jazyk BPMN 2.0, jBPM navíc přidává svůj vlastní jazyk jPDL. Ani jeden z engines nedefinuje všechny elementy a atributy, které jsou definovány v BPMN 2.0 specifikaci. Podporují pouze určitou podmnožinu, která zahrnuje elementy a atributy, jež jsou považovány za běžné, přesněji, zahrnují téměř všechny elementy, které jsou definovány v podtřídě BPMN *Common executable* a ta se zabývá modely, které mohou běžet v prostředí workflow engine.

Oba však využívají principu *Process Virtual Machine,* která překládá proces namodelovaný v BPMN 2.0 do podoby stavového automatu, díky tomu je případně možné použít pro modelování procesů jiný jazyk. BPMN 2.0 je pro modelování procesů dominantním standardem, a proto ho oba engines používají.

Základní pohled Activiti a jBPM na BPMN se liší. Zatímco jBPM se striktně drží a pracuje s XML definicí procesu tak, jak specifikuje BPMN, Activiti považuje některé konstrukce v XML za těžkopádné, a proto přináší své rozšíření *Activiti BPMN extensions.* [8] To však neznamená, že Activiti neumožňuje používat BPMN standardizovanou cestou.

Activiti umožňuje tedy pracovat s BPMN dvěma způsoby. Při použití Activiti rozšířením BPMN, je nutno uvádět namespace prefix “activiti” při definování jednotlivých XML elementů a atributů. To znamená, že lze oba přístupy i kombinovat, některé části mohou být řešeny standardizovaným způsobem a jiné za pomoci rozšíření. Cílem Activiti rozšíření BPMN je zjednodušit a zefektivnit zápis XML definice procesu. Rozšíření je vyvinuto tak, aby bylo možné jednoduše transformovat XML procesní definici do standardizované podoby.

Rozdíl nepanuje pouze v přístupu ke standartu, ale také, jak již bylo naznačeno, v množině prvků, jaké daný engine podporuje. Při pohledu na podporované prvky BPMN dle kategorií, jBPM podporuje více prvků z kategorie *Event*(Události), u jBPM jsou tedy větší možnosti, jak

reagovat na události, které mohou nastat během procesu. Co se týče prvků z kategorie *Activities*(Aktivity) jsou na tom oba engines při standartní práci s BPMN podobně. BPMN rozšíření Activiti přidává implementaci některých aktivit, které nejsou součástí BPMN standartu a jsou běžně implementovány jako *Service Task*. Takovou aktivitou je například *Email Task*. Další kategorie BPMN prvků jsou analogické. Activiti ještě nad rámec BPMN přidává tzv. listeners, tedy posluchače, kteří jsou schopni reagovat na události, které nastanou během procesu a v reakci provést implementovanou logiku, díky tomu Activiti dohání nedostatky, které má u prvků typu Event, avšak nestandartní cestou.

3.5 Verzování definic procesu

Oba engine poskytují možnost verzovat definice procesů. Umožňují běžet více verzím stejného procesu zároveň. To se hodí v době, kdy probíhá přech na novou verzi procesu. JBPM oproti Activiti dokáže i migrovat bežící procesy na nové verze.

3.6 Integrace s dalšími technologiemi

Oba engines je možné integrovat s dalšími technologiemi z prostředí Javy.

3.6.1 Podpora integrace technologií

Jelikož se jedná o Java workflow engine, tak je možné při vývoji použít různé frameworky a technologie z tohoto prostředí. Společná integrace obou workflow engines s populárními frameworky je snadná. Některé technologie jsou v rámci engine nativně podporovány, engine pro tyto technologie poskytují knihovny, ve kterých využívá koncepty dané technologie k usnadnění její integrace s workflow engine. V tabulce (Tab. 1) je porovnáno

	Activiti	**jBPM**
CDI	Ano	Ano
EJB	Částečně	Ano
OSGi	Ne	Ano
Spring	Ano	Ano

Tabulka 1 Nativně podporované technologie

3.6.2 Implementované typy úloh

Součástí procesů, jsou aktivity, které vykonávají dílčí kroky procesu a k provedení daného kroku jsou potřeba různé technické prostředky. Základní typem úlohy je *Service task*, který lze implementovat dle potřeby. Oba engines však pro některé technologie poskytují již účelově implementované *Service tasks*, které umožňují okamžité nasazení. V tabulce (Tab. 2) jsou uvedeny implementované typy úloh.

	Activiti	**jBPM**
Business rule task	Ano	Ano
Email task	Ano	Ne
Web service task (WSDL)	Ano	Ne
REST service task	Ne	Ano
Mule Task	Ano	Ne
Camel Task	Ano	Ne

Tabulka 2 Implementované typy úloh

JBPM je součástí projektu Drools a jedná se o Business Logic integration Platform. Tato platforma poskytuje společné prostředí pro workflow, zpracování událostí, automatické plánování a byznys pravidla [10]. Díky integraci s dalšími součástmi projektu Drools lze rozšířit možnosti jBPM.

3.7 Obtížnost zvládnutí technologie

Zhodnocení obtížnosti zvládnutí technologie je těžké zhodnotit nesubjektivně. Rozjetí tzv. „Hello world“ příkladů má v obou případech podobnou obtížnost a časovou náročnost. Další práce je však s jBPM kvůli napojení na projekt Drools celkově těžkopádnější, s tím souvisí rozsáhlejší API, které je nad rámec potřeb workflow engine, oproti Activiti, které se soustředí pouze na řízení procesů. Se zvládnutím dané technologie souvisí i přístup k dokumentaci a k dalším dostupným zdrojům.

3.8 Dokumentace

Dokumentace u obou workflow engines je na velmi dobré úrovni. Oba engines poskytují uživatelskou příručku, tutoriály a příklady použití, dále je dostupná i literatura. Oba engines mají rozdílný přístup ke zpracování dokumentace a dalších informací k engine. JBPM cílí spíše na zkušenější uživatele, zatímco Activiti popisuje i elementární konstrukty.

3.9 Aktivita komunity a vývoje.

Na základě dat dostupných z openhub.net, je zjevné že u obou workflow engine je aktivní vývoj. U jBPM je aktivita vývoje vyšší, ale to souvisí s komplexností engine a jeho napojením na další části, které přímo nesouvisí s procesy. Oba engine mají aktivní komunitu. Na příslušných fórech často odpovídají přímo vývojáři.

3.10 Shrnutí

V tabulkách níže je uvedeno je uvedeno shrnutí, které vychází z předchozího textu. V tabulce (Tab. 3) jsou uvedeny základní informace.

	Activiti	**jBPM**
Licence	Apache License 2.0.	Apache License 2.0
Hlavní vývojáři	Alfresco	Red Hat
Zaměření	Pro účely řízení procesů	Komplexní, propojení s projektem Drools

Tabulka 3 Základní informace

V následující tabulce (Tab. 4) je shrnut způsob práce s BPMN.

	Activiti	**jBPM**
Práce s BPMN	Dle standartu, nebo pomocí vlastního rozšíření.	Dle standartu.
Škála podporovaných BPMN elementů	Většina elementů z podtřídy Common executable.	Větší množina elementů z podtřídy Common executable než u Activiti.
Verzování procesů	Verzování procesů, běh více verzí stejného procesu zároveň.	Stejně jako Activiti. Navíc možnost migrovat běžící procesy na nové verze.

Tabulka 4 Práce s procesy v BPMN

V tabulce (Tab. 5) je uvedena podpora technologií a dalších nástrojů.

	Activiti	**jBPM**
Implementované úlohy, připravené k použití	Business rule, Email, Web Service, Mule a Camel task	Business rule a REST service task
Integrace frameworků a technologií pro	CDI, EJB, Spring	CDI, OSGi, EJB, Spring
Podporované databáze	H2, MySQL, Oracle, PostgreSQL, DB2 MSSQL	MySQL, Oracle, PostreSQL, DB2, MSSQL, Sybase, Maria DB
Přidané nástroje	Webová platforma pro základní práci. Elipse plugin pro modelování procesů.	Webová platforma pro základní práci. Elipse plugin pro modelování procesů.

Tabulka 5 Podpora technologií a dalších nástrojů

V následující tabulce (Tab. 6) jsou uvedeny kritéria související se zvládnutím technologie a vhodností jeho použití z pohledu podpory vývoje a komunity.

	Activiti	**jBPM**
Dokumentace a další informační zdroje	Určena pro středně pokročilé uživatele.	Určeny pro zkušené uživatele.
Obtížnost zvládnutí technologie.	Snadnější, díky dokumentaci a vymezení pouze na procesy.	Komplexní kvůli napojení na projekt Drools.
Aktivní vývoj	Ano	Ano, vyšší než Activiti
Aktivní komunita	Ano	Ano

Tabulka 6 Podpora vývoje

4 Praktická část

Smyslem praktické části bakalářské práce je vytvoření vzorové workflow aplikace, která implementuje žádost o přidělení zdroje. Při popisu návrhu a vývoje aplikace se zaměřuji především na vysvětlení částí týkající se právě workflow. Součástí je také vytvoření uživatelského rozhraní pomocí moderních webových technologiích. Po praktické části následuje zhodnocení využití workflow engine a jsou zhodnoceny výhody a nevýhody použití workflow engine oproti vytvoření podobné aplikace bez něj.

4.1 Specifikace aplikace

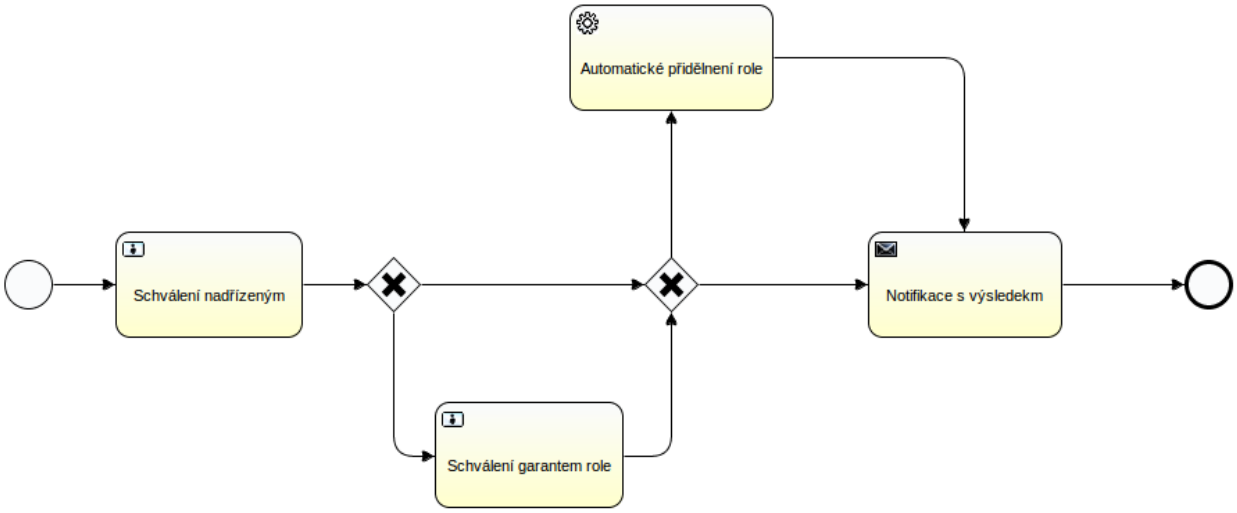
Aplikace musí poskytovat možnost vytvořit žádost, schvalovat ji, zobrazit průběh jejího řešení a zobrazit seznamy žádostí a úkolů celkově. S tím souvisí i vytvoření uživatelského rozhraní pomocí moderních webových technologií, jehož prostřednictvím se budou žádosti zpracovávat.

4.2 Referenční aplikace

Pro demonstrativní účely je vytvořena aplikace, která spočívá ve správě zaměstnanců a jejich rolí. Předmětem žádosti v aplikaci je tedy role. V aplikaci jsou dva typy rolí, interní a chráněná. Každý zaměstnanec může podat žádost o roli, která je následně zpracovávána a na základě rozhodnutí o schválení žádosti dojde k případnému automatickému přidělení role.

4.2.1 Proces zpracování žádosti

Celý proces začíná podáním žádosti, další průběh závisí na typu role, pokud je role interní, tak ke schválení postačí schválení nadřízeným, pokud je role chráněná, tak je potřeba schválit žádost i garantem role. Jestliže je daná žádost schválená, tak je role automaticky přidělena, při zamítnutí žádosti proces přejde rovnou k odeslání emailové notifikace s výsledkem žádosti, ve které je zaměstnanec informován o přidělení, či nepřidělení role a může být také přiložen text se zdůvodněním výsledku procesu. Tento proces je znázorněn na obrázku (Obr. 7).



Obrázek 7 Proces žádosti

4.3 Návrh

4.3.1 Architektura aplikace

Architektura aplikace je navržena jako klient-server. Aplikace je rozdělena do dvou samostatných celků, klientské a serverové části. Komunikace mezi klientem a serverem probíhá pomocí REST API. Celková architektura aplikace je rozdělena do tří vrstev, prezentační, servisní, a datová.

Prezentační vrstva

Klient komunikuje se serverem pomocí REST API. Na straně serveru jsou tedy vytvořeny endpoitny, které požadavky delegují do servisní vrstvy, kde jsou provedeny požadované operace, pokud jsou výstupem těchto operací doménové objekty, tak jsou tyto objekty převedeny na tzv. view modely, tedy objekty, které obsahují pouze data, jenž mohou být zobrazena na prezentační vrstvě. Tyto data jsou poslána na klienta ve formátu JSON, kde s nimi klient dále pracuje. Část prezentační logiky je tedy přesunuta do klientského prohlížeče.

Servisní vrstva

Servisní vrstva je prostřední vrstva mezi datovou a prezentační vrstvou. Tato vrstva obsahuje veškerou logiku aplikace a také vytváří komunikační kanál mezi vrstvou prezentační a datovou. Z hlediska datové vrstvy je také nezbytná pro řízení transakcí.

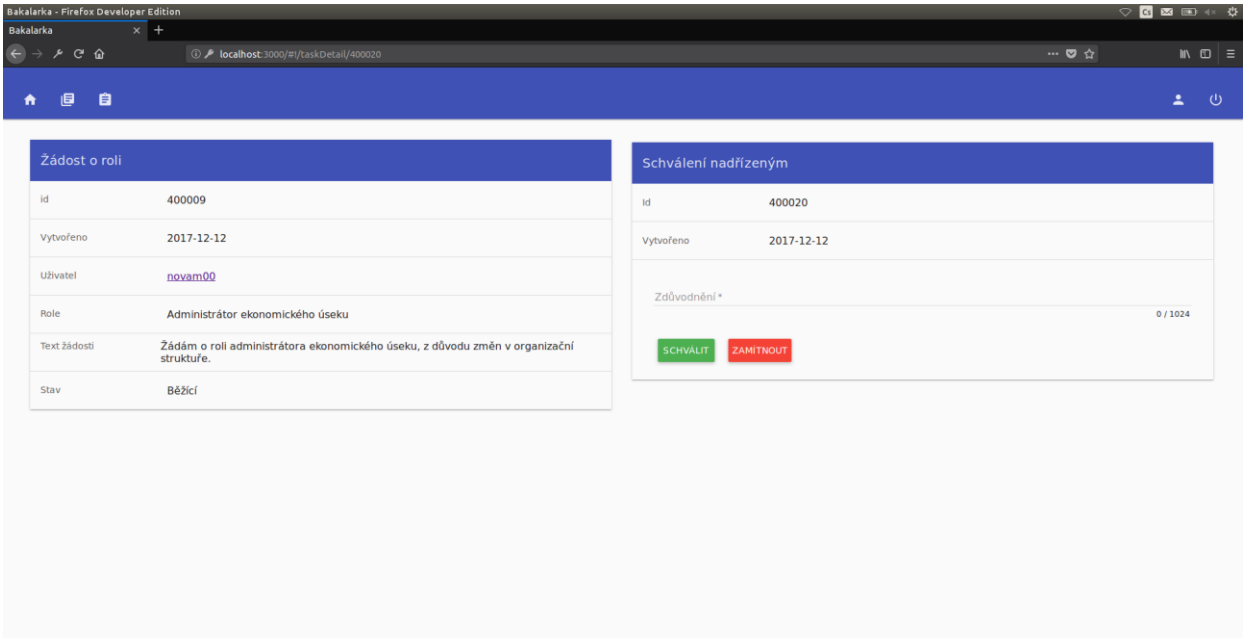
Datová vrstva

Datová vrstva slouží ke komunikaci s databází. Activiti engine k uchovávání svých dat používá relační databázi a pro tyto data provádí objektově relační mapování.

4.3.2 Klient

Uživatelské rozhraní je vytvořeno jako *single-page aplikace,* prostřednictvím javascriptového frameworku AngularJS. Důvodem výběru AngularJS je, že umožňuje vytvořit přehledný kód, který je možné rychle a jednoduše vyvíjet.

Další použitý framework je AngularJS Material. Komponenty tohoto framework slouží k rychlému vytvoření material designu dle Google specifikace. [11] Celý design uživatelského rozhraní je responzivní a je tedy optimalizovaný pro různá zařízení.



Obrázek 8 Uživatelské rozhraní

4.3.3 Server

V této části je popsán návrh serverové části aplikace, ve které je nasazen workflow engine. Jsou zde popsány použité technologie a návrh balíčků.

4.3.3.1 Použité technologie

V této části jsou stručně popsány technologie, které byly využity při vývoji aplikace a je nastíněno jejich využití v aplikaci. Activiti, je na javě založený workflow engine, tudíž i další použité technologie jsou založené na javě, či s ní dokáží kooperovat.

Apache Maven

Maven popisuje softwarový projekt pomocí tak zvaného *Project Object Model*. Projekt je popsán nejen z pohledu jeho zdrojového kódu, ale také včetně závislostí na externích knihovnách a popisu procesu sestavení a s tím spojených funkcí [12].

Spring

Klíčovou technologií aplikace je framework Spring. Spring posyktuje komplexní infrastrukturu pro vývoje java aplikací, díky tomu se lze soustředit pouze na vývoj logiky aplikace [13]. Infrastrukturou je myšleno přebírání zodpovědnosti za běžně se opakující funkce aplikace, kdy aplikace tyto funkce pouze využívá. Programátor se pak může soustředit pouze na logiku a zbytek nechat na framework, jenž jeho aplikaci využívá.

Zásadním rysem Springu je návrhový vzor *Inversion of Control*. Ten uvolňuje pevné vazby mezi jednotlivými komponentami a přesouvá odpovědnost za vznik vazeb, závislostí, na framework. Vznik závislosti mezi jednotlivými komponentami systému je řešen pomocí *Dependency injection,* což česky znamená vkládání závislostí.

V aplikaci je využíván Spring boot. Spring boot je založen na Spring, díky Spring boot je možné vytvářet rychle a jednoduše na Spring založené aplikace s minimální námahou [13]. Spring boot využívá myšlenku: *convention over configuration*, tedy upřednostňuje konvence před konfigurací, je nutné specifikovat pouze nestandartní části aplikace.

Activiti spolupracuje s vývojáři frameworku Spring a poskytuje knihovny, které jsou navržené ke kooperaci s framework Spring. Spring v aplikaci dále zajišťuje integraci REST API a JPA.

Hibernate

Hibernate je framework provádějící objektově-relační mapování. Hibernate poskytuje implementaci specifikace Java Persitance API (JPA). JPA je specifikace, které popisuje, jakým způsobem se mají převádět data mezi modelovými třídami a relační databází [14].

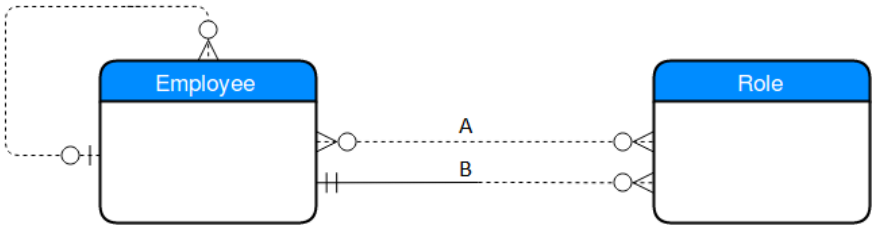
PostgreSQL

PostgreSQL je open-source objektově-relační databázový systém a je podporovaná Activiti, jako jedna z možných databází pro uchování dat.

4.3.3.2 Datový model aplikace

V aplikaci je potřeba modelovat entitu pro zaměstnance a roli. Entitou je myšlen libovolný objekt z reálného světa se shodnou strukturou vlastností, který je zachycen v datovém modelu.

Zaměstnanec je v aplikaci modelován entitou *User* a role objektem *Role*. Vztah těchto objektů je zachycen na entitně relačním diagramu, obrázek číslo (Obr. 7).



Tabulka 7 Datový model

Na entitně relačním diagramu je znázorněno, že zaměstnanec může mít nadřízeného a jeden nadřízený může mít více podřízených. *User* a *Role* mají mezi sebou dvě relace, relace A vyjadřuje, že zaměstnanec může mít libovolný počet rolí a danou roli může mít samozřejmě i více zaměstnanců zároveň. Relace B modeluje garanta dané role. Role musí mít právě jednoho garanta a zaměstnanec může být garantem libovolného počtu rolí.

Žádost modeluje samotný workflow engine, respektive jeho datový model.

Pro přidání vlastních entit a jejich objektově-relační mapování je přidána podpora pro Java Persintence API (JPA). Jako implementace JPA je použito Hibernate. Pro tyto entity jsou definovány DAO objekty pro práci s daty těchto entit.

4.3.3.3 Návrh balíčků

Aplikace je rozdělena do těchto balíčků.

- Main
- Model
- Model.TransferObjects
- DAO
- Services
- Controllers

Main

Hlavní balíček, obsahuje třídu *App* s metodou *main.* Třída *App* je označená anotací *@SpringBootApplication,* která umožní definovat *Spring beans* pomocí java konfigurace a přidat *beans* které jsou definované v classpath, provede tedy tzv. component scan a přidá nejen konfigurace ale také další komponenty Spring. Je blíže specifikováno v části zabývající se implementací.

Model

V tomto balíčku jsou obsaženy třídy reprezentující entity *User* a *Role.* Pro tyto entity je prováděno objektově-relační mapování pomocí JPA, třídy tedy obsahují JPA anotace, které specifukují jakým způsobem se mají objekty těchto tříd mapovat do databáze.

Další část modelu je tvořena třídami, které již implementuje samotné Activiti a nejsou tedy obsahem balíčku *Model*. Jsou to tyto třídy:

- *ProcessInstance* z balíčku org.activiti.engine.runtime.ProcessInstance
    - Třída reprezentuje probíhající instanci procesu
- TaskInstance z balíčku org.activiti.engine.runtime.TaskInstance

    - Třída reprezentuje dílčí úkol procesu
- HistoryProcessInstance z balíčku org.activiti.engine.history.HistoricProcessInstance
    - Třída reprezentuje dokončené instance procesu

TransferObjects

Tento balíček obsahuje třídy, které předávají data mezi vrstvou prezentační, servisní a naopak. Pokud uživatel chce získat informace z datového modelu, tak jsou z instancí modelových tříd vytvořeny tyto *transfer objekty* a ty jsou následně serializovány do formátu JSON a odeslány na klienta. V obráceném pořadí to funguje, když je potřeba získat data od uživatele. Uživatel odešle na serverovou stranu data ve formátu JSON a data obsahují atributy, které jsou potřebné k vytvoření instance daného *transfer objektu a* instance je tak automaticky vytvořena. Z daného transfer objektu je následně vytvořena instance dané modelové třídy. O automatickou konverzi Java objektů do JSON a naopak se stará Spring.

DAO

V tomto balíčku jsou obsaženy *data access object* pro entity *User* a *Role.* Třídy v tomto balíčku slouží pro získávání dat zmiňovaných entit.

Services

V balíčku jsou obsaženy třídy vykonávající aplikační logiku. Některé z těchto servisních tříd aplikace kooperují s Activiti workflow engine skrze jeho servisní třídy. Jsou to:

- RequestService
- RequestTaskService

Další třídy v tomto balíčku obstarávají práci s modelovanými entitami *Role* a *User*. Jedná se o následující třídy:

- UserService
- RoleService

Controllers

Balíček obsahuje třídy, které zpracovávají požadavky ze strany klienta a ty delegují do servisních tříd. Jsou to třídy:

- RequestController
- TaskController

ServiceTasks

V tomto balíčku je třída vykonávající přidání role. Třída:

- AddRole

Z pohledu *Activiti* je tato třída *Java Service Task.* Je tedy přímou součástí workflow, jako jedna z jeho aktivit.

Security

Tento balíček obsahuje třídu:

- SecurityConfig

Třída slouží pro základní autentizaci k aplikace.

4.4 Implementace

Zde je popisován vývoj serverové části aplikace, zaměřuji se především na části týkající se zpracování žádosti, respektive zpracování procesu, s tím souvisí i útržky kódu, které jsou přidány pro ilustraci. Jsou vysvětleny použité závislosti, popsána implementace žádosti a je nastíněno, jak souvisejí její jednotlivé části s aplikací. Dále je popsána integrace workflow engine s ostatními komponentami aplikace.

4.4.1 Závislosti

Pro vývoj aplikace dle návrhu a vybraných technologií bylo potřeba přidat následující závislosti:

- org.postgresql.postgresql - závislost přidává JDBC konektor pro databázi postreSQL.

- org.activiti.activiti-spring-boot-starter-basic - tato závislost přidá všechny potřebné závislosti frameworku springboot a workflow engine Activiti. Vytváří konvenční konfiguraci ProcessEngine. Umožní automatické vytvoření *ProcessEngine* a zpřístupnění všech Activiti *service* jako *Spring beans.* Dále automaticky nasazuje definice procesu do engine.
- org.springframework.boot.spring-boot-starter-web - tato závislost slouží pro vytoření REST API.
- org.activiti.activiti-spring-boot-starter-jpa - závislost přidává podporu JPA. V aplikaci slouží pro provádění objektově relačního mapování pro entity *Role* a *User.*
- org.codehaus.groovy.groovy-all - tato závislost umožňuje spustit kód programovacího jazyka Groovy, který je zapsán v definici procesu.

4.4.2 Žádost

V activiti není potřeba aby samotná žádost měla přímo svoji entitu. Pro nasazení do prostředí Activiti, je podstatný XML popis procesu, jak bylo popsáno v teoretické části. XML definice procesu je zodpovědná za strukturu a chování každého kroku procesu. Definice procesu je v Activiti vlastně Java protějšekem BPMN 2.0 procesu. Uchování žádosti, respektive jakýkoliv procesů v prostředí Activiti, obstarává *repository service* a třída *ProcessDefinition* z balíčku *org.activiti.engine.repository,* která reprezentuje spustitelný proces. Activiti také přidává nad rámec možností BPMN verzování definic procesu. Activiti tedy umožňuje běh více verzí procesu zároveň. Jednotlivé procesy spouští *runtime service* a běžící instance procesů jsou v Activiti reprezentovány třídou *ProcessInstance* z balíčku *org.activiti.engine.runtime*.

Jak je z výše zmíněného textu patrné, žádost je realizována jako definice procesu s odpovídajícím popisem. Definice procesu byla vytvořena na základě konzultací se školitelem. Při popisu vycházím z teoretické části, kde byl konstrukt BPMN vysvětlen.

Každá definice procesu začíná elementem *process,* klíčovým atributem tohoto elementu je *id,* který identifikuje definici procesu a workflow engine podle něho vybírá jaký proces má být spuštěn.

```
<process id="roleRequest" name="Role request" isExecutable="true">
```

Celý proces začíná *startEvent*, aplikaci proces odstartuje, když uživatel podá žádost.

```
<startEvent id="startevent" name="Start"></startEvent>
```

Pro realizaci žádosti je nutné při podání žádosti dodat tyto proměnné:

- role
  - id
  - name
  - type
  - guarantor
- user
  - username
  - manager
- requestText

S těmito proměnnými aplikace vytvoří novou instanci procesu. Hodnoty těchto proměnných se již během procesu nemění.

Další proměnné jsou inicializovány až při běhu procesu a jejich hodnota závisí na jeho průběhu.

- approvalByManager – booleanovská proměnná indikující schválení či zamítnutí žádosti nadřízeným.
- approvalByGuarantor – booleanovská proměnná indikující schválení či zamítnutí žádosti garantem.
- state – stav žádosti, žádost bude moci nabývat těchto stavů
  - RUNNING – žádost je podaná a zpracovávána
  - APPROVED – tento stav indikuje, že žádost byla schválená
  - REJECTED – žádost byla zamítnutá
  - CANCELED – zrušená žádost, uživatel sám žádost zrušil
- approvalByManagerReason – zdůvodnění schválení či neschválení od nadřízeného
- approvalByGuarantorReason – zdůvodnění od garanta

O inicializaci proměnné *state* se startá *executionListener*, který při startu procesu tyto proměnné inicializuje.

```
<extensionElements>
   <activiti:executionListener  event="start"
       class="org.activiti.engine.impl.bpmn.listener.ScriptExecutionListener">
       <activiti:field name="script">
           <activiti:string>
               execution.setVariable('state', 'RUNNING');
           </activiti:string>
       </activiti:field>
       <activiti:field name="language" stringValue="groovy" />
   </activiti:executionListener>
</extensionElements>
```

Prostřednictvím *ScriptExecutionListener* je možné provést logiku nadefinovanou uvnitř definice procesu.

Průběh procesu se po *startEvent* přesouvá k první schvalovací aktivitě. Schvalovací aktivity jsou v BPMN diagramu reprezentovány jako *UserTask. UserTask* se používá, když je potřeba indikovat, že daná aktivita bude provedena nějakou osobou. Při podání žádosti o jakoukoliv dostupnou roli v aplikaci, první schvalování řeší nadřízený zaměstnance, který danou žádost podal. Tato aktivita je definována následovně:

```
    <userTask id="approvalByManager" name="Approval by manager">
       <extensionElements>
           <activiti:taskListener event="complete"
class="org.activiti.engine.impl.bpmn.listener.ScriptTaskListener">
               <activiti:field name="script">
                   <activiti:string>approved</activiti:string>
               </activiti:field>
               <activiti:field name="language" stringValue="groovy" />
               <activiti:field name="resultVariable" stringValue="approvalByManager" />
           </activiti:taskListener>
       </extensionElements>
    </userTask>
```

V aplikaci si nadřízený může zobrazit seznam žádostí jeho podřízených a tyto žádosti řešit. Součástí zpracování aktivity typu userTask, je možnost přidávat nové proměnné procesu. V rámci zpracování aktivity je přidána booleanovská proměnná *approved* a *reason*, která indikuje dílčí schválení či zamítnutí žádosti. Součástí obou schvalovacích aktivit je také rozšiřující element: `<extensionElements>`. Ten zajišťuje to, aby šlo pracovat s jednotlivými schvalovacími aktivitami stejným způsobem. V rámci úkolu se proměnná jmenuje *approved*, ale v rámci celého procesu

approvalByManager. Díky tomu je možné zpracovávat a zobrazovat schvalovací úkoly stejným způsobem a zároveň řídit proces tak jak byl zamýšlen.

Po této aktivitě se průběh procesu přesouvá do první brány tzv. exclusive gateway, kde se proces větví. Zde další průběh procesu závisí na typu role, o kterou se žádá a zda nadřízený žádost schválil. Pokud byla předmětem žádosti chráněná role a nadřízený žádost schválil, proces pokračuje ke schválení garantem role. Definice této podmínky vypadá následovně:

```
<exclusiveGateway id="isProtectedAndApproved" name="Exclusive gatawa"></exclusiveGateway>
<sequenceFlow id="ApprovalByGuaranteeFlow" sourceRef="isProtectedAndApproved"
     targetRef="ApprovalByGuarantee">
        <conditionExpression xsi:type="tFormalExpression">
               ${role.type == "PROTECTED" and approvalByManager == true}
        </conditionExpression>
</sequenceFlow>
```

Pokud byla žádost nadřízeným zamítnuta, či se jednalo o interní roli, průběh procesu se přesouvá do druhé brány.

```
<sequenceFlow id="flow3" sourceRef="isProtectedAndApproved" targetRef="exclusivegateway2">
</sequenceFlow>
```

V případě, že se průběh procesu dostaneme k *userTask approvalByGurantee,* tak je chování v aplikaci podobné první schvalovací aktivitě. Garant dané role si může zobrazit žádosti týkající se této role a ty následně zpracovávat. V rámci této schvalovací aktivity probíhá vše stejně jako u první. V rámci procesu je proměnná *approved* přidána jako *approvalByGuarantor*. Definice této aktivity vypadá následovně:

```
     <userTask id="approvalByGurantee" name="Approval by gurantee">
        <extensionElements>
            <activiti:taskListener event="complete"
class="org.activiti.engine.impl.bpmn.listener.ScriptTaskListener">
                <activiti:field name="script">
                    <activiti:string>result</activiti:string>
                </activiti:field>
                <activiti:field name="language" stringValue="groovy" />
                <activiti:field name="resultVariable" stringValue="approvalByGuarantor" />
            </activiti:taskListener>
        </extensionElements>
     </userTask>
```

Po této aktivitě se průběh procesu dostává do druhé brány. V této bráně další průběh procesu závisí na schválení či zamítnutí žádosti. Pokud byla žádost schválená, tak dojde k přidělení role. Tato podmínka je definovaná následovně:

```
<exclusiveGateway id="isApproved" name="Exclusive Gateway"></exclusiveGateway>
<sequenceFlow id="approvedFlow" sourceRef="isApproved"  targetRef="addRole">
       <conditionExpression xsi:type="tFormalExpression">
              ${approval == true}
       </conditionExpression>
</sequenceFlow>
```

Pokud je žádost zamítnuta tak proces pokračuje k odeslání notifikace uživateli s výsledkem žádosti.

```
<sequenceFlow id="flow8" sourceRef="isApproved" targetRef="sendNotification">
</sequenceFlow>
```

O přidělení role se stará *Java Service Task* s patřičnou logikou. Jeho definice vypadá následovně.

```
<serviceTask id="addRole" name="Add role" activiti:delegateExpression="${addRole}">
</serviceTask>
```

Implementace *Java Service Task* pro přidělení role je součástí aplikace. Bude vysvětleno dále.

Nakonec je uživateli odeslána notifikace s výsledkem žádosti, kde tedy obsah emailu závisí na schválení či zamítnutí žádosti. Tento *email task* není officiální součástí BPMN standartu, jedná se tedy o *Service Task* který implementovalo Activiti. Definice:

```
<serviceTask id="mailtask1" name="Mail Task" activiti:type="mail">
       <extensionElements>
         <activiti:field name="from" stringValue="roleRequest@ecompany.com" />
         <activiti:field name="to" expression="${user.email}" />
         <activiti:field name="subject" expression="Žádost o roli" />
         <activiti:field name="html">
                  <activiti:expression>
                         <![CDATA[
                               <html>
                                 <body>
                                       // Text emailu
                                 </body>
                               </html>
                            ]]>
                   </activiti:expression>
         </activiti:field>
     </extensionElements>
</serviceTask>
```

4.4.3 Aplikace

Jednotlivé části aplikace jsou poskládány díky možnostem Spring, tedy pomocí auto-detekce jednotlivých komponent, při konfiguraci založené na anotacích a scanování classpath, kde jsou takto anotované třídy vyhledávány a dále pomocí dependency injection vkládány tam, kde je

potřebné s nimi pracovat. Takovýto přístup je blíže specifikován při popisu jednotlivých typů komponent.

Stěžejní částí aplikace je integrace Activiti workflow engine do aplikace, tak aby bylo možné pracovat s Activiti API, které představuje workflow engine. To lze jednoduše udělat díky zmiňovaným závislostem, které zařídí, že se jednotlivé třídy vytvoří jako *Spring Beans* a je tedy možné je pomocí dependecy injection získat a dále s nimi pracovat. Práce s Activiti API je naznačena dále v ukázkách kódu.

Model

Po přidání závislosti byly vytvořeny třídy z balíčku *Model*. Třídy v tomto balíčku byly označeny JPA anotacemi kvůli provádění objektově relačního mapování. Dále byli vytvořeny třídy z balíčku TransferObjects, tyto třídy obsahují pouze atiributy, které jsou nutné zobrazit uživateli, či atributy, které jsou potřebné pro vytvoření instance třídy z balíčku model, podání žádosti či zpracování dílčích úkolů žádosti.

DAO

Pro práci s entitami *User* a *Role* byla vytvořena *data access objects.* Pomocí třídy *JpaRepository* s balíčku *org.springframework.data.jpa.repository.* Díky použití této třídy Spring automaticky implementuje požadované CRUD operace na základně jeho konvencí. Kód pro získání rolí vypadá následovně.

```
public interface RoleDAO extends JpaRepository<Role, Long> {
   Role findById(long id);
   List<Role> findAll();
}
```

Service

Stěžejní bussines logika je implementována v servisních třídách z balíčku *Services*. Tyto třídy jsou anotovány anotací *@Service.* Díky tomu, že je Activiti API zpřístupněno jako *Spring beans,* tak je lze jednoduše získat pomocí *dependency injection,* respektive anotace *@autowired*, která zajistí jejich vložení. Příkladem takovéto servisní třídy je třída *RequestService.* Ta se se stará o vytvoření žádosti a zobrazení žádostí.

Níže přikládám kód, který souvisí s vytvořením žádosti.

```
@Service
public class RequestService {
   @Autowired
   private RuntimeService runtimeService;
   @Autowired
   private UserService userService;
   @Autowired
   private RoleService roleService;

   /**
    * Metoda vytvoří žádost
    * @param requestTO identifikátor definice procesu
    * @param username uživatelské jméno uživatele který žádost podal
    * @param roleName jméno role o kterou je žádáno
    */
   public void createRequest(RequestCreateTO requestTO) {
        Map<String, Object> variables = new HashMap<>();
        Map<String, Object> role = new HashMap<>();
        Map<String, Object> user = new HashMap<>();
        Role requestedRole = roleService.findById(requestTO.getRoleId());
        User requestedUser = EmployeeService.findByUsername(requestTO.getUsername());
        role.put("id", requestedRole.getId());
        role.put("name", requestedRole.getName());
        role.put("type", requestedRole.getRoleType());
        role.put("guarantor", requestedRole.getGuarantee().getUsername());
        user.put("username", requestedUser.getUsername());
        user.put("manager", requestedUser.getManager().getUsername());
        variables.put("requestText", requestTO.getRequestText());
        variables.put("role", role);
        variables.put("user", user);
        runtimeService.startProcessInstanceByKey("roleRequest", variables);
    }
```

Metoda pro vytvoření žádosti je volána z příslušného *Controlleru.*

Controller

Třídy z balíčku *Controllers* jsou anotovány anotací *@RestController*. Metody této třídy dále obsahují anotaci @RequestMapping. Tato anotace slouží k mapování http požadavků na dané třídy, či metody. Následující ukázka ukazuje kód související s vytvořením žádosti a navazuje na předchozí ukázku servisní třídy.

```
@RestController
@RequestMapping(value = "/activiti/request")
public class RequestController {
  @Autowired
  RequestService requestService;
  @RequestMapping(value = "/create", method = RequestMethod.POST, consumes = "application/json")
    public ResponseEntity<Map<String, String>> create(@AuthenticationPrincipal final UserDetails
user, @RequestBody RequestCreateTO createRequest) {
        if(!createRequest.getUsername().equals(user.getUsername())) {
           return new ResponseEntity(new HashMap<>().put("Message", "FORBIDDEN"),
               HttpStatus.FORBIDDEN);
        }
        requestService.createRequest(createRequest);
        return new ResponseEntity(new HashMap<>().put("Message", "Request has created"),
            HttpStatus.OK);
    }
```

ServiceTask

Tento balíček obsahuje jednu třídu AddRole. Tato třída představuje logiku, která se provede, když se průběh procesu dostane k tomuto kroku. V rámci definice procesu je *ServiceTask* definován takto:

```
<serviceTask id="addRole" name="Add role" activiti:delegateExpression="${addRole}">
</serviceTask>
```

Součástí aplikace musí být implementována logika pro tuto aktivitu. Třída reprezentující tuto aktivitu musí být anotována spring anotací *@Component* a musí implementovat rozhraní *JavaDelegate* a překrývat metodu *execute* tohoto rozhraní. Logika obsažená v metodě *execute* bude provedena, až se průběh procesu dostane do této části. Kód vypadá následovně:

```
@Component
public class AddRole implements JavaDelegate {
   @Autowired
   UserService userService;
   @Override
public void execute(DelegateExecution de) throws Exception {
   Map<String, String> user = (Map<String, String>) de.getVariable("user");
   Map<String, Object> role = (Map<String, Object>) de.getVariable("role");
   employeeService.addRole(user.get("username"), (long) role.get("id"));
}
```

4.4.3.1 Práce s Activiti API

Díky Activiti API je možné velice snadno splnit všechny požadované operace, tedy

- vytvořit žádost
- schvalovat žádost
- zobrazit seznamy žádostí a úkolů celkově
- zobrazit průběh jejího řešení

Všechny třídy potřebné pro provedení všech zmiňovaných operací jsou z balíčku org.activiti.engine.

Vytvoření žádosti

O vytvoření žádosti, respektive procesu se stará třída RuntimenService a její metoda `startProcessInstanceByKey(String processDefinitionKey, Map<String,Object> variables).` Tato metoda spustí novou instanci procesu dle identifikátoru a přidá proměnné.

Schvalovat žádost

Ke schválení žádosti je potřeba třída TaskService a její metoda `complete(String taskId, Map<String,Object> variables)`. Pro zpracování musíme znát id úkolu a v případě referenční aplikace id schvalovací aktivity, kterou chceme zpracovat. Díky parametru `variables` je možné do procesu přidat další proměnné, které lze v procesu použít k dalšímu řízení. V případě referenční aplikace jsou do procesu přidány proměnné vyjadřující výsledek schvalovací aktivity a zdůvodnění tohoto výsledku.

Zobrazit seznamy žádostí a úkolů celkově

Zobrazení informací o žádostech a úkolů zajišťuje HistoryService. Pro získání seznamu procesů, či jednoho konkrétního, slouží metoda `createHistoricProcessInstanceQuery()`. Tato metoda vytvoří instance třídy `HistoricProcessInstanceQuery` z balíčku org.activiti.engine.history. Z této třídy se dá využít široká škála metod k dotazování na požadované procesy a umožňuje různě filtrovat procesy, včetně podle hodnot proměnných procesů. Velice podobně lze postupovat při získávání informací o dílčích aktivitách.

Zobrazit průběh řešení žádosti

Pro zobrazení průběhu procesu je nutné získat informace o daném procesu, zde je postup analogický k předchozímu bodu. Jedná se pouze o zobrazení informací o procesu a úkolech, ze kterých lze vyčíst, v jakém stavu se proces nachází.

4.4.4 Testování

Business procesy jsou nedílnou součástí aplikace a měly by být testovány stejně jako je testována logika aplikace, ale místo kódu je testován proces.

Activiti poskytuje možnost testovat procesy stejně, jako je možné testovat i další logiku aplikace pomocí unit testů. Proces je testován tak, že pomocí method Activiti API se projde celý jeho průběh a kontroluje se, zda každý úkol skončil dle očekávání. To lze zjistit tak, že proměnná procesu měla po dokončení úkolu očekávanou hodnotu a podobně.

5 Zhodnocení využití workflow engine

V této kapitole je zhodnoceno využití workflow engine. Pro účely srovnání je provedena analýza k vytvoření podobné aplikace zpracovávající žádosti bez využití workflow engine. Následně je provedeno srovnání systémů s využitím a bez využití workflow engine a jsou zhodnoceny výhody a nevýhody jeho nasazení.

5.1 Aplikace bez využití workflow engine

V této části je provedena analýza možného řešení pro vytvoření aplikace bez workflow engine. Je proveden návrh takového řešení a implementovány části podstatné pro srovnání obou přístupů. Výsledkem této části je získání poznatků pro řízení procesů bez využití workflow engine.

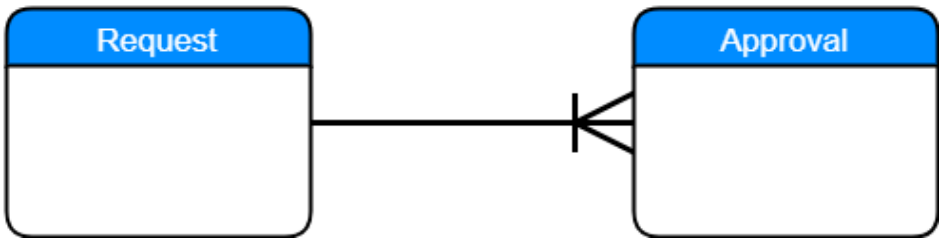
5.1.1 Analýza

Pro vytvoření podobné aplikace, které by mohla zpracovávat stejnou žádost bez využití workflow engine, je nutné přesně reflektovat zamýšlený průběh. Nejjednodušším přístupem je vytvoření konkrétního řešení přesně na míru požadavkům. Tento přístup se neodborně nazývá „*hardocrování”*. Stěžejní částí takového řešení je datový model reprezentující žádost. S tímto datovým modelem se v aplikace dále pracuje. Je nutné především zajistit, aby úpravy datového modelu probíhaly v takovém pořadí, aby co nejpřesněji reflektovaly zamýšlený průběh žádosti a dále vyřešit pozastavení automatizovaného zpracování a opětovné navázání na něj při zpracování aktivit, kde je potřeba lidská interakce, tedy u aktivit, které mají být zpracovány manuálně, v tomto případě schvalování žádosti.

5.1.2 Návrh

Zásadní je návrh datového modelu reprezentující žádost a řízení průběhu takového procesu. Způsobů, jakým tento datový model vytvořit je více.

5.1.2.1 Datový model

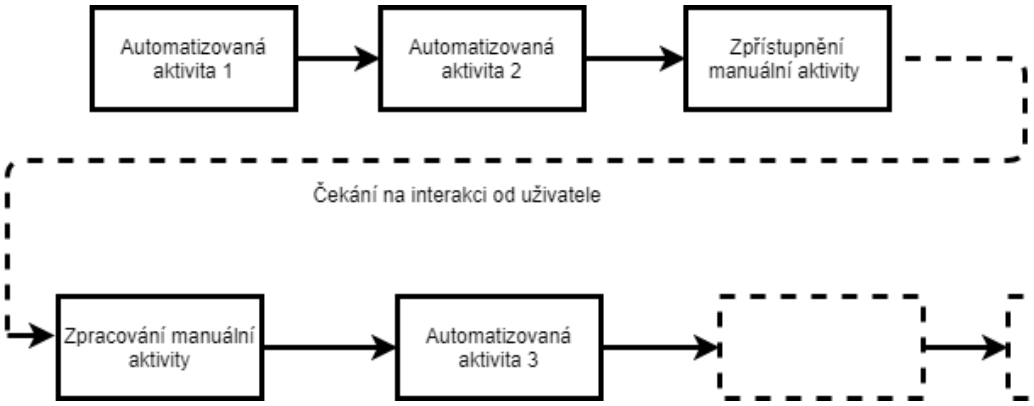


Obrázek 9 Entitně relační diagram – bez workflow engine

Entity *Request* představuje žádost a entita *Approval*, její schvalovací aktivity, o kterých je nutné také udržovat informace. Každá schvalovací aktivita náleží k nějaké žádosti a žádost může mít více těchto schvalovacích aktivit. S takovýmto datovým modelem, je také žádost připravena pro další rozšíření o schvalovací aktivity.

5.1.2.2 Řízení procesu

Stěžejní částí je návrh řízení procesu. Pro plně automatizované části postačí navazování jednotlivých metod, které reprezentují danou aktivitu, jednu za druhou, to zajistí provádění těchto kroků v přesně definovaném pořadí. Pro aktivity, kde je potřeba interakce od uživatele, je nutno ověřit, jestli je danou aktivitu možné provést, a to je zařízené pomocí atributu *currentManualStep* entity reprezentující proces. Po nastavení tohoto atributu bude zpřístupněná manuální aktivita, tedy aktivita, která je zpracovávaná nějakou osobou. Součástí jejího zpracování je navázání na automatizované kroky procesu, pokud existují další takové aktivity, které je potřeba vykonat. Abstrakce tohoto přístupu je znázorněna na obrázku níže (Obr. 2).



Obrázek 10 Řízení procesu bez workflow engine

5.1.3 Implementace

Dle návrhu je vytvořený datový model a řízení procesu. Průběh procesu je reprezentován navazováním metod, jak je definováno v návrhu. Důležitý je přechod mezi manuální aktivitou a automatizovaným zpracováním a naopak. Toto navázaní musí proběhnout po zpracování manuální aktivity. Taková implementace vypadá obecně následovně:

```
public void manualActivity(){

    // Zpracování manuální aktivity

    // Přechod na automatizované zpracování
    processAutomatedActivities();

}
```

Obrázek 11 Navázání z manuálního na automatické zpracování

Navázání z manuálního zpracování na automatizované lze vyřešit pomocí nastavování příznaku, který zpřístupní manuální aktivitu. Před zpracováním dané manuální aktivity je na základě takového atributu validována možnost jejího provedení.

```
public void automatedActivities(){

    automatedActivity1();

    automatedActivity2();

    // Přechod na manuální zpracování
    request.currentManualStep("step");

}
```

Obrázek 12 Navázání z automatizovaného zpracování na manuální

5.2 Srovnání přístupu s workflow engine a bez něj

V této kapitole jsou srovnávány oba přístupy k vytvoření aplikace a zhodnoceny jejich výhody a nevýhody. Srovnání se zaměřuje na části související se zpracováním žádosti. V následujícím textu je slovem aplikace myšlena pouze část zpracovávající žádost, respektive proces, pokud není řečeno jinak.

5.2.1 Kritéria srovnání

Zásadním kritériem pro srovnání je průběh softwarového cyklu, tedy jakým způsobem probíhá vývoj aplikace, od návrhu k implementaci, testování a údržbě, s tím souvisí i náročnost na provádění změn. Dalším kritériem je srozumitelnost zpracování žádosti pro business analytika, respektive člověka, který není zainteresovaný do vývoje aplikace. Výčet kritérií:

- Softwarový cyklus
    - Návrh
    - Implementace
        - Náročnost
        - Pracné části
        - Dokumentace

    - Testování
- Náročnost na provedení změn.
    - Verzování
- Výstavba komplexních procesů
- Srozumitelnost procesu

5.2.1.1 Softwarový cyklus

Workflow engine musí být nejdříve nakonfigurován. Při vývoji aplikace využívající workflow engine byla pro konfiguraci využita závislost od Activiti, která poskytla konvenční konfigurace. Konveční konfigurace však nemusí vždy vyhovovat. Po nakonfigurávání workflow engine můžu využívat jeho API a zpracovávat procesy. Bez využití workflow engine je jádrem zpracování procesu datový model reprezentující proces a CRUD operace nad databází, jejichž prostřednictvím je proces zpracováván.

Návrh

Použití workflow engine nás odstiňuje od spousty problémů, jejíž řešení musí být navrženo bez jeho použití. S využitím workflow engine je při návrhu zásadní integrace workflow engine s ostatními komponentami aplikace. Dále se návrh odvíjí od navržené procesní definice a zpracování její jednotlivých prvků.

Bez workflow engine je základem návrhu pro realizaci procesu vhodný datový model, jenž bude proces reprezentovat. S tím souvisí návrh komunikace s databází, respektive zpracování jednotlivých entit datového modelu. Aplikační logika musí být navržena tak, aby vhodně reflektovala požadovanou realizaci procesu.

Při návrhu je nevýhodou nevyužití workflow engine pohled na proces jako na celek. Návrh balíčku a tříd, které mají reprezentovat jeho zpracování, nepřináší vhodný nadhled a je obtížné si průběh procesu představit. Oproti tomu využití workflow engine Activiti tento pohled poskytuje. podmínkou je však znalost BPMN.

Implementace

Samotný kód je s použitím workflow engine Activiti rozdělen do více částí. Stěžejní částí je definice procesu, tedy v případě Activiti XML zapsané dle BPMN standartu, obsahem tohoto souboru je nejen popis procesu, ale i logika pro řízení procesu, či zde může být obsažena logika dílčích aktivit, jako je například *scriptTask,* dále z procesu může být vyvolána Java logika v podobě *ServiceTask*. Druhou částí je práce s Activiti API pro zpracování manuálních aktivit, respektive aktivit typu *userTask* a další práce s Activiti API pro kontrolu a správu procesu. Implementace je tedy vlastně rozdělena na dvě části, implementace procesu a implementace jeho zpracování. Bez workflow engine je vše vytvářeno v rámci jednoho logického celku.

Už v návrhu bylo naznačeno, že nás workflow engine odstiňuje od široké škály věcí, který sám implementuje. Díky tomu se lze při implementaci soustředit hlavně na samotný proces. Implementace probíhá na základně vizuálního návrhu procesu v BPMN. K vizuální podobě procesu v BPMN existuje jeho XML protějšek, který musí být doplněný o patřičnou logiku podobně jako v procesu žádosti v referenční aplikaci. Dále se implementace odvíjí od jednotlivých kroků procesu, respektive je implementováno zpracování jednotlivých prvků BPMN pomocí Activiti API.

Bez využití workflow engine musí být naimplementována komunikace s databází a s tím tedy souvisí i implementace CRUD operací pro doménové objekty. Dále je implementována nezbytná aplikační logika, která vhodně upravuje datový model procesu, či provádí další operace související s procesem. Změny datového modelu musejí být prováděny v určitém pořadí tak, aby zrcadlil průběh procesu a je nutné kontrolovat, zda může být daný krok procesu proveden, respektive jestli byly provedeny kroky, které mu měly předcházet. To je zásadní rozdíl oproti použití workflow engine. Workflow engine udržuje integritu procesu a kroky nelze přeskakovat.

S použitím workflow engine je pro vývojáře hlavním rozdílem v implementaci právě práce s API workflow engine, respektive to, že vývojář nezačíná vytvářet řešení od začátku a pracuje s dostupnou dokumentací k danému workflow engine. Dostupná dokumentace je na velmi vysoké úrovni a s tím přichází výhody při začleňování dalších osob do vývoje. Díky engine API se nechá soustředit pouze na implementaci zpracování jednotlivých kroků procesu. Nevýhodou je, že

workflow engine nemusí podporovat všechny potřebné prvky standartu BPMN a může tedy vývojáře omezovat.

Shrnutí vytvoření jednotlivých částí aplikace souvisejících přímo se zpracováním procesu.

S workflow engine

- Vytvořit XML definici procesu
- Integrace Activiti
  - Konfigurace engine
- Aplikační logika
  - Práce s Activiti API

Bez workflow engine

- Vytvořit datový model
  - doménové objekty
  - fyzický model dat
- Přístup k databázi
  - Komunikace s databází
  - DAO
- Objektově-relační mapování
  - konfigurace ORM frameworku
- Aplikační logika
  - Kontrola, jestli může být krok procesu proveden
  - Provedení

Práce s Activiti API je opravdu jednoduchá. Pro ukázku uvádím, co je potřeba udělat pro schválení úkolu nadřízeným. S Activiti pouze využiji instanci TaskService a metodu complete(), s parametry id procesu a kolekce proměnných, které procesu nastavit.

```
public void completeManagerApproval(long id, Map<String, Object> taskVariables) throws
ActivitiObjectNotFoundException {
    //taskService je instatnční promněná třídy TaskService
    taskService.complete(id, taskVariables);
}
```

V řešení bez workflow engine by byl kód složitější a musí být získána žádost, se kterou je potřeba pracovat, zkontrolovat, zdali může být následující krok proveden, upravit žádost a uložit, pokud nemůže být daný krok procesu proveden, tak vyvolat výjimku. Pro práci s databází musí být vytvořena DAO třída pro práci s daty entity reprezentující žádost.

```
public void completeManagerApproval(long id, StateEnum state, String reason) throws
IllegalArgumentException {
    // Získání žádosti
    Request request = requestDAO.getRequestById(id);
       // Kontrola jestli může být krok proveden
       if(request.getCurrentManualStep() == approvalByManager) {
           // úprava žádosti
          request.setMangerApproval(state);
          request.setManagerApprovalReason(reason);
          // update žádosti
          requestDAO.updateRequest(request);
          request.setCurrentManualStep();
       } else {
         // vyhození vyjímky pokud nemůžu provést tento krok
         throw new BadCurrentStep(request.currentStep);
       }
}
```

Z ukázky je patrné že pro vývojáře použití Activiti znamená méně práce. Analogicky se s Activiti API pracuje i v jiných případech. Výhodou Activiti je, že samo drží integritu procesu a není možné jednotlivé kroky přeskakovat, Activiti zajistí, že kroky procesu budou provedeny v pořadí, jaké bylo specifikováno.

Testování

Proces je součástí software a měl by se testovat stejně jako ostatní aplikační logika. Activiti poskytuje možnost testovat procesy pomocí JUnit. Možnosti testování jsou při nasazení, či nenasazení workflow engine podobné.

5.2.1.2 Náročnost na provedení změn

Proces se může během jeho životního cyklu dále vyvíjet a může být tedy nutné provést nezbytné úpravy. Při provádění úprav je nutné myslet na to, že stále mohou běžet procesy dle původního návrhu a nemusí být vždy žádoucí přejít okamžitě na novou verzi procesu. Optimálně je tedy potřeba zajistit, aby v jedné chvíli běžely instance procesů různých verzích a definice procesu zároveň a postupně přecházet k nové verzi. Samozřejmě záleží na rozsahu úprav. Mohou stačit málo nákladné úpravy, které nemusí ovlivnit již běžící procesy.

Workflow engine Activiti přidává nad rámec BPMN verzování definic procesu a umožňuje běžet více verzím procesu zároveň. Výchozím chováním Activiti workflow engine je spuštění instance procesu dle nejnovější verze definice procesu, ale je také možné spouštět i starší verze

procesu. Je nutné zajistit, aby aplikace dokázala zpracovávat jednotlivé prvky BPMN z obou verzí.

Bez využití workflow engine je nutné zachovat aplikační logiku a datový model původní verze procesu a zároveň vytvořit vše potřebné dle nového návrhu a zajistit, aby obě verze dokázaly běžet zároveň. Velice záleží na rozsahu úprav a na rozporu s původní aplikační logikou a datovým modelem. Může jít o přidání atributu do datového modelu a jeho zpracování v rámci již implementované aplikační logiky, nebo je nutné vytvořit úplně nový datový model a aplikační logiku. Kvůli zachování dvojí aplikační logiky a datového modelu může dojít k nabývání, či redundanci kódu. U některých změn lze snadno zachovat zpětnou kompatibilitu.

Přidání povinného atributu

Pokud je potřeba přidávat při vytvoření žádosti nový povinný atribut, který se již v průběhu žádosti nebude měnit a bude pouze zobrazován, tak při využití workflow engine se přidá jako jedna z proměnných při vytváření procesu.

Podobně jednoduché to je i bez využití workflow engine. Rozšíří se datový model žádosti a obstará zpracování nového atributu.

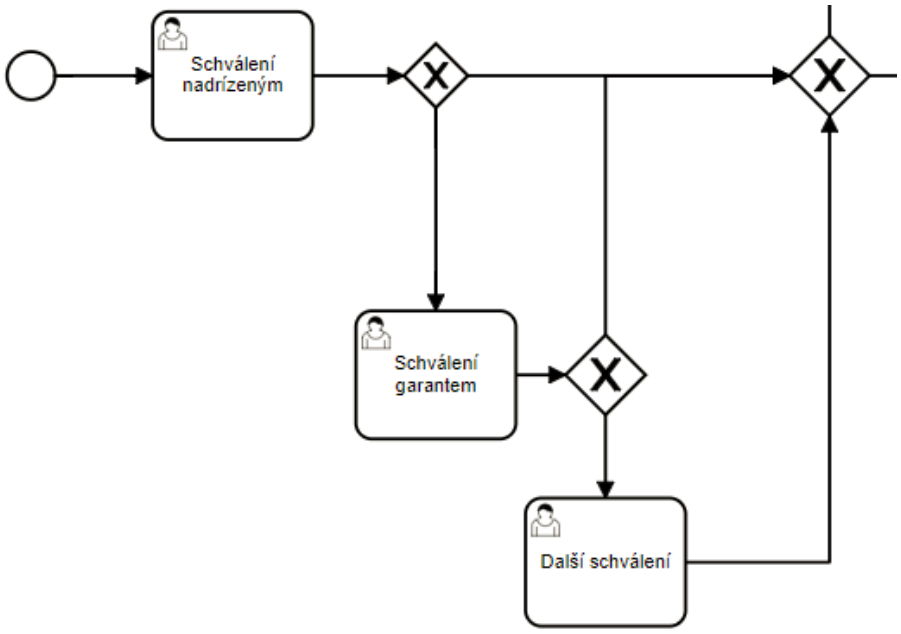
Přidání nového atributu neovlivní, již vytvořené žádosti. Nově vytvořené žádosti budou tento atribut obsahovat.

Problém nastává, když je potřeba tento atribut přidat i k již dokončeným procesům. Do uvedeného procesu žádosti o roli by mohlo být požadováno přidání časového rozmezí, od kdy do kdy má být role přidělena. Ve staré verzi procesu bude požadováno, aby hodnoty těchto atributů byly doplněny na implicitní hodnotu nebo dopočteny. Ve verzi bez workflow engine lze tuto změnu snadno provést. Do datového modelu se přidají potřebné atributy a nastaví se na požadovanou hodnotu. S workflow engine nastává problém. Proměnné, které reprezentují proces, nelze prostřednictvím workflow engine Activiti přidávat po doběhnutí procesu, tudíž data o procesu nemůžou být v takovém případě součástí jednoho celku. Pokud bude potřeba tyto atributy doplnit bude nutné jednotlivé instance staré verze procesu propojit s externí datovým zdrojem, kde budou tyto informace uloženy a tyto data pro zobrazování agregovat.

Samotné přidání atributu do procesu má v obou případech podobnou náročnost. Při požadavku, kdy je potřeba doplnit hodnoty i pro staré verze procesu, jsou možnosti workflow engine activiti omezené a tento požadavek nelze provést pouze pomocí Activiti API a je nutné doprogramovat vlastní přístup. Celkově není datový model workflow engine určen pro trvalé uložený dat a k provádění agregovaných business inteligence operací nad těmito daty. V omezené míře se dá v tom směru použít, ale přirozeně reprezentuje pouze průběh procesu. Pro další práci s daty, které jsou získány během průběhu procesu, je vhodné vytvořit samostatný datový model.

Přidání aktivity

Obtížnější je, když se proces znatelně změnil, byly změněny atributy a průběh procesu, respektive jeho kroky. Příkladem takové změny je rozšíření žádosti o další schvalovací aktivitu.



Obrázek 13 Přidání schvalovací aktivity

S Activiti je změna jednoduší. Průběh procesu je samostatný celek, tudíž se nemusíme starat o změny týkající se řízení procesu, tedy větvení dle verze, díky Activiti poběží dvě verze procesu zároveň, je nutné zajistit jen zpracování aktivit, které vyžadují lidskou režii, tedy aktivity typu *userTask*.

Bez workflow engine je nutné přidat do modelu žádosti atribut, který bude označovat verzi procesu, tak aby se dalo při řízení průběhu rozpoznat o jakou verzi procesu se jedná. V tomto případě by musel být upraven:

- datový model
    - přidat atributy týkající se schvalovací aktivity
- logika průběhu žádosti
    - zde je nutné zařídit zpětnou kompatibilitu pomocí větvení dle verze žádosti. Na základě verze se průběh procesu přesune k danému kroku
- uživatelské rozhraní
    - zpracování nové schvalovací aktivity
    - zachovat zpracování obou verzí procesu

To je příklad pouze s jednou dílčí změnou, při větších změnách procesu bude nutné více takovýchto úprav. Tyto úpravy samozřejmě zesložiťují kód.

V obou případech se musí myslet i na zpětnou údržbu. To znamená následnou úpravu kódu za účelem odstranění nepotřebné logiky týkající se již neběžící verze procesu. Ani v jednom případě se tedy neubráníme následné režii a dalším změnám kódu.

Activiti engine a obecně workflow engine jsou navrženy tak, aby změna byla co nejrychlejší a co nejméně časově nákladná. Výhodou workflow engine je tedy snadná rozšiřitelnost procesu a práce s různými verzemi procesu. Bez využití workflow engine jsou obě tyto činnosti náročnější. Zpracování procesu bez využití workflow engine je vhodné tam, kde se neočekávají změny v návrhu procesu.

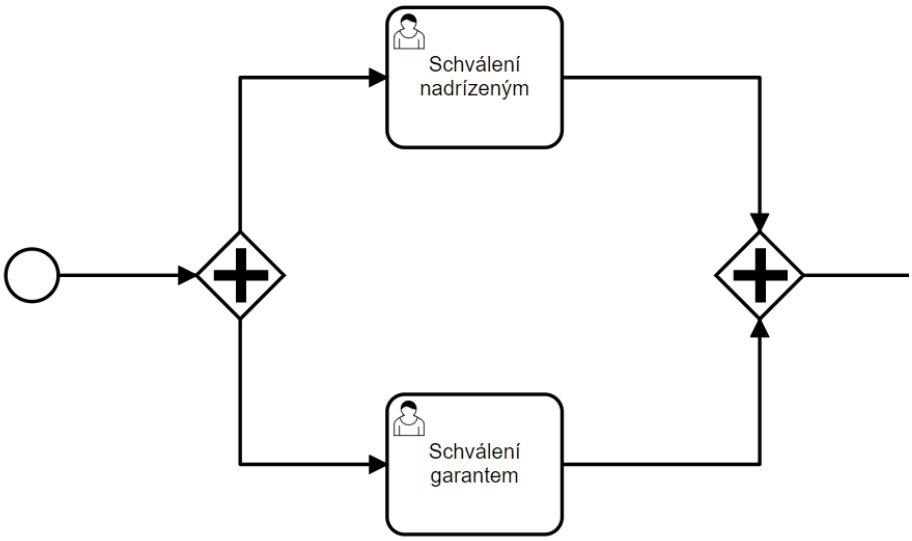
5.2.1.3 Výstavba komplexních procesů

Výstavba komplexních procesů, tím je myšleno, jaký má proces schopnosti v komunikaci s okolím, eskalací, či paralelní průběhem.

Paralelní průběh procesu

Další změna, která by mohla být požadována u procesu žádosti je provádět schvalování žádosti paralelně. Součástí BPMN je možnost návrhu procesu s paralelním průběhem aktivit.

Návrh části procesu s paralelním zpracováním je na obrázku (Obr. 6). Toho lze docílit použitím tzv. paralelních bran. Ty slouží k rozvětvení procesu do více cest a zároveň zařídí i jejich opětovné sloučení.



Obrázek 14 Paralelní schvalování

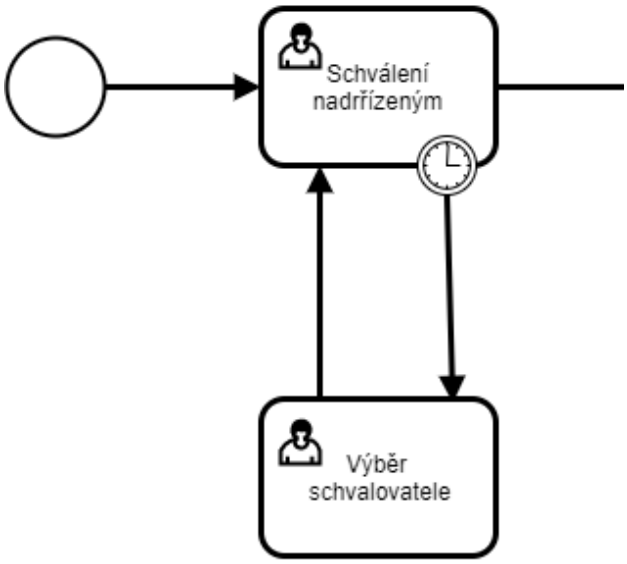
S využitím workflow engine Activiti je možné tuto změnu snadno provést. Workflow engine provádí obě větve procesu a zároveň zařídí opětovné spojení průběhu procesu po doběhnutí obou větví procesu a na následné naváže na další průběh. V tomhle případě stačí upravit XML soubor s definicí procesu dle návrhu.

Bez workflow engine je nutné vytvořit kontrolní mechanismus, který další průběh spustí až po provedení obou aktivit. Náročnost vytvoření takového řešení je malá.

Eskalace

Daná aktivita nemusí být vyřešena vždy standartním způsobem, a proto je potřeba během procesů řešit eskalace takovýchto problémů. Součástí BPMN jsou tzv. *Boundary events.* Tedy události, které jsou navázány na danou aktivitu a čekají na určitou spoušť, či signál. Příkladem takového problému v rámci zpracování žádosti je dlouhé čekání na vyřešení žádosti. To lze vyřešit pomocí

Timer Boundary Event. Tento typ události se chová jako stopky a když se průběh procesu dostane k aktivitě, které má na sobě navázaný tento typ události, tak se spustí čas a pokud aktivita nebude zpracována během tohoto časového limitu přejde průběh procesu k aktivitě, která se má spustit po vyvolání této události. Návrh procesu s tímto typem události je na (Obr. 7).



Obrázek 15 Eskalace

Workflow engine Activiti podporuje takovéto zpracování procesu a dále jsou dostupné i další varianty událostí tohoto typu.

Bez workflow engine by samotný vývoj stejného chování, který by zajišťoval spolehlivou funkcionalitu, byl náročný.

Komunikace s okolím

Workflow engine Activiti poskytuje na úrovní aktivit:

- Email Task – Posílání emailů
- Web Service Task – Komunikace s webovou službou pomocí WSDL
- Business Rule Task – Provedení business pravidel a napojení na Drools Expert.
- Mule task – Komunikace s Mule ESB
- Camel Task – Konektor na Apache Camel

Vytvořit a přidat další konektory lze prostřednictvím *Java service Task*. Workflow engine Activiti neposkytuje širokou škálu konektorů na další služby a software. Použití workflow engine při specifických požadavcích nepřináší v tomto ohledu velké výhody oproti systému bez workflow engine.

5.2.1.4 Srozumitelnost procesu

Navržený proces, který je znázornění pomocí BPMN, lze jednoduše nasadit a jeho průběh řídit pomocí workflow engine. Workflow engine zajišťuje, že aktivity business procesu jsou prováděny ve specifikovaném pořadí, díky tomu je přesně reflektován záměr business analytika, či osoby, která proces navrhovala.

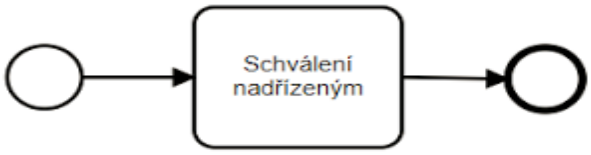
Pokud máme navržený proces a chceme automatizovat jeho zpracování bez využití workflow engine, je potřeba implementovat logiku jenž bude průběh procesu zpracovávat, tím se ale vytrácí můstek mezi business analytikem a vývojáři. V řešení, které by mělo splňovat stejné požadavky bez použití workflow engine se operace pro zpracování procesu smrskly na úpravu datového modelu a provádění těchto úprav v přesně definovaném pořadí, což nemusí vždy přesně zrcadlit zamýšlený průběh procesu. Je zde obtížnější pochopit proces jako celek, pro jeho pochopení je potřeba analyzovat kód, který oproti BPMN má nižší míru abstrakce procesu. Ze stejného důvodu jde proces i obtížněji optimalizovat, obtížněji se zjišťují informace o procesu.

Nevýhodou je, že Activiti a také další workflow enginy neimplementují celý standart BPMN, takže může být navrhnut proces, který sice bude popsán pomocí BPMN, ale nebude ho možné nasadit ke zpracování workflow engine, jelikož obsahuje prvky BPMN standartu, které nejsou podporovány. Pak je nutné návrh procesu pozměnit a upravit dle možností workflow engine, což nemusí být vždy žádoucí. Pokud proces implementujeme bez použití workflow engine tak nejsme omezení standartem, ale jsou zde výše zmíněné nevýhody.

5.2.1.5 Obecný přístup oproti konkrétnímu

Workflow engine nabízí obecný přístup, je tedy možné jeho pomocí implementovat většinu procesů navržených s použitím BPMN standartu. Nasazení workflow engine pro jednoduchý proces může být zbytečně těžkopádné. Například pokud by bylo požadováno v rámci malé

organizační struktury řešit žádost o dovolenou, jejíž vyřízení by měla na starosti jedna osoba mohl by daný proces vypadat takto:



Obrázek 16 Jednoduché schvalování

V rámci vytváření žádosti by se dodali potřebné proměnné. Zpracovat takovýto proces prostřednictvím Activiti je snadné. Pro zpracování procesu na úrovni Activiti jsou zapotřebí pouze dvě metody z Activti API. Při zpracování bez workflow engine bude nutné vytvořit datový model pro takovou žádost a operace pro úpravu tohoto modelu. Pokud by v projektu, kam chceme tento proces nasadit byl již integrován workflow engine, bylo by vhodné nasadit proces jeho prostřednictvím. V případě kde tomu tak není je vhodnější možností zpracovávat podobný proces bez workflow engine, všechny implementační úkony, které je v tomto případě nutné provést jsou součástí standartního softwarového inženýrství, které se týkají zpracování datového modelu.

Activiti workflow engine nabízí plno vlastností, které nemusí být využity, oproti konkrétní implementaci daného procesu, kde jsou samozřejmě implementovány pouze potřebné části. V tom případě je řešení na míru ve většině případů, záleží na složitosti procesu, méně náročné. Avšak při nasazení více procesů, které se budou lišit může být vytváření unikátního řešení pro každý proces náročné. Větší výhody přináší workflow engine se složitějšími procesy a s množstvím různých procesů. Vytvoření konkrétního řešení pro jeden proces je vhodné pouze při nasazení jednoduchých, či malého množství procesů.

5.3 Interpretace výsledků

V tabulkách níže je uvedeno shrnutí vycházející z výše uvedeného textu.

V následující tabulce (Tab. 8) je uvedeno shrnutí implementace

	S workflow engine	**Bez workflow engine**
Srozumitelnost kódu	Srozumitelný, proces je implementován v definici procesu. O řízení se stará engine. Zpracování manuálních aktivit a získávání informací o procesu probíhá prostřednictvím Activiti API. Jasné rozdělení zodpovědností.	Méně srozumitelný. Průběh procesu je po sobě jdoucí skupina metod, kde se musí propojit automatické zpracování s manuálními aktivitami, kde je potřeba čekat na interakci od uživatele. Obtížné rozdělení zodpovědností do logických celků.
Náročnost implementace	Jednoduší, jsme odstínění od implementačních detailů a lze se soustředit pouze na proces.	Složitější, v řešení musí být implementována logika pro proces samotný, jeho řízení a kontrolu.
Pracné části implementace	Konfigurace a integrace workflow engine.	Zajištění správného řízení procesu.

Tabulka 8 Implementace

V tabulce (Tab. 9) kritéria jsou zhodnocena kritéria související s definicí procesu.

	S workflow engine	**Bez workflow engine**
Srozumitelnost implementace procesu pro business analytika	Proces je ve formě, který by měl business analytik znát.	Programový kód nepřináší standartní náhled na proces jako celek.
Vhodnost systému dle počtu různých procesů	Při malém množství jednoduchých procesů může být nasazení workflow engine těžkopádné.	Při implementaci jedno, či malého množství procesů, které se nebudou měnit.
Omezení v návrhu procesu	Omezení podporovanými elementy modelovacího jazyka a možnostmi workflow engine.	Neomezený. Omezení pouze možnostmi programovacího jazyka.

Tabulka 9 Definice procesu

V tabulce (Tab. 10) jsou uvedeno srovnání související s prováděním změn v procesu.

	S workflow engine	**Bez workflow engine**
Náročnost na provedení změn	Pro většinu případů méně náročné.	Obecně náročnější, ale záleží na požadované změně.
Běh více verzí stejného procesu	Méně náročný, snadno lze zachovat dvě verze procesu.	Náročný, musí se zachovat dvojí aplikační logika a kontrolovat verzi procesu pro jeho další řízení.

Tabulka 10 Změny v procesu

V tabulce (Tab. 11) jsou uvedeny možnosti pro výstavbu komplexních procesů. Řešení bez workflow engine u těchto vlastností nepřináší žádnou podporu a vše musí být vytvořeno.

	S workflow engine	**Bez workflow engine**
Komunikace s okolím – rozhraní do dalších systémů	Implementována pouze malá množina, nepřináší znatelné výhody.	–
Paralelní zpracování procesů	Podpora na úrovni definice procesu. Možnost namodelovat pomocí BPMN.	–
Eskalace	Podpora na úrovni definice procesu. Možnost namodelovat pomocí BPMN.	–

Tabulka 11 Možnosti pro komplexní procesy

Na základě shrnutí nelze zvolit jeden konkrétní přístup, který by vyhovoval všem případů. Cílový uživatel se na základě otázek v tabulce (Tab. 12) může rozhodnout jaký přístup zvolit.

	Ano	**Ne**
Je proces, který chcete nasadit jednoduchý, tedy obsahuje malé množství kroků?	Bez workflow engine	Workflow engine
Očekáváte časté změny procesu?	Workflow engine	Bez workflow engine
Chcete, aby v systému bylo více druhů procesu?	Workflow engine	Bez workflow engine
Očekáváte další práci s daty, které jsou získány během procesu?	Bez workflow engine	Workflow engine
Chcete začlenit více lidí do vývoje a mít pro to vytvořenou potřebnou dokumentaci?	Workflow engine	Bez workflow engine
Proces navrhuje netechnický uživatel?	Workflow engine	Bez workflow engine
Oddělení logiky procesu od aplikační logiky?	Workflow engine	Bez workflow engine

Tabulka 12 Otázky pro výběr přístupu

6 Závěr

V teoretické části je představen software typu Workflow engine a s ním souvisejícími pojmy, přičemž zvláštní důraz byl kladen na přiblížení pojmů z tohoto business prostředí vývojáři. Stěžejní přínosem workflow engine při zakomponování do aplikace je vytvoření můstku mezi business zainteresovanými lidmi a vývojáři.

V navazující rešeršní části jsou srovnány workflow engine Activiti a jBPM. Jsou stanovena kritéria, podle kterých je vhodné vybírat workflow engine nad kterým lze stavět vlastní řešení. Kritéria srovnání jsou vybrána tak, aby zahrnovala zájmy business analytika i vývojáře. Workflow engines jsou podle těchto kritérií srovnány. Účelem srovnání nebylo vybrat lepší workflow engine, ale spíš zvolit kritéria, kterými se při volbě zabývat. Vhodným rozšířením této části by bylo porovnání možnosti škálovatelnosti workflow engine, tedy jeho schopnosti reagovat na vzrůstající nebo klesající nároky na výkon systému. Dále porovnání open-source řešení s komerčními.

V praktické části je vytvořena referenční aplikace využívající workflow engine Activiti. Ta demonstruje použití workflow engine. Prostřednictvím jejího vytvoření bylo možné zhodnotit možnosti a vlastnosti workflow engine, které slouží pro další srovnání.

Využití workflow engine je dále srovnáno s přístupem, kde by byl proces řízen bez jeho použití. Výsledkem tohoto srovnání je zhodnocení kritérií ve spojení s daným přístupem.

Nelze zvolit jeden přístup, který by se dal obecně použít na všechny případy. Volba přístupu pro řízení procesů tedy není jednoznačná. V návaznosti na tento fakt je stanoven seznam otázek na základě, kterých si cílová osoba, může zvolit přístup, které pro své řešení zvolí.

Nejvhodnějším rozšířením práce by bylo reálné nasazení workflow engine pro řízení procesů, respektive nahrazení přístupu, kde byly procesy řízeny bez workflow engine. Účelem by bylo zhodnotit náročnost a přínos této migrace. Dlouhodobé aktivní používání takového systému by také přineslo objektivnější pohled na jeho použitelnost.

Literatura

[1] ŘEPA, Václav. *Podnikové procesy: procesní řízení a modelování*. 2., aktualiz. a rozš. vyd. Praha: Grada, 2007. Management v informační společnosti. ISBN 978-802-4722-528.

[2] SVOZILOVÁ, Alena. *Zlepšování podnikových procesů*. Praha: Grada Publishing a.s., 2011. ISBN 8024772965.

[3] HOLLINGSWORT, D. *Terminology & Glossary: WFMC-TC-1011*. Workflow Management Coalition, 1999.

[4] WESKE, Mathias. *Business process management: concepts, languages, architectures*. 2nd ed. New York: Springer, 2012. ISBN 978-364-2286-155.

[5] VONDRÁK, I. *Informační technologie pro praxi: sborník přednášek semináře ..* [online]. Ostrava: Repronis ve spolupráci s Tanger, 1997-2012 [cit. 2017-11-27]. ISBN 80-859-8863-1.

[6] RADEMAKERS, Tijs. *Activiti in action: executable business processes in BPMN 2.0*. Shelter Island, NY: Manning Publications, 2012. ISBN 978-161-7290-121.

[7] About-BPMN. *OMG* [online]. Object Management Group®, 2010 [cit. 2017-5-12]. Dostupné z: http://www.omg.org/spec/BPMN/2.0/About-BPMN/

[8] Activiti. *Activiti* [online]. Alfresco Software, Inc, 2017 [cit. 2017-11-27]. Dostupné z: https://www.activiti.org

[9] *JBPM* [online]. Red Hat, Inc, 2017 [cit. 2017-11-27]. Dostupné z: http://www.jbpm.org/

[10] Drools. *Drools* [online]. Red Hat, Inc., 2016 [cit. 2017-11-27]. Dostupné z: http://drools.jboss.org/

[11] *Angular Material* [online]. Google, 2017 [cit. 2017-11-27]. Dostupné z: https://material.angularjs.org

[12] *Maven* [online]. The Apache Software Foundation, 2017 [cit. 2017-11-27]. Dostupné z: http://maven.apache.org

[13] *Spring* [online]. Pivotal Software, Inc, 2017 [cit. 2017-11-27]. Dostupné z: https://spring.io

[14] *Hibernate* [online]. Red hat, Inc, 2017 [cit. 2017-11-27]. Dostupné z: http://hibernate.org/

Seznam zkratek

BPM – Business process management

WSDL – Web Services Description Language

ESB – Enterprise service bus

API – Application Programming Interface

REST – Representational State Transfer

JPA – Java Persistence API

DAO – Data access object

POM – Project Object Model

XML – Extensible Markup Language

JSON – JavaScript Object Notation

Seznam obrázků

Seznam tabulek

Seznam příloh

Přiložené CD obsahuje elektronickou verzi práce a kód referenční aplikace.