TALLINNA TEHNIKA ÜLIKOOL
Infotehnoloogia teaduskond
Informaatikainstituut
Infosüsteemide õppetool

IDU70LT

Protsessimootorite valiku metoodika

Magistritöö

Üliõpilane: Edvard-Sander Põldmäe
Üliõpilaskood: 144331
Juhendaja: dotsent Enn Õunapuu

Tallinn
2016

Autorideklaratsioon

Kinnitan, et olen koostanud antud lõputöö iseseisvalt ning seda ei ole kellegi teise poolt varem kaitsmisele esitatud. Kõik töö koostamisel kasutatud teiste autorite tööd, olulised seisukohad, kirjandusallikatest ja mujalt pärinevad andmed on töös viidatud.

(*kuupäev*)	(*allkiri*)

Annotatsioon

Lõputöös „Protsessimootorite valiku metoodika“ uuritakse erinevaid protsessimootoreid ja võrreldakse neid tarkvara kvaliteedinäitajatega. Võrdluse käigus tuuakse välja ühe või teise mootori eripärad, lihtsustamaks protsessimootorite valikut, mis on ühtlasi lõputöö peaeesmärk. Samuti viib lõputöö autor läbi iga protsessi peal ka praktilise osa ning kirjeldab selle teostamist ühe mootori peal, andmaks sissejuhatuse ühe mootori kasutamise kohta.

Lõputöö piiranguks on nõuete püstitamisel ärinõuete puudumine. Autor võrdleb mootoreid kõige üldisemas võtmes ehk ei võrrelda spetsiifilistest nõuetest lähtuvalt. See asjaolu raskendab mootorite võrdlust, kuna taolisel moel ei ole võimalik kõiki kvaliteedinõudeid arvesse võtta.

Töö põhitulemus on dokument, mille abil on võimalik näha ja võrrelda valitud mootorite erinevaid omadusi eelkõige selle jaoks, lihtsustamaks projektijuhtide eelanalüüsimise vaeva projektides, kus on plaanitud kasutada ühte valitud mootoritest. Dokument on sobilik sissejuhatus ka protsessimootoritega esmase tutvumise tegemiseks.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 37 leheküljel, 5 peatükki, 10 joonist, 6 tabelit.

Abstract

The thesis „Process Engine selection methodology“ aims to analyze different Process Engines and compare them against each other using the Software product Quality Requirements and Evaluation – Quality model. The main goal of this thesis is to bring out the differences of the engines while comparing them which in return will help anyone to find a Process Engine that meets their requirements more easiliy. The author of this thesis will also test out each Process Engine that will be compared in this thesis.

There are drawbacks, however, because of comparing the engines in the absence of business requirements. The author will compare the engines in the most general way which in return will make some of the requirements of the Quality model invalid.

The main outcome of the thesis is a document which can be used to see and compare the main differences in the functionalities of the chosen Process Engines. This document will be most helpful for project leads who plan to use Process Engines in their future projects. This document is also useful for an introduction to Process Engines.

The thesis is in Estonian and contains 37 pages of text, 5 chapters, 10 figures, 6 tables.

Lühendite ja mõistete sõnastik

BPM	Business Process Management, äriprotsesside juhtimine
BPMN	Business Process Modelling Notation, äriprotsesside modelleerimise standard
User task	Ärimodelleerimise element: kasutaja ülesanne
Service task	Ärimodelleerimise element: teenuse ülesanne
Protsessi eksemplar	Protsessi eksemplar (*process instance*) on ühte kindlat protsessi läbiv või läbinud tegevuste jada
Protsessi definitsioon	Reeglite ja tingimustega kirjeldatud protsess
Plugin [5]	Pistikprogramm, *plugin* – tarkvaramoodul, mis lisab suuremale süsteemile teatud omaduse või teenuse
Maven	Project and build manager by Apache - Apache poolt loodud projekti ja järgu manageerija
Juurutamine [5]	*Deploy* –Uue tarkvara tööle panemine, installeerimine, konfigureerimine uues keskkonas
Start event	Ärimodelleerimise element: algatav sündmus
End event	Ärimodelleerimise element: lõpetav sündmus
Gateway	Ärimodelleerimise element: värav

Jooniste nimekiri

Tabelite nimekiri

Sisukord

1. Sissejuhatus

Eestkätt tehnoloogia arengu kiirus on see, mis sunnib tänapäeval tarkvaraarendajaid kasutama valmis loodud tarkvara komponente infosüsteemide arendamisel. Protsessimootori loomine on üks neist arendusprotsessi osadest, mis on väga aeganõudev ja kulukas ettevõtmine. Protsessimootor on tarkvara raamistik, mis on äriprotsesside südameks nende käivitamisel ja töökorras hoidmisel. Protsessimootori valimisel kaob küll spetsiifilise lahenduse paindlikkus, kuid säästetakse palju aega ja raha - kusjuures paljudel juhtudel, kus arvatakse, et läheb tarvis spetsiifilist lahendust, saaks endiselt hakkama ka valmis protsessimootoriga.

Käesoleva lõputöö „Protsessimootorite valiku metoodika“ teema valikul mängis olulist rolli see, et autor ise on juba sellise olukorraga kokku puutunud, kus on otsustatud kasutusele võtta üks olemasolevatest protsessimootoritest, kuid kuna nõudluse ajal ei olnud mootorid niivõrd hästi dokumenteeritud, oli võrdlemine raskendatud. Lisaks on tekkinud isiklik huvi antud teema suhtes, kuna autor on töötanud ligi kolm aastat infotehnoloogilisi lahendusi pakkuvas erafirmas ning osalenud protsessimootoreid kaustatavates projektides.

1.1 Taust ja probleem

Selle lõputöö eesmärk on uurida erinevaid protsessimootoreid ja võrrelda neid autori poolt püstitatud kvaliteedinäitajatega.

Teema valiku tingis fakt, et on suurenenud vajadus kasutada valmis protsessimootoreid infosüsteemide lahendustes, säästmaks aega ise mootorite arendamise arvelt. Keeruliseks võib aga osutuda kindla mootori valik, kuna pakutavaid mootoreid on tänapäeval üle kahekümne. Mootori sobitamisel süsteemi on kõige tähtsam viia vastavusse mootori omadused kliendi nõuetega. See aga võib jätta sõelale veel mitu protsessimootorit või hoopis mitte ühtegi, eestkätt vähese kogemuse või vähese informatsiooni tõttu.

See töö on vajalik kõikidele osapooltele, kes on huvitatud mõne protsessimootori kasutamisest oma lahendustes. Protsessimootorite tundmine on suureks eeliseks projektide

algstaadiumis. Tänu lõputöös antavale ülevaatele protsessimootorite kohta on võimalik palju täpsemalt hinnata projektide mahtu ja ressursikulusid.

1.2 Ülesande püstitus

Lõputööle on püstitatud järgmised ülesanded:

- Välja tuua täpsed nõuded, millele toetudes võrreldakse omavahel valitud protsessimootoreid.
- Läbi proovida paigaldamine, lihtsa protsessi loomine ja juurutamine kolmel mootoril: Camunda, Activiti ja Bonita.
- Võrreldakse mootorite omaduste vastavust püstitatud nõuetele.
- Tuua näidis ühe mootori paigaldamise ja protsessi loomise ning juurutamise kohta.

1.3 Metoodika

Lõputöö strateegiaks on *Action research* ehk tegevusuuring. Kõigepealt püstitab autor kvaliteedinõuded mootoritele. Seejärel algab tegevusuuringu osa, mis koosneb järgnevatest tegevustest:

- Protsessimootori seadistamine
- Protsessimootori konfigureerimine
- Näidisprotsessi loomine
- Näidisprotsessi juurutamine ja käivitamine

Pärast tegevusuuringut analüüsib autor valitud mootoreid ning võrdleb mootoreid omavahel nõuetest lähtuvalt, võttes informatsiooni kinnitamiseks kasutusele ka antud protsessimootorite dokumentatsioonid.

1.4 Ülevaade tööst

Töö sisu jaguneb kolmeks suureks peatükiks. Esimeses peatükis põhjendatakse mootorite valimit ning kirjeldatakse lühidalt mootorite tausta. Täpsemalt hakatakse võrdlema omavahel Camunda, Bonita ja Activiti protsessimootoreid. Samas peatükis püstitatakse ka kvaliteedinõuded, mida hakatakse hiljem võrdlema valitud protsessimootoritega. Teises peatükis võetakse ette üks haaval püstitatud nõudeid ning kirjeldatakse mootorite omadusi igast nõudest lähtuvalt. Võrdlemise käigus kõrvutatakse mootoreid, et oleks kergem välja tuua nende erinevusi. Viimasena kirjeldatakse tehtud praktilist osa. Dokumendi valmimisega paralleelselt katsetatakse iga mootoriga läbi mootori seadistamine ja konfigureerimine, mootoril äriprotsessi loomine ning juurutamine. Viimases peatükis kirjeldatakse praktilise osa teostamist ühel mootoril.

2. Nõuete püstitamine

Järgnevalt põhjendab autor, miks valiti just need kolm mootorit ning püstitatakse üldised nõuded mootoritele.

2.1 Mootorite valim

Antud töös valis autor võrdlemiseks kolm protsessimootorit – Activiti, Bonita ja Camunda.

Need kolm valiti, sest kõik on vabavaralised ja avatud lähtekoodiga, neil on suur kogukond kasutajaid ning need on populaarsed valikud, mistõttu käivad nende nimed läbi tõenäoliselt igast valimist, kui keegi peaks otsima valmis protsessimootorit oma süsteemi.

Kõigil kolmel on nii *community* versioon, mis on tasuta ja ka *enterprise* versioon, mis on tasu eest, kuid firmad pakuvad omapoolset nõustamist. Mõnel juhul on ka mõned lisafunktsioonid.

Activiti mootorit hakati looma juba 2010.aasta suvel, eesmärgiga lihtsustada pidevat käsitööd ja üritada automatiseerida võimalikult palju äriprotsessidega seonduvaid tegevusi. 2012.aasta alguses *branch*'iti Activiti koodi ning sellest saigi alguse Camunda. Seetõttu on Camunda ja Activiti sarnase ehitusega. Bonita on neist eraldiseisev ning kõige uuem. Bonita kallal hakati tegelema 2013.aasta suvest.

Järgnev tabel annab üldise ülevaate iga valitud mootori kohta.

Tabel 1. Protsessimootorite ülevaade

	Bonita [4]	Activiti [3]	Camunda [2]
Hetkel viimane versioon	7.1.5	5.19.0.2	7.4.0
Viimase versiooni väljalaske kuupäev	08.01.2016	29.01.2016	01.11.2015
Raamistiku programmeerimise keel	Java	Java	Java
Litsents	LGPL v2/GPL v2	Apache Software License 2.0	Apache Software License 2.0

2.2 Nõuete püstitamise piirangud

Tarkvara sobitamine kliendi süsteemiga või kasutuselevõtt uues süsteemis sõltub suures osas kliendi vajadustest tulenevatest ärinõuetest. Igal kliendil on oma nägemus ja omad nõudmised, kui on soov kasutusele võtta protsessimootor. Selle töö raames peame välistama kliendi nõuded, kuna üldise ülevaate saamiseks ei saa tuua sisse spetsfiilisi kliendi nõudeid.Seetõttu tuleb jätta ärinõuded kõrvale ja keskenduda eelkõige mittefunktsionaalsetele nõuetele, mis pigem näitavad toote kvaliteeti.

Protsessimootori nõuete püstitamiseks võttis autor aluseks kvaliteedinäitajad standardi ISO/IEC 25000 ja ISO/IEC 9126 seeriate põhjal [1, 102]. ISO/IEC 25000 seeria olulisemaid osi on kvaliteedimudelite standard ISO/IEC 25010 *Software engineering: Software product Quality Requirements and Evaluation*, mille eestikeelne tõlge on standard EVS-ISO/IEC 25010:2011.

Tabel 2. Tarkvaratoote kvaliteet (EVS-ISO/IEC 25010:2011 põhjal)

Funktsionaalne sobivus	Soorituse tõhusus	Töökindlus	Ühilduvus	Kasutatavus	Turvalisus	Hooldatavus	Porditavus
Funktsionaalne täielikkus	Ajaline käitumine	Küpsus	Koosolu-võime	Kohasuse äratuntavus	Konfiden-tsiaalsus	Modulaarsus	Sobitatavus
Funktsionaalne õigsus	Ressursi kasutus	Tõrketaluvus	Koostalitlus võime	Õpitavus	Terviklus	Taaskasutatavus	Installeeritavus
Funktsionaalne kohasus	Suutvus	Taastuvus		Käsitsetavus	Salgamatus	Analüüsitavus	Asendatavus
		Töökindluse vastavus		Eksituskindlus	Jälitatavus	Modifitseeri-tavus	
				Kasutaja-liidese esteetika	Autentsus	Testitavus	
				Hõlpsus			

- Funktsionaalne sobivus. Seda, kas mootor täidab üht või teist funktsionaalsust, käesolevas töös ei käsitleta, kuna funktsionaalsuse dikteerivad üldjuhul kliendi poolt tulenevad nõuded, kes antud töö raames puudub.

- Sooritуse tõhusus. Kuna käesolevas töös realiseeritakse erinevatel mootoritel lihtne näidisprotsess, ei saa sooritуse tõhususe kohta adekvaatset hinnangut anda ning võimalusel võrdleb autor seda kaudselt mootorite dokumentatsiooni põhjal.
- Töökindlus on suures osas samuti kindlaks määratud kliendi nõuetega, kuid võimalusel hindab autor töökindlust kaudselt mootorite dokumentatsiooni põhjal.
- Ühilduvust hindab autor suures osas näidisprotsessi käigus. Näidisprotsesside testimisel jääb ühilduvuse hindamisest kõrvale ainult ühilduvus väliste süsteemidega, mille hindamiseks võetakse kasutusele mootorite dokumentatsioonid.
- Kasutatavus tuleneb samuti suures osas kliendi nõuetest, eriti lõppkasutajate poolt vaadatuna. Omavahel saab aga kasutatavust hinnata arendaja vaatenurgast protsessi loomisel ja protsessi juurutamisel.
- Turvalisus sõltub otseselt süsteemi arhitektuurist, mis on tihedalt seotud kliendi nõuetega.
- Hooldatavust hindab autor nii näidisprotsessi läbi tehes kui ka protsessimootorite dokumentatsioone võrreldes antud teema raames.
- Porditavuse nõude kohta saab autor piisava ülevaate mootori seadistamisel.

2.3 Nõuete nimekiri

Kvaliteedimääradest tulenevalt uurib autor mootorite juures järgnevaid nõudeid:

- Funktsionaalne sobivus
- Sooritуse tõhusus
- Töökindlus
- Ühilduvus
- Kasutatavus
- Turvalisus
- Hooldatavus
- Porditavus

Eelnevalt sai määratletud, milliseid mootorite kvaliteedinõudeid hakatakse võrdlema. Kitsendav asjaolu on see, et puudub kindel projekt või klient, seega tuleb nõudeid vaadata väga üldiselt. Tavalise projekti juures kõige olulisem nõue, funktsionaalne sobivus, ning ka nii mõnigi teine oluline nõue on antud töö raames väga väikese kui mitte olematu mõjuga.

3. Protsessimootorite võrdlemine nõuetega

Järgnevalt on kirjeldatud, mismoodi oleks võimalik hinnata antud kvaliteedinäitajaid mootori juures ning seejärel antakse hinnang nõude kohta mootoritest lähtuvalt, kui see on võimalik. Mootoreid käsitletakse väga üldises pildis ja puudu on kindlad ärinõuded, millest lähtuda.

Autor on läbi proovinud iga valitud mootori seadistamise ja lühikese näidisülesande. Nõuete võrdlemiseks on autor kasutusele võtnud ka kõigi kolme mootori dokumentatsioonid.

3.1 Funktsionaalne sobivus

Funktsionaalne sobivus ehk mootori suutelisus täita neid ülesandeid, mis on antud projekti raames vajalikud.

Põgusalt võrreldakse kolme mootori eripärasid, et saada esimene arusaam, millises olukorras võidakse ühte mootorit eelistada teisele. Kõiki kolme mootorit uuendatakse pidevalt ja neid toetavad aktiivsed kogukonnad.

Bonita strateegiast lähtuvalt üritatakse kogu protsesside loomine viia sellisele tasemel, kus ei pea kodeerima, et protsessi loomisega saaksid hakkama ka kõik programmeerimist mitte oskavad kasutajad. Suuremate projektide puhul võivad aga vähese paindlikkuse tõttu Bonita sisemised reeglid olla hoopis piiranguks.

Activiti ja Camunda eelisteks on konfigureeritavus ning arusaadavus. Mõlemad eeldavad kasutajatelt programmeerimise oskust. Camunda eelis Activiti ees on see, et Camunda toetab lisaks BPMN 2.0’ile (Business Process Modelling Notation) ka DMMN (Decision Managemenet Modelling Notation) ning CMMN (Case Management Modelling Notation) elemente.

3.2 Soorituse tõhusus

Soorituse tõhusus mootori juures näitab seda, kui mitu protsessi eksemplari suudab mootor paralleelselt jooksutada ilma nähtava aeglustumise või isegi protsessi kokku jooksmiseta.

Protsessimootor käivitab vastavalt reeglitele ülesandeid, mis on suures osas kas kasutaja ülesanded või teenuse ülesanded. Kasutaja ülesande puhul kuvab mootor reeglite järgi kasutajale vormi, teenuse ülesande puhul, kutsub mootor välja teenuse või käivitab koodijupi. See iseenesest on ühe protsessi eksemplari (*process instance*) juhtimine.

Soorituse tõhusust on keeruline hinnata ilma spetsiifiliste koormustestideta, kuna see on sõltuv väga mitmest asjaolust: lahenduse arhitektuurist, koodist ja riistvarast. Adekvaatse hinnangu saamiseks tuleks igal mootoril realiseerida sama arhitektuuriline lahendus ning luua võimalikul sarnane protsess ning seda paralleelselt jooksutada sadu ja tuhandeid kordi, samal ajal võrreldes mootorite jõudlust.

3.3 Töökindlus

Töökindlus protsessimootori puhul tähendab seda, kui suur või väike on tõenäosus, et protsess peaks kokku jooksma ja kui jooksebki, siis kui kerge vaevaga on võimalik seda taastada ja kui palju andmeid võib sellisel juhul kaotsi minna.

Töökindlus sõltub suuresti kogu lahenduse arhitektuurist, kus mootorit kasutatakse. Protsess võib katkeda, kui protsessimootor suhtleb välise süsteemi või teenusega, sellisel juhul aga ei pruugi olla see mootor, mis tõrgub, vaid hoopis teine osapool, mis aga ei vähenda otseselt protsessimootori töökindlust.

Töökindlust antud töö käigus arvesse ei võeta.

3.4 Ühilduvus

Ühilduvus protsessimootori puhul on väga oluline, sest protsessimootor on alati osa süsteemist. Mootoril on oma andmebaas, kus hoitakse protsessi eksemplaride muutujaid, et saaks protsessi eksemplari lõpuni juhtida.

Mootoriga on tarvis ühendada ka teisi rakendusi, et mootori omadusi saaks üldse ära kasutada. Näiteks veebirakendus, kus mootor hakkab protsessi kulgedes reeglite alusel jagama

vastavates rollides kasutajatele ülesandeid. Kindlasti on mõne lahenduse juures vajalik info saamine välistest süsteemidest või info välja saatmine.

Bonita puhul on kasutusel *connector*'id ehk ühendused, kus kasutaja kirjeldab protsessi luues, millega see tegevus protsessis ühendatakse. Kuna Bonita üritab minna võimalikult vähese programmeerimise teed, tuleb kasutajal valida ühenduse tüüp etteantud valimist, mida loojate poolt pidevalt täiendatakse. Valimi seas on veebiteenused, mis vajavad aga natuke rohkem praktiseerimist, kuid on ka ühendusi andmebaasidega ja ka SAPiga. Kuna Bonita on raamistik, kus kasutaja peab tegutsema etteantud reeglite järgi, võivad erilahenduste puhul Bonita valimid talle endale saatuslikuks saada kui kasutaja peaks hakkama ise ühendusi juurde tegema, mis on väga aeganõudev ja nõuab tehnilisi teadmisi.

Activiti on seevastu väga konfigureeritav. Võib-olla, et isegi liiga konfigureeritav. See tähendab, et arendamisel tuleb kõik vajalikud plugin'id ja raamistikud külge panna. Kindlasti on see suurema süsteemi puhul mõistlikum lahendus, mis lisab paindlikkust, kuid väiksema süsteemi puhul ei pruugi olla kõik see seadistamine kõige mugavam.

Camunda on eelneva kahe mootori vahepealne. Kuna ta on Activiti põhjal loodud, on ta väga konfigureeritav. Samas, olles nii-öelda Activiti branchist tulnud edasiarendus, on võetud sealt palju Aciviti häid omadusi ja tehtud veel paremaks. Camunda tuleb juba paketina, mille seadistamine pole nii aeganõudev, olles samas endiselt paindlik lahendus.

Kolme mootori ühilduvus on kokku võetud järgnevas tabelis.

Tabel 3. Protsessimootorite ühilduvus

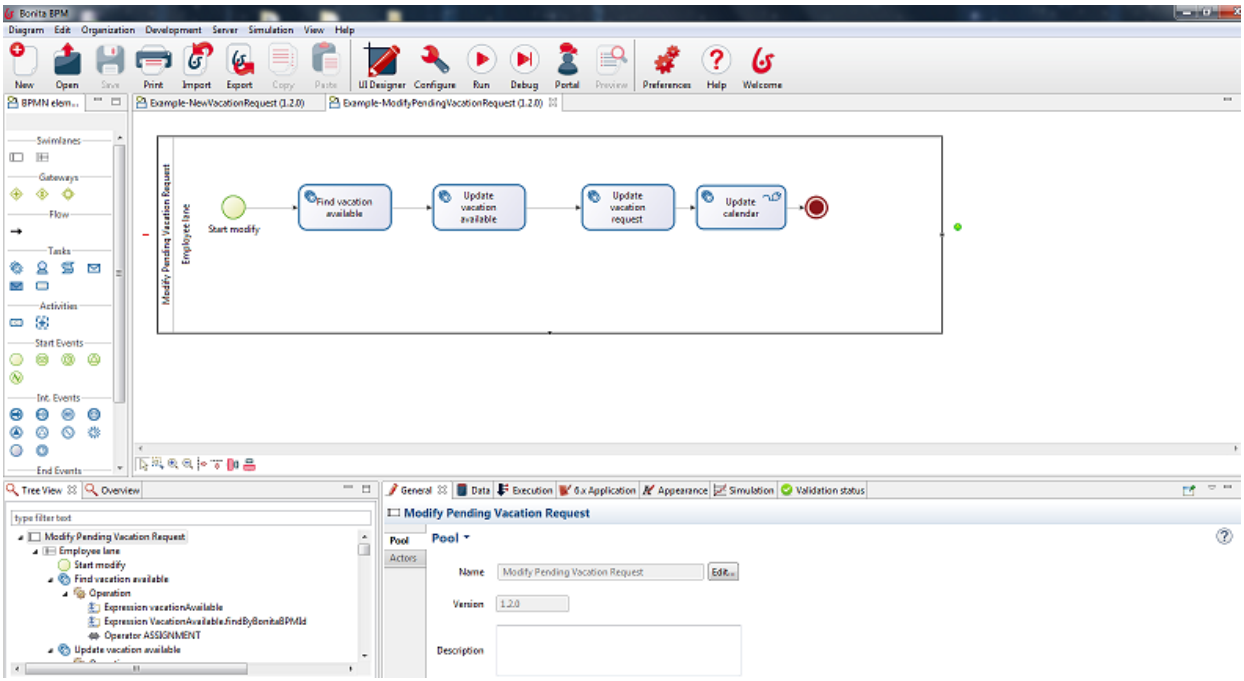
Ühilduvus	Bonita [4]	Activiti [3]	Camunda [2]
Andmebaas	MySQL, PostgreSQL, Oracle	MySQL, PostgreSQL, Oracle	MySQL, PostgreSQL, Oracle
Veebiteenused	Web service connector (SOAP)	Rest API, Web service task	Rest API, SOAP task,
Java	Java 7+ (Vajalik Studio jaoks)	Java 6 / 7 / 8	Java 6 / 7 / 8
Server	Tomcat, Jboss, Weblogic	Tomcat, Jboss, GlassFish, WebSphere	Tomcat, Jboss, GlassFish, Wildfly
Veebilehitseja	Internet Explorer 9+, Firefox, Chrome	Internet Explorer 9+, Firefox, Chrome	Internet Explorer 9+, Firefox, Chrome

3.5 Kasutatavus

Kasutatavuse puhul ei saa me rääkida lõppkasutaja kasutatavusest, sest isegi väiksemate lahenduste puhul on tõenäoliselt veebirakendus see, mis luuakse projekti täitjate poolt. Seega võib väita, et suur osa otsesest mootori kasutatavusest tuleneb samuti kliendi nõuetest. Autor vaatles seda kliendinõuet rohkem arendaja vaatenurgast – kui mugav oli ühe või teise mootori peal protsessi loomine ja selle juurutamine.

Bonita on loonud kasutajatele eraldi keskkonna, kus kasutajad saavad mugavalt luua protsesse, seejuures võimalikult vähe koodi kirjutades. Mugavus säilib kuni lahendus nõuab keerulisemaid protsesse, kus on vaja luua ühendusi, mida Bonita ise ei paku. Sellisel juhul peab kasutaja juba teadma vastavalt ühendusele siis näiteks SQL'i või mõistma põhitõdesid programmeerimise juures, et kasutaja oskaks viidata õigele kohale. Protsessi definitsiooni saab testida kiirelt, vajutades samas keskkonnas „Run". See on ka ainuke suurem erinevus juurutamise osas kolme mootori vahel. Juurutada teise keskkonda, kus mootor juba jookseb, saab exportides WAR faili, mida Tomcat oskab ise lahti pakkida.

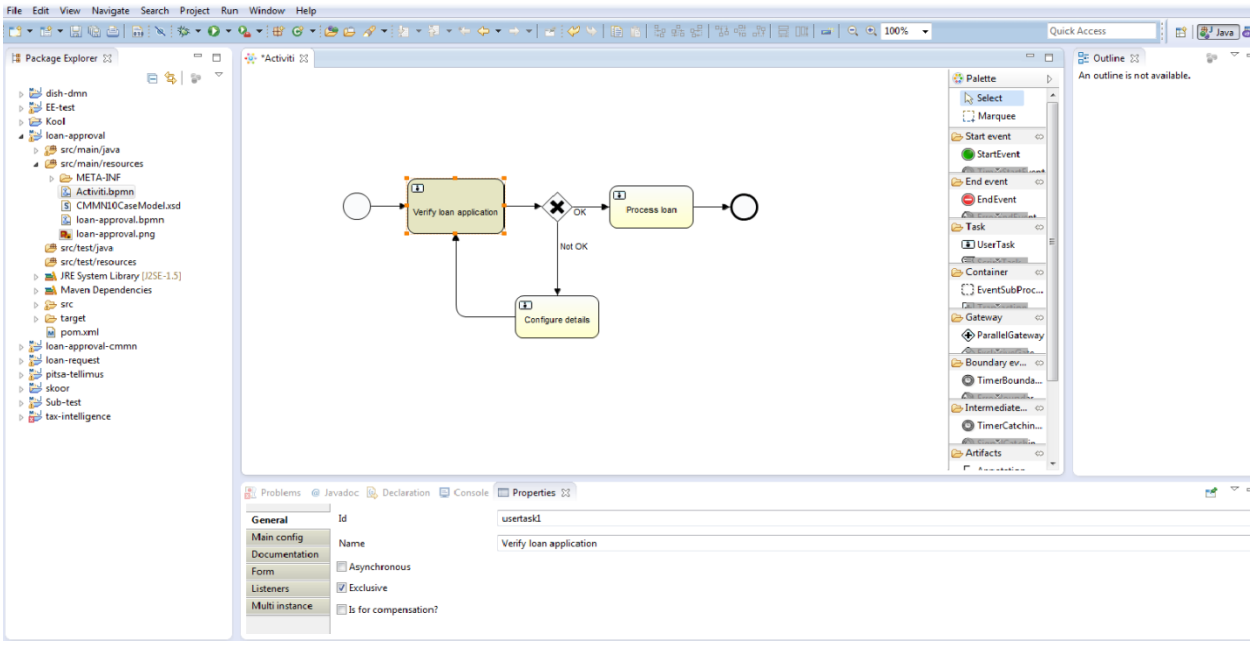
Joonisel 1 on Bonita BPM Studio, kus on näha, et rõhku on pandud eelkõige modelleerimisele. Kasutajatel tõesti pole vaja palju programmeerimisalaseid teadmisi, kuid üldisemas mõttes on nad sunnitud protsessi detaile kirjeldama Bonita reeglite ja võimekuse järgi.



Joonis 1. Bonita BPM

Activiti puhul käib protsessi loomine Eclipse'is, kuhu kasutaja peab paigaldama Eclipse'i *plugin*'i „Activiti BPMN 2.0 designer", kus saab kirjeldada protsesse. Erinevalt Bonitast, on siin vaja teada programmeerimise põhitõdesid, kuna mootorile on vaja ette anda üks rakenduse klass, mis on liideseks rakenduse ja protsessimootori vahel. Samuti on vaja iga teenuse ülesanne kirjeldada koodiga. Juhendis antakse kasutajale kood kaasa, kust on mugav sisu kopeerida, kuid failide hierarhial tuleb endiselt silma peal hoida. Programmeerimisega kokku puutunud inimestel probleemi ei teki ning lihtsamad protsessid saavad vaevata kirjeldatud. Juurutamine toimub samamoodi nagu on kirjeldatud Bonita puhul, ainukese erinevusega, et lokaalseks testimiseks on enda arvutis vaja käivitada näiteks Tomcat või Glassfish.

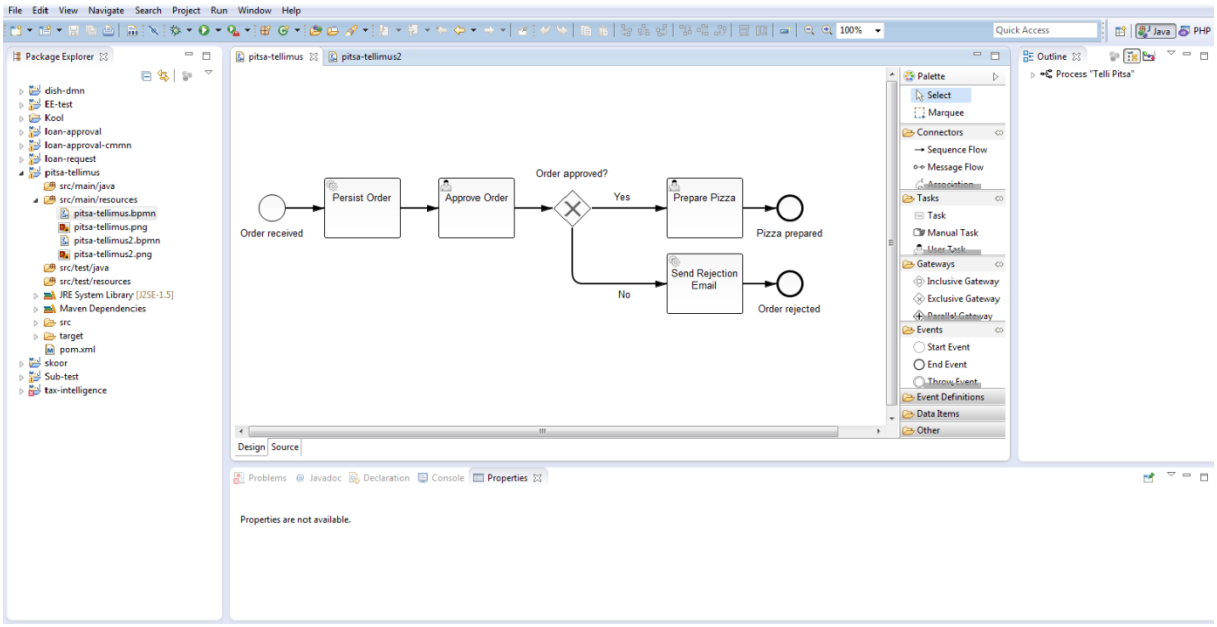
Joonisel 2 on näha Activiti modelleerimise *plugin*'i Eclipse'is. Erinevalt Bonitast tuleb protsessi spetsiifilisi detaile programmeerida Javas ning mudelis vaid viidata õigele klassile või meetodile klassi sees.



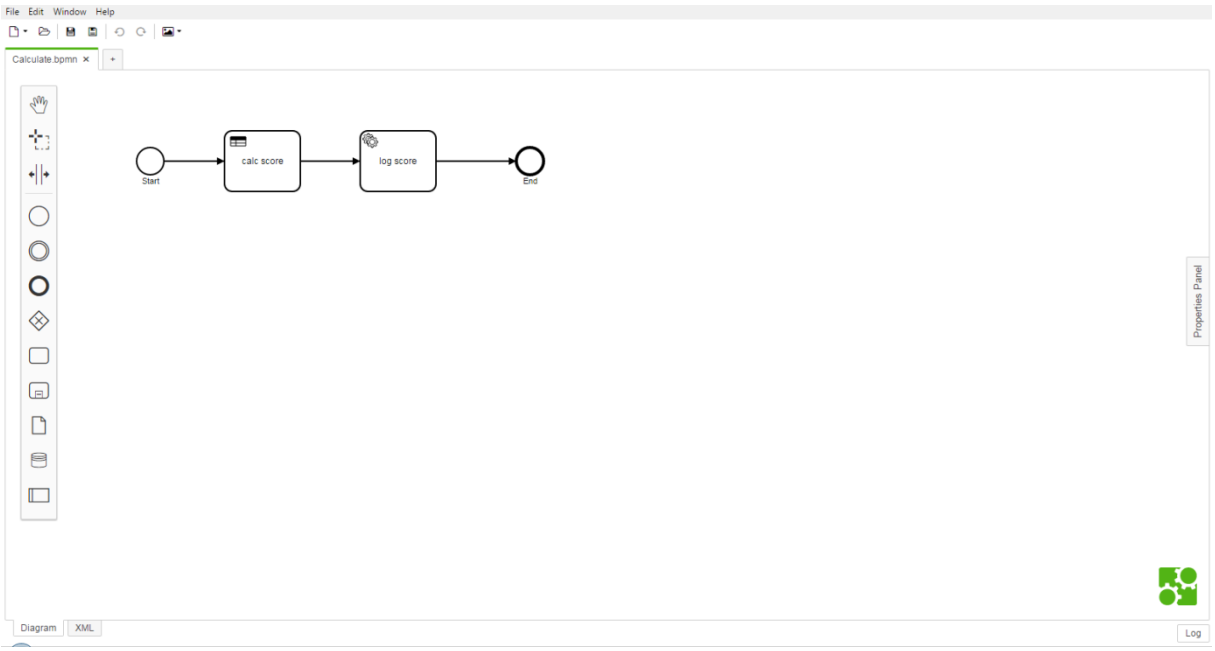
Joonis 2. Activiti BPMN 2.0 designer

Camunda protsesse saab luua kahel moel. Esimesel juhul nagu Activitis ehk siis alla laadida Eclipse'i plugin ning kirjeldada protsessi seal keskkonnas. Teisel juhul on võimalik kasutada Camunda poolt loodud eraldi programmi Camunda Modeler. Camunda ise üritab minna rohkem teist teed pidi, näitamaks, et modelleerimise jaoks ei ole vaja programmeerimiseoskust, küll aga, et protsessi lõpuni kirjeldada, on endiselt vaja programmeerimist, sellest ei saa üle ega ümber. Protsessi juurutamine käib samamoodi nagu Activitil.

Joonisel 3 on Camunda Modeler *plugin* Eclipse'is, mis on väga sarnane Activiti Modeler *plugin*'ile. Näidisprotsessi luues erinevusi nende kahe *plugini* vahel välja ei tulnud, peale kerge visuaalse erinevuse.



Joonis 3. Camunda Modeler plugin



Joonis 4. Camunda Modeler eraldiseisev programm

Joonisel 4 on näha Camunda Modeler, mis on väga puhas keskkond protsesside modelleerimiseks. Kasutades eraldiseisvat modelleerimise vahendit, on võimalik eraldada kodeerimine modelleerimisest.

Kolme mootori kasutatavuse hinnang on kokku võetud järgnevas tabelis.

Tabel 4. Protsessimootorite kasutatavus

Kasutatavus	Bonita [4]	Activiti [3]	Camunda [2]
Protsessi loomine: programmeermine	Programmeermise asemel kasutaja kirjeldab protsessi kasutades eeldefineeritud juppe.	Programmeerimine JAVAs, kasutusele saab võtta ka näiteks Spring raamistiku.	Programmeerimine JAVAs, kasutusele saab võtta ka näiteks Spring raamistiku.
Protsessi loomine: modelleerimine	Modelleerimine eraldi Studios, kus modelleerimine on kõige tähtsamal kohal – see on mugav, kiire ja arusaadav.	Modelleerimiseks on Eclipse'i plugin, mis tundub veidi aegunud, kuid ajab asja ära.	Modelleerimiseks on Eclipse'i plugin ja eraldiseisev programm. Camunda liigub modelleerimise kvaliteedi parendamise poole.
Modelleerimise keskkond	Bonita Studio	Eclipse'i keskkonnas modelleerimise *plugin*	Eclipse'is *plugin* ja eraldiseisev Camunda Modeller
Protsessi juurutamine	Lokaalis väga mugav, piisab nupule vajutamisest. Mujale serverile tuleb eksportida WAR ja pakkida lahti serveril, kus jookseb ka mootor.	Nii lokaalis kui ka välisel serveril peab jooksma mootor ning sinna tuleb paigutada projekti WAR fail.	Nii lokaalis kui ka välisel serveril peab jooksma mootor ning sinna tuleb paigutada projekti WAR fail.

3.6 Turvalisus

Protsessimootori enda turvalisust mõõta ei saa, kuna tema turvalisuse määrab süsteemi arhitektuur, milles mootor ise on vaid üks osa.

Süsteemi üldisemas mõttes on üks turvarisk iga kasutaja ise, kas andmete lekitamise mõttes või näpuviga tehes ja süsteemi kokku jooksutades. Siinkohal on kõik kolm mootorit sarnased ning kõik võimaldavad õiguste süsteemi loomist kasutajate vahel, andes ja võttes kindlatelt kasutajatelt õigusi.

Turvalisust antud töö käigus rohkem ei käsitleta.

3.7 Hooldatavus

Hooldatavuse all saab mootorite mõistes vaadelda modulaarsust ehk muudatuse mõju teistele komponentidele, taaskasutatavust, analüüsitavust ja testitavust.

Bonita puhul muudatuse tegemine mõjutab tervet protsessi. Üldiste muudatuste puhul tuleb muuta igat protsessi eraldi. Muudatuse tegemisel tuleb terve protsess uuesti juurutada. Taaskasutavus Bonitas tähendabki Bonita connectorite ja teiste valmis klotside kasutamist. Bonita puhul toimub enne protsessi juurutamist protsessi konfigureerimine. Konfigureerimisel on võimalik seadistada protsess testimiseks. On võimalik konfigureerida protsessi nii, et testimiseks määratud protsessi saavad käivitada ainult teatud kasutajad. Monitoorimine ja analüüsimine toimub Bonitas käsitsi pärast protsesside juurutamist, kasutades Bonita monitooringu veebirakendust, kus on mitmesugust statistikat protsessi definitsiooni ja sammude kohta.

Activiti puhul on võimalik muuta nii mootori loogikat ja ka spetsiifilist protsessi. Koodi puhul on võimalik taaskasutada ühte koodijuppi igas protsessis. Analüüsimine toimub Activitis käsitsi olemasolevate protsesside põhjal. Monitoorimise veebirakendust pakub Activiti ainult *Enterprise* versioonis. Activitis on võimalik protsessi koodid üle kontrollida Unit Testinguga. Protsessi mudeli loogika kontrollib samuti *plugin* ära, kuid protsessi voogu tuleb kontrollida käsitsi peale juurutamist.

Camunda on väga sarnane Activitiga. Võimalus on muuta nii mootori loogikat kui ka spetsiifilisi protsesse puudutavaid meetodeid. Samamoodi on võimalik valmis koodi taaskasutada mitme erineva protsessi puhul. Koodi on võimalik testida Unit Testidega ja mudeli loogikat valideeritakse nii Eclipse'i plugin'is kui ka modelleerimise keskkonnas. Camunda veebirakenduses *Cockpit* on näha kõiki juurutatud protsesse ja *Enterprise* versioonis lisandub veel heatmapping ehk Camunda poolt loodud automaatne statistika, mis näitab protsesside nõrku kohti sellel viisil, milleks protsessis kulub rohkem aega. Protsessi voogu tuleb testida aga käsitsi.

Järgnevas tabelis on kokku võetud mootorite hooldatavus.

Tabel 5. Protsessimootorite hooldatavus

Hooldatavus	Bonita [4]	Activiti [3]	Camunda [2]
Testimine	Studio valideerib mudeli loogikavigade vastu. Läbi studio on võimalik testida protsesse eraldi keskkonnas. Võimalik testida eraldi *connectoreid*. Võimalik ka konfigureerida juurutamine testimiseks ainult teatud kasutajatele.	Koodi testimine Unit automaattestidega. Modelleerimise keskkond kontrollib mudeli loogikavigu. Protsessi voogu peab kontrollima käsitsi.	Koodi testimine Unit automaattestidega. Modelleerimise keskkond kontrollib mudeli loogikavigu. Protsessi voogu peab kontrollima käsitsi.
Modulaarsus	Üldisema muutuse puhul tuleb igat protsessi eraldi muuta. Muutes ühendusi üldiselt on võimalik muudatus läbi viia mitmes protsesis korraga, sellisel juhul peab kasutama omaloodud ühendusi.	Võimalik muuta nii spetsiifilisi protsesse kui ka üldisemat loogikat, mis mõjutab kõiki protsesse, mis seda koodi kasutab.	Võimalik muuta nii spetsiifilisi protsesse kui ka üldisemat loogikat, mis mõjutab kõiki protsesse, mis seda koodi kasutab.
Taaskasutamine	Taaskasutamine toimub Bonitas olemasolevate klotsidega.	Kood on taaskasutatav mitme protsessi juures.	Kood on taaskasutatav mitme protsessi juures.
Analüüsimine	Bonita pakub analüüsimiskes veebirakendust, kus on võimalik näha statistikat olemasolevate protsesside kohta.	Monitoorimist pakub Activiti Enterprise versioonis.	Camunda veebirakenduses Cockpit on võimalik jälgida jooksvaid protsesse ja teha statistikat. Enterprise versioonis lisandub heatmapping.

3.8 Porditavus

Porditavuse all saab protsessimootoreid võrrelda selle alusel, kui keeruline on mootori paigaldamine teise keskkonda ja seadistamine uuel keskkonnal. Kuna tegemist on üldiste kvaliteedinõuete võrdlemisega siis ei võta arvesse erilahendusi ning paigaldamisel kasutame vaikimisi määratu väärtusi.

Bonita pakub meile mootori installeerimiseks pakke, kus on kõik vajalikud asjad olemas. Pakid on nii serveriga koos olevad pakid kui ka ilma serverita pakid. Serveritega pakkides tuleb kõigepealt konfigureerida rakendusserveril ja seejärel pakkida lahti Bonita mootor serveril. Samuti on ka detailsed õpetused mootori ühendamiseks veel teise andmebaasi külge.

Activiti ja Camunda töötavad siinkohal sama moodi. Mõlema tööle panemiseks peab keskkonnas olema JDK 1.6+ ja süsteemi muutujatest JAVA_HOME seadistatud. Teisena peab samuti olema rakendusserver näiteks Tomcat. Seejärel peab olema serverisse installeeritud mootor ning lõpetuseks protsesside juurutamiseks tuleb lihtsalt rakendusserveri webapps kausta kopeerida protsessi projekti WAR fail. Ka Camunda ja Activiti puhul on detailsed õpetused andmebaasi konfigureerimise kohta.

Porditavus on kolme mootori peal väga sarnane. Ainukeseks erinevuseks Bonita puhul, on tema installeerimisel saab lisana alla laadida ka vajaliku Java, mida ei pea hiljem ise käsitsi süsteemi muutujatesse salvestama.

3.9 Nõuete võrdluse kokkuvõte

Eelnevate peatükkide sees sai võrreldud mootoreid välja toodud kvaliteedinõuetega. On näha, et üldiste nõuete püstitamine tootele, mida hakatakse kasutama väga spetsiifilistes lahendustes on väga keeruline. Võrdlusest jäid kõrvale kaheksast nõudest kolm: funktsionaalne sobivus, turvalisus ja töökindlus. Need nõuded said kirjeldatud selles võtmes, mismoodi neid tuleks kontrollida nõuetega.

Kokkuvõtteks võib öelda, et ärinõuete puudumise tõttu ei olnud võrdlus kuigivõrd efektiivne ja jäi väga pinnapealseks. Selle peale võib öelda, et võrdlemise eesmärk sai täidetud vaid osaliselt.

Andmaks siiski veidike ülevaadet mootorite kohta on järgnevalt kokku võetud valitud mootorite eripärad. Eripärad on välja toodud neid loonud firmade vaatenurgast, andmaks vähesel määral ka ülevaate firmade visioonide ja strateegiate kohta.

Tabel 6. Mootorite võrdluse kokkuvõte

Kokkuvõte	Bonita	Activiti	Camunda
Toote ja firma strateegia	Protsesse peab saama kirjeldatud ainult modelleerides ja ilma programmeerimiseta.	Modelleerimine ja programmeerimine JAVAs on võrdsed, neid peaks tegema koos.	Modelleerimine ja programmeerimine JAVAs on võrdsed, kuid neid peaks tegema eraldi. Camunda liigub samuti modelleerimise studio kasutamise poole, kuid programmeerimist nad kaotada ei kavatse.
Strateegia eelised	Kasutajad ei pea oskama programmeerimist, mudeleid kirjeldatakse vastavalt Bonita poolt seatud reeglistikule.	Kasutaja on võimalik protsesse kirjeldada väga detailselt ja omal meetodil. Ei pea järgima ette määratud malle.	Kergesti kasutatav keskkond analüütikule, saab keskenduda modelleerimisele ja vähem aega õppimisele ja konfigureerimisele. Programmeerijad saavad endiselt kirjeldada protsesse lõplikult väga detailselt. Paindlikus säilib.
Strateegia miinused	Bonita reeglistik piirab ta paindlikust, seega ei pruugi Bonita mootoril olla kõik keerulisemad protsessid kergesti või üldse realiseeritavad.	Põhikasutaja programmeerimise oskus, mistõttu peavad programmeerija ja analüütik tihedalt koostööd tegema.	Äriinimesed modelleerivad ja programmeerijad kodeerivad eraldi keskkondades.

4. Näidisprotsessi loomine

Järgnevalt on välja toodud näidisprotsessi loomine, kasutades selleks Eclipse'i arenduskeskkonda. Seejärel valminud protsessi juurutamine ja käivitamine Camunda protsessimootoril, näitamaks ühte kolmest praktilisest ülesandest, mis autor läbi tegi. Samuti tuuakse välja ka edasised sammud, mida Camunda protsessimootoril on võimalik teostada.

Näidisprotsessi demonstreerimiseks valis autor Camunda eelkõige seetõttu, et Camundaga kaasatulevad veebirakendused on kolmest mootorist kõige mitmekülgsemad, näitamaks seda, kuidas saab loodavaid protsesse ära kasutada.

4.1 Protsessi loomine Eclipse'is

Näidisprotsessi loomisel Camundas oli paralleelselt abiks Camunda „Getting started with Camunda BPMN 2.0". On võimalik alla laadida ka valmis projekte proovimiseks ja katsetamiseks, kuid antud töö nägi ette, et autor katsetab paigaldusest kuni protsessi juurutamiseni mootori ise läbi.

Tähtsaim asi kogu näidisprotsessi loomise ajal oli hoida silm peal projekti kaustade hierarhial, et käsitsi juurde lisatud asjad, eriti konfigureerimise faasis, saaksid õigetesse kohtadesse salvestatud.

Protsessi loomise eeldusteks on:

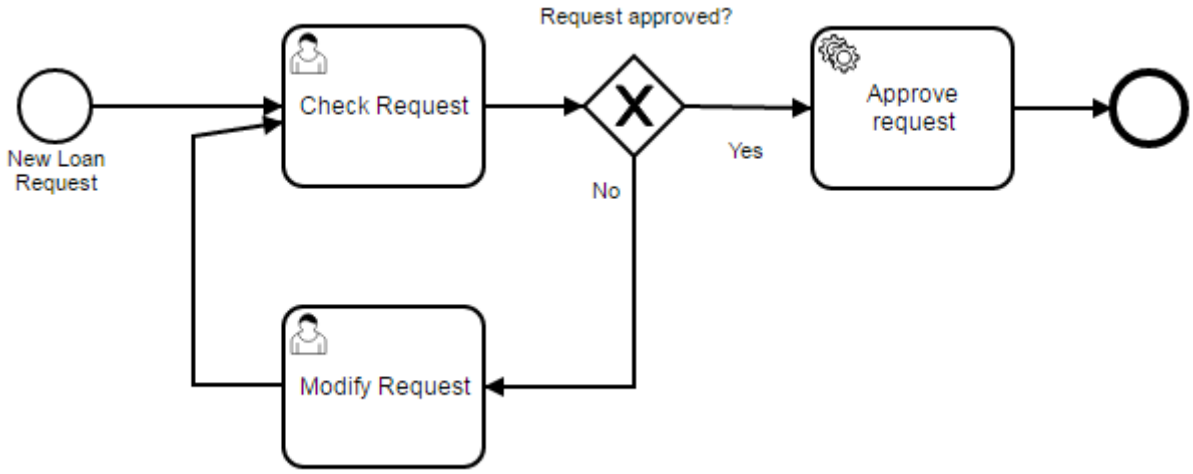
- Java JDK 1.6+
- Eclipse, Camunda BPMN Modeler *plugin*'iga

Camunda protsessid luuakse Maven projekti. Seega alustada tuleb uue Maven'i projekti loomisega. Seejärel tuleb konfigureerida projekt, anda projektile teada Camunda olemasolust. Tuleb lisada Maven *dependencies* ehk sõltuvused. Maveni sõltuvused pom.xml'is on välja toodud Lisas 1.

Järgmisena on vajalik luua liides rakenduse ja Camunda vahel. Selleks on vaja luua Java klass, kus on oluline see, et klass oleks ServletProcessApplication'i pikendus.

Viimane samm seadistamisel on mootorile kirjeldamine, mis mootor täpsemalt tegema peab protsessi definitsiooni juurutamisel. Seadistused process.xml sees on kirjeldatud Lisas 2.

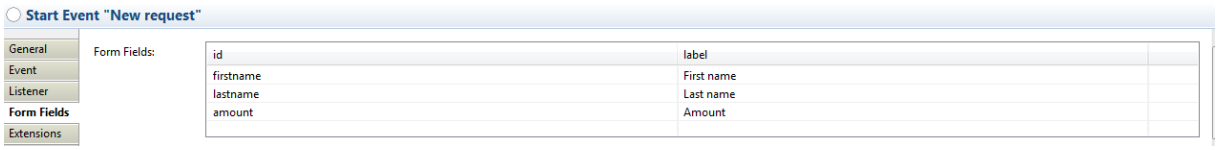
Lõpuks seadistatud rakenduses on võimalik luua protsessi definitsioon.



Joonis 5. Näidisprotsessi mudel

Protsess on väga lihtsustatud ning tõeliselt päris selline protsess ei toimiks. Selline protsess loodi selleks, et lihtsustatult anda ülevaade Camunda mootori toimimisest.

Keegi *taotleja* soovib teha laenutaotlust. Selleks täidab ta vormi, mis on kirjeldatud Start event detailides (joonis 6). Seejärel on keegi töötaja, kes taotluse üle vaatab. Töötaja otsustada on, kas lubada sellist laenu või mitte. Tema otsusest sõltuvalt väärtustatakse protsessis muutuja *approved,* mille järgi mootor otsustab, kumba jada protsess jätkab. Olgu, et esimene kord töötaja lükkas taotluse tagasi, kuna summa oli liiga suur. Seega *approved* muutuja väärtus on *false* ja järgmiseks sammuks tuleb Modify Request, kus *taotlejal* on võimalik summa väärtust muuta. Uuesti taotlust kinnitades läheb taotlus jälle töötajale üle vaatamiseks. Seekord töötaja kinnitab taotluse, protsess läbib veel ühe Service task'i enne, kui jõuab jada lõppu. Siinkohal Service task logib lihtsalt väikese info, kuid suuremas lahenduses võib see Service task näiteks välja kutsuda mõne välise teenuse.



Joonis 6. Näidisprotsessi Start Event detailid

Olles rahul loodud protsessiga, tuleks teha projektist *Maven Install*, mille tulemusena saab projekti WAR faili.

4.2 Protsessi juurutamine Camundas

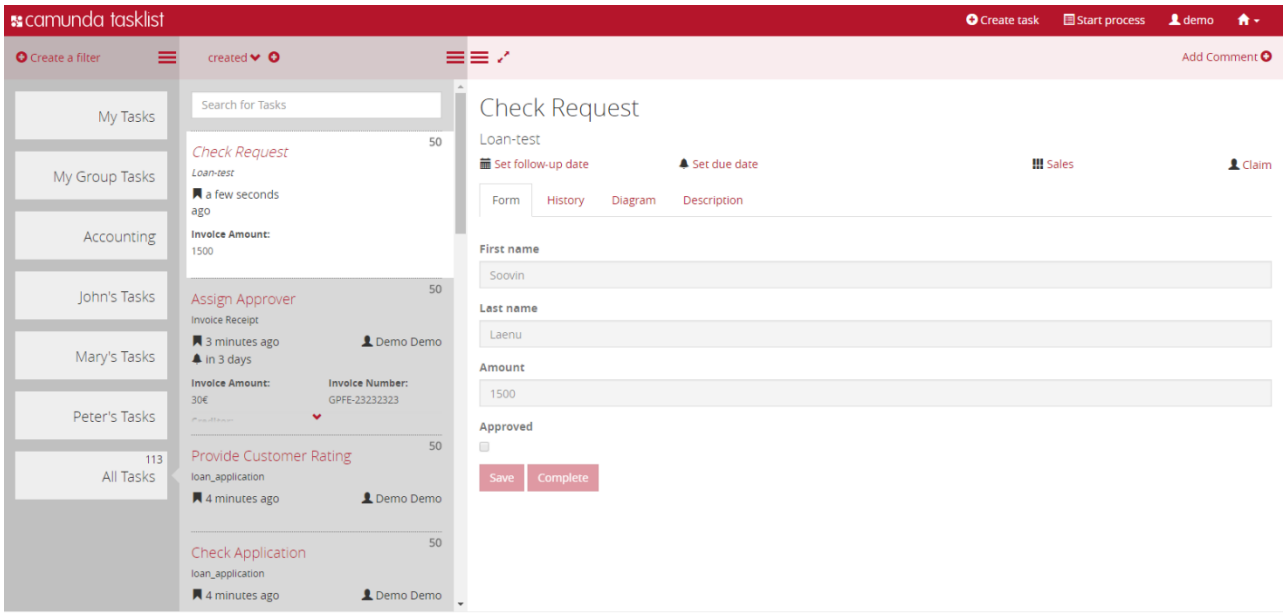
Järgnevalt on kirjeldatud eelnevas punktis loodud protsessi juurutamist ning läbi proovimist Camunda mootoril.

Camundat alla laadides tuleb kaasa ka Tomcat, mis tuleks käivitada. Seejärel tuleb Tomcati *webapps* kausta tõsta näidisprotsessi projekti WAR fail, mille Tomcat mõne hetke pärast lahti pakib.

Katsetades lokaalselt ja töötava Tomcatiga, saab nüüd Camunda kätte aadressil *http://localhost:8080/camunda-welcome/*, kus Camunda pakub liikumist oma kolme veebirakendusse – Tasklist'i, Cockpit'i ja Admin'i.

4.2.1 Camunda veebirakendus: Tasklist

Camunda Tasklist, eestkätt mõeldud lõppkasutajatele, on üks kolmest Camunda poolt pakutavast veebirakendusest, mille eesmärk on *Human Workflow Management* ehk inimeste tööprotsesside haldamine. Tasklist veebirakenduses on võimalik algatada loodud näidisprotsessi, jälgida sisselogitud kasutaja ülesandeid ja neid ka täita. Joonisel 7 kõige vasakpoolsemas tulbas on näha ülesannete filtreerimist, järgmises tulbas filtreeritud ülesandeid ja seejärel suure vormina aktiivset ülesannet.

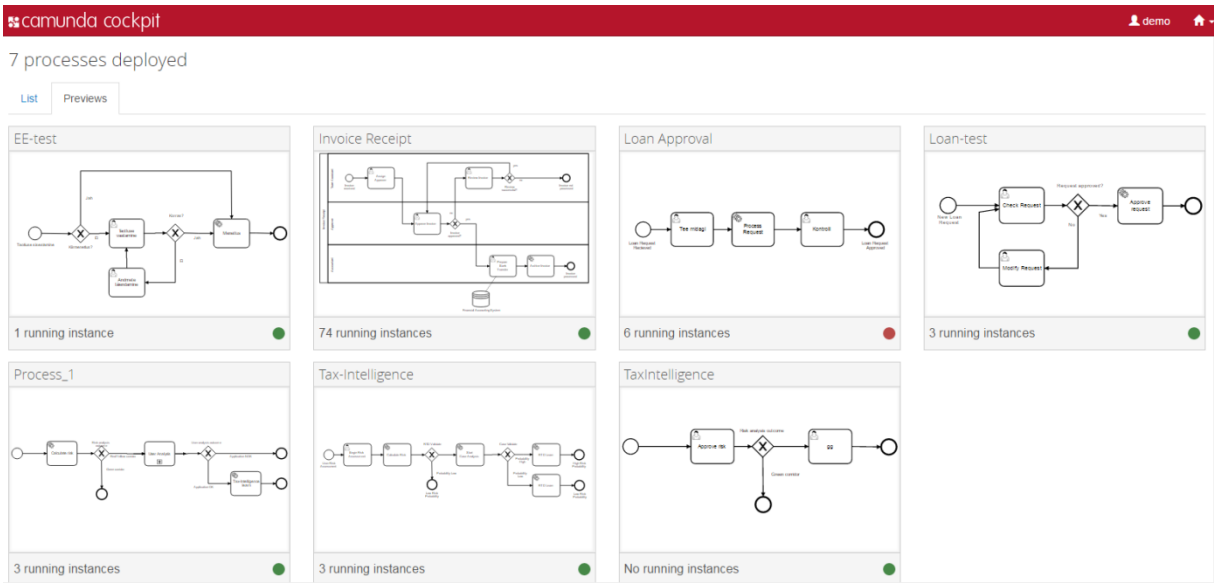


Joonis 7. Camunda Tasklist

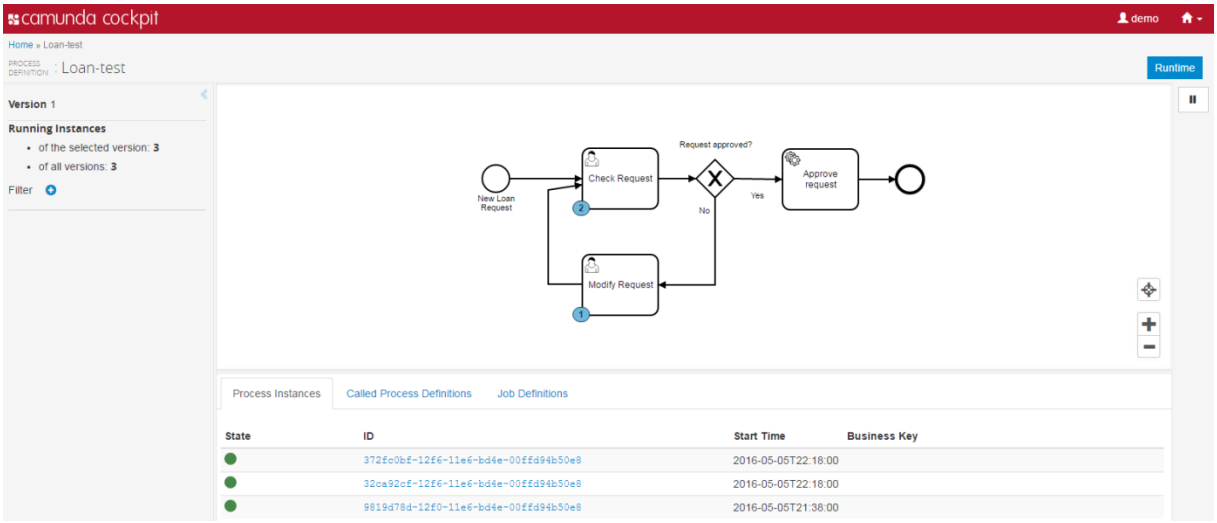
Näidisprotsessi algatamisel sisestab kasutaja taotluse algandmed ning seejärel suunab protsessi juba Camunda. Näidisprotsessis sai defineeritud, et esimese taski peab saama keegi *Sales* grupist. Seega, kõik kasutajad, kes on määratud grupis, näevad tekkinud ülesannet ning saavad selle endale *Claim*'ida ehk määrata, pärast mida saab antud ülesannet teha ainult ülesande külge määratud kasutaja. Kasutajale kuvatakse defineeritud vorm, kus kasutaja saab vaadata üle taotluse ja siis otsustada, kas rahuldada taotlus või mitte – sellisena nagu sai määratud ka protsessi definitsioonis.

4.2.2 Camunda veebirakendus: Cockpit

Camunda Cockpit on oluline tööriist ärile ja administraatoritele, mille eesmärk on monitoorida Camundas juurutatud protsesse. Cockpitis näidatakse kõigepealt kõiki olemasolevate protsesside definitsioone (joonis 8) ja võimalik on ka vaadata igat protsessi detailsemalt, kus on näha valitud protsessi definitsiooni käimasolevaid protsessi eksemplare (joonis 9).



Joonis 8. Camunda Cockpit: protsesside loetelu



Joonis 9. Camunda Cockpit: protsessi detailvaade

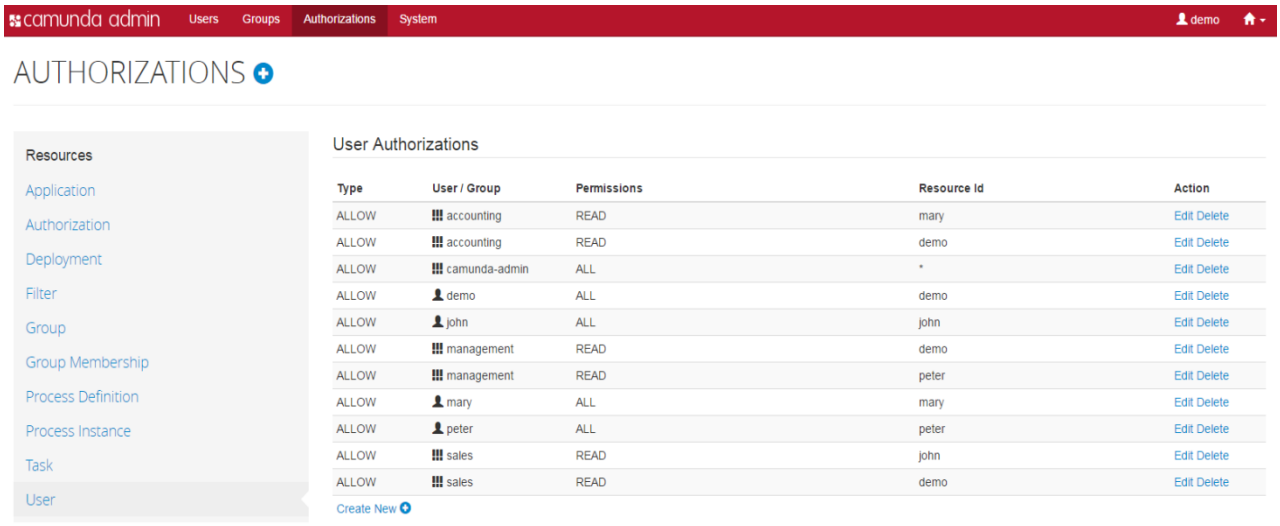
Cockpit on väga võimas tööriist, selgitamaks välja protsesside nõrku kohti, mida oleks seejärel võimalik parendada. Detailvaates on näha, kui mitu protsessi eksemplari on parasjagu millises protsessi sammus ning valides kindla eksemplari, on näha ka protsessi jooksvaid muutujaid ja ajalugu – milline kasutaja mida ja millal muutnud või lisanud on.

Camunda toetab ka versioneerimist, seega on võimalik protsessist uuem versioon juurutada kahjustamata pooleliolevaid protsessi eksemplare. Vasakul ääres (joonisel 9) on näha ka

protsessi versioon, mitu protsessi eksemplari on parasjagu käimas ning mitu neist on käesoleva versiooni protsessid.

4.2.3 Camunda veebirakendus: Admin

Camunda Admin, kolmas veebirakendus, on mõeldud eelkõige kasutajate ja gruppide haldamiseks ning õiguste jagamiseks. See on oluline tööülesannete haldamisel ning samuti ka eristamaks õigustega neid, kellel on rohkem võimu. Joonisel 10 on näha kasutajate õigused.



Joonis 10. Camunda Admin

Camunda kokkuvõtteks võib öelda, et tegemist on väga kergesti õpitava mootoriga. Järgides juhendit oli näidisprotsessi loomine kerge, kuna juhend oli väga detailselt kirjutatud. Näidisprotsessi läbi tehes on võimalik ka loodud protsessi muuta omanäolisemaks tänu sellele, et põhilised funktsionaalsused on sellega läbi võetud. Camundaga kaasa tulevad veebirakendused andsid mitmekülgse ülevaate, mida on protsessimootoriga võimalik teha seoses ühe kirjeldatud protsessiga.

Kokkuvõte

Selle lõputöö eesmärk oli uurida erinevaid protsessimootoreid ja võrrelda neid üldisemate kvaliteedinäitajatega, lihtsustamaks tulevikus protsessimootorite valikut, kui on vajadus protsessimootori järele projektis.

Kõigepealt määratleti nõuded, millega hakati võrdlema mootoreid. Sellisel moel saab lõppotsuse teha käesoleva töö lugeja, milline mootor sobib üheks või teiseks lahenduseks paremini. Järgmise sammuna proovis autor läbi iga mootori paigaldamise, lõi ühe näidisprotsessi, juurutas selle ja proovis selle elutsüklit mootoril. Seejärel kirjeldas nõudeid iga mootori vaatenurgast lähtuvalt.

Käesolev lõputöö näitas, et selline protsessimootorite võrdlemine on kasulik ainult mootorite üldiste erinevuste leidmiseks. Adekvaatse hinnangu saamiseks tuleks mootoreid võrrelda vastu samu kvaliteedinäitajaid mõnest lahendusest lähtuvalt, kus oleksid püstitatud mootoritele ka kindlad ärireeglid. Sellisel moel oleks võimalik võrrelda mootoreid kõikide kaheksa kvaliteedinäitajaga. Samuti saaks suurendada protsessimootorite loetelu, kuna läbi prooviti ainult kolm – Camunda, Activiti ja Bonita. Raskemaks muudab aga loetelu suurendamise asjaolu, et kõik mootori tarkvarad ei ole tasuta kättesaadavad. Samuti on mootorite õppimine aeganõudev.

Lõputöö täitis püstitatud eesmärgid ainult osaliselt. Lõputöö annab küll ülevaate valitud mootorite erinevuse kohta ning on näha mootorite üldiseid iseloomujooni, kuid ärinõuete puudumisel, jäid mootorite erinevused vähemõjuvaks või liiga abstraktseks.

Summary

The aim of this thesis was to compare three different Process Engines and through comparison bring out the differences to simplify the choice between different Engines for any one searching to use a Process Engine in their system.

The engines were compared against the Software product Quality Requirements and Evaluation – Quality model. However due to the absence of business requirements the comparison of the engines was more difficult and not all quality requirements could be used. This study should be conducted when specific business requirements are present. In this case all eight Quality Requirements could be taken in account when comparing the engines. Also for future studies more engines could be used.

The thesis fulfilled its purpose of providing an overview of the differences between the engines however the differences found were too abstract and mostly only give the general idea.

Kasutatud kirjandus

1. Jaak Tepandi „Tarkvara protsessid, kvaliteet ja standardid“ [WWW] http://tepandi.ee/tks-loeng.pdf (versioon 11.01.2016)
2. Camunda dokumentatsioon [WWW] https://docs.camunda.org/manual/7.4/ (01.05.2016)
3. Activiti dokumentatsioon [WWW] http://activiti.org/userguide/index.html (01.05.2016)
4. Bonita dokumentatsioon [WWW] http://documentation.bonitasoft.com/bos-version/bonita-bpm-72-0 (01.05.2016)
5. It-teatmik [WWW] http://www.vallaste.ee
6. Technopedia [WWW] https://www.techopedia.com/definition/26689/business-process-engine-bpe (01.05.2016)

Lisa 1

Näidisprotsessi pom.xml

```
<project xmlns=http://maven.apache.org/POM/4.0.0 mlns:xsi=http://www.w3.org/2001/XMLSchema-instance
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>org.camunda.bpm.getstarted</groupId>
  <artifactId>loan-approval</artifactId>
  <version>0.1.0-SNAPSHOT</version>
  <packaging>war</packaging>
  <dependencyManagement>
    <dependencies>
      <dependency>
        <groupId>org.camunda.bpm</groupId>
        <artifactId>camunda-bom</artifactId>
        <version>7.4.0</version>
        <scope>import</scope>
        <type>pom</type>
      </dependency>
    </dependencies>
  </dependencyManagement>
  <dependencies>
    <dependency>
      <groupId>org.camunda.bpm</groupId>
      <artifactId>camunda-engine</artifactId>
      <scope>provided</scope>
    </dependency>
    <dependency>
      <groupId>javax.servlet</groupId>
      <artifactId>javax.servlet-api</artifactId>
      <version>3.0.1</version>
      <scope>provided</scope>
    </dependency>
  </dependencies>
  <build>
    <plugins>
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-war-plugin</artifactId>
        <version>2.3</version>
        <configuration>
          <failOnMissingWebXml>false</failOnMissingWebXml>
        </configuration>
      </plugin></plugins></build></project>
```

Lisa 2

Näidisprotsessi processes.xml

```
<?xml version="1.0" encoding="UTF-8" ?>
<process-application
    xmlns="http://www.camunda.org/schema/1.0/ProcessApplication"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <process-archive name="loan-approval">
    <process-engine>default</process-engine>
    <properties>
      <property name="isDeleteUponUndeploy">false</property>
      <property name="isScanForProcessDefinitions">true</property>
    </properties>
  </process-archive>
</process-application>
```