TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Artjom Aleksejev 163918IABB

TARKVARA KOMPONENTIDE KORDUVKASUTUSE METOODIKA ANALÜÜS ADM INTERACTIVE OÜ NÄITEL

Bakalaureusetöö

Juhendaja: Mart Roost,
MSc

Kaasjuhendaja: Rain Benrot,
ADM Interactive OÜ,
COO

Tallinn 2020

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.

Autor: Artjom Aleksejev

18.05.2020

Annotatsioon

Kõik suured IT-ettevõtted, mis ei seisa paigal ja pidevalt arenevad, on huvitatud programmeerimisprotsessi lihtsustamisest ja ka selle kulude vähendamisest. Tarkvara korduvkasutamise protsess on üks aktuaalsetest lahendustest, mida paljud ettevõtted on juba hakanud praktiseerima.

Praeguse seisuga ainult üksikud ADM Interactive OÜ arendajad üritavad oma koodis varem tehtud lahendusi uuesti kasutada. Toimiva tarkvara korduvkasutusetehnika rakendamiseks on tarvis hästi defineeritud strateegilist protsessi, mida tänapäeval ei ole veel käesolevas firmas ette võetud.

Käesoleva lõputöö eesmärgiks on anda põhjalik kirjanduse ülevaade tarkvara korduvkasutamisest, analüüsida ADM'i arendusüksuste praegust arendusprotsessi (AS-IS) ning viia läbi küsitlus Eesti suurte IT-firmade vahel nende tarkvara korduvkasutamise seisu välja selgitamiseks. Saadud tulemuste põhjal luuakse tarkvara komponentide korduvkasutamise metoodikat. Välja töötatud lahendus (TO-BE) peaks ADM'is tarkvara tootmise aega ja sellega kaasnevaid kulusid vähendama ning tõstma toodete kvaliteeti.

Antud töö tulemuseks on modelleeritud ja kirjeldatud komponendipõhine tarkvara korduvkasutamise protsess, mis laiendab olemasolevat kosemudeli tüüpi arendusprotsessi. Lisaks on defineeritud metoodikat toetava keskkonna nõuded.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 33 leheküljel, 6 peatükki, 14 joonist.

Abstract

Analysis of software components reuse methodology on the example of ADM Interactive OÜ

Constantly evolving large IT companies try to simplify its programming process and reduce expenses. Software reuse process is one of the solutions that many companies have recently started practicing.

Currently, only a few developers at ADM Interactive OÜ are trying to reuse previously made solutions. Implementing a working software reuse technique requires a well-defined strategic process, which has not been undertaken by the company yet.

The main goal of current thesis is to analyse the component-based methodology, the current development process of ADM's development units (AS-IS) and to conduct a survey among largest Estonian IT companies to find out their software reuse situation. Based on the obtained results, a methodology for reusing software components was created. The developed solution (TO-BE) should reduce the software development time and related expenses for ADM and also increase the quality of company's products.

As the result, a component-based software reuse process was modelled and described. This model extends the existing waterfall model type development process. In addition, the requirements of methodology supporting environment have been defined.

The thesis is in Estonian language and contains 33 pages of text, 6 chapters and 14 figures.

Lühendite ja mõistete sõnastik

AJAX	*Asynchronous JavaScript And XML*, asünkroonne JavaScript ja XML
Artifakt	UML kujul vara
Back-end	Tagarakendus
Black-box reuse	Musta-kasti korduvkasutamine
CBSD	*Component-based system development*, komponendipõhine tarkvara arendamine
CMS	*Content Management System*, sisuhaldussüsteem
COTS	*Commercial Off-The-Shelf*, kommertsi tooted
Custom CMS	*Custom Content Management System*, kohandatud sisuhaldussüsteem
CVS	*Concurrent Versions System*, versioonihalduse süsteem
Debug	Silumine
Drupal	Vabavaraline sisuhaldussüsteem
Front-end	Esirakendus
HP	*Hewlett-Packard*, IT-ettevõte
Laravel	PHP raamistik
PHP	*Hypertext Preprocessor*, hüperteksti eeltöötleja
SCRUM	*Sprint Continious Rugby Unified Methodology*, agiilse tarkvara arendamise raamistik
UI	*User Interface*, kasutajaliides
UML	*Unified Modeling Language*, ühtne modelleerimiskeel
UX	*User Experience*, kasutajakogemus
White-box reuse	Valge-kasti korduvkasutamine
Wordpress	Avatud lähtekoodiga sisuhaldussüsteem

Sisukord

Jooniste loetelu

1 Sissejuhatus

Vanasti oli tarkvara funktsionaalsus keeruline ja selle koodi maht pidevalt tõusis. Nüüd on tarkvara arendamisel uus trend, et muuta tarkvara võimalikult lihtsamaks ja tõhusamaks. Tulenevalt sellest püüavad kõik IT-ettevõtted oma arendusprotsessi võimalikult optimeerida ja uute toodete välja arendamiseks kuluvat aega vähendada. Olemasoleva tarkvara korduvkasutamine teistes rakendustes või selle laenamine muudest allikatest on juba ammu saanud kõigi programmeerijate jaoks normiks. Tarkvara korduvkasutamine on süsteemide ehitamise meetod, kus võetakse kasutusele lahendused, mis olid ennem juba arendajate poolt välja arendatud [1].

ADM Interactive OÜ on Eesti suurimaid digiagentuure, mis fokuseerib veebilahenduste ja e-kommertside arendamisele ning aitab klientidel digistrateegiaid välja töötada aastast 1997 [2]. Kui vaadata ettevõtte üldpilti, siis täna puudub ADM'il hea ülevaade kõikidest lahendustest, mida on seni välja arendatud. Sellest tulenevalt firmas pole väljakujunenud protsessi, mille järgi saaks uute projektide puhul varem valmistanud tarkvara komponentidest uusi tooteid veelgi lihtsamal kujul ja efektiivsemalt luua.

Lõputöö eesmärk on kõigepealt analüüsida ADM'i arendusprotsessi hetkeolukorda, et leida ühiseid tarkvara komponendid, mida saaks korduvkasutada. Lisaks tuleb läbi viia küsitlust Eesti suurte IT-firmade vahel, et saada terviklik pilt nende AS-IS tarkvara korduvkasutamise olukorrast. Küsimustiku vastused aitavad selgitada välja, kuidas on tänapäeva firmadel korduvkasutamise protsess lahendatud. Saadud tulemuste põhjal töötatakse välja tarkvara komponentide korduvkasutuse metoodika, mida saaks lähitulevikus ADM'is juurutada.

Käesolevas töös käsitletakse kirjanduse ülevaate osas kõik võimalikud tarkvaralised korduvkasutamise tüübid ja täpsemalt pööratakse tähelepanu komponendipõhisele tarkvara korduvkasutusele. Seejärel autor analüüsib ADM'i AS-IS korduvkasutamise seisu ning modelleerib hetkelist arenduse protsessi. Küsitluse taustauuringu kui ka analüüsist saadud tulemuste alusel oli autori poolt pakutud TO-BE lahendus.

2 Kirjanduse analüüs

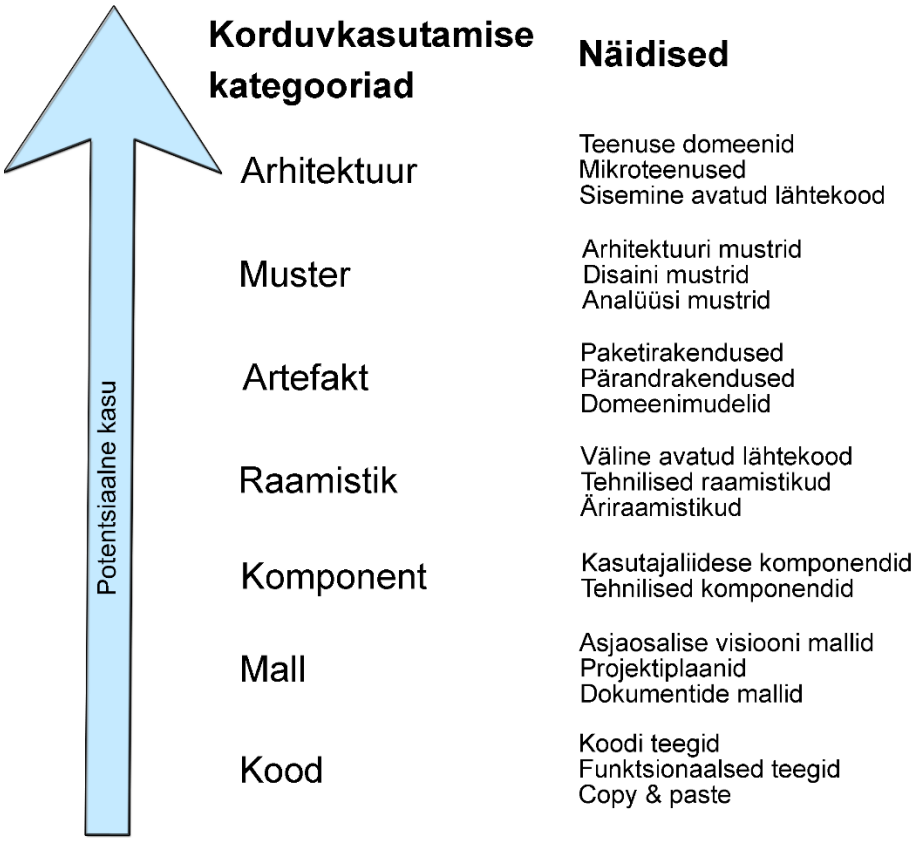
2.1 Ülevaade tarkvara korduvkasutamisest

Täna ehitatakse harva tarkvara süsteeme nullist, vaid pigem üritatakse korduvkasutada olemasolevaid lahendusi ja kohandada vastavalt uutele nõuetele. Selline meetod märkimisväärselt vähendab arendaja koormust, võimaldades aega ja ressursse kokku hoida. Siiamaani aga pole see tavapäraseks standardmeetodiks arendusprotsessis muutunud [3].

Tarkvara korduvkasutamine on uue süsteemi ehitamise protsess, kus võetakse kasutusele lahendused, mis olid ennem juba arendajate poolt loodud. Sellist metoodikat praktiseeriti alates tarkvara arenduse algusest. Korduvkasutamisest kui eraldi metoodikast mainiti esmakordselt aastal 1968 Doug Mcilroy poolt kirjutatud artiklis. Tema tegi ettepaneku käsitleda tarkvaratööstust korduvkasutatavate komponentide alusel [4].

Tarkvara korduvkasutamine on mõeldud eelkõige kulude ja aja säästmiseks ning toodete kvaliteedi parandamiseks. Korduvkasutades edukalt valmistatud toodet, sama kvaliteedi saab tagada ka teistes toodetes, rakendades eelnevalt omandatud kogemusi. Kuigi sellel on palju eeliseid, siiski paljudel ettevõtetel ei õnnestu tarkvara korduvkasutamise protsessi juurutada. Sageli on see põhjustatud selgelt määratletud strateegia, ressursside ja pädevate töötajate puudumisega [5].

Paljud tarkvarasüsteemid sisaldavad nullist ehitatud samalaadseid ja identseid elemente, mis paneb arendajaid mõttele, et üritada kõiki tarkvara vahendeid korduvkasutada. Viimased teadusuuringud näitavad, et korduvkasutada saab nii teeke, domeene, disaini, raamistikke, mustreid, arhitektuuri kui ka teisi analüüsi ja mõõdikutega seotuid vahendeid (Joonis 1) [6].



Joonis 1. Potentsiaalsed korduvkasutamise liigid [6].

2.2 Tarkvara korduvkasutamise eelised

Praeguste IT-toodete funktsionaalsus sõltub rohkem tarkvarast kui riistvarast, seega korduvkasutamise protsess on viimasel ajal tarkvara ettevõtetele erilist huvi pakkunud. Konkurentide surve sunnib ettevõtteid investeerima suuremaid summasid, et juurutada korduvkasutamise metoodikat oma kontsernis. See omakorda võimaldab luua veelgi keerulisemaid, usaldusväärsemaid ning odavamaid süsteeme lühikese aja jooksul [1].

Esialgsed investeeringud tarkvara korduvkasutamise metoodikasse aitavad edaspidi ettevõttel arendamis- ja hoolduskulusid kokku hoida, mis mõjub arenduse produktiivsusele positiivselt. Produktiivsust saavutatakse vähema koodi kirjutamise vajaduse tõttu, mis nõuab vähem testimist, analüüsimist, hinnanguid ja muid eeltöid [1], [3]. Näiteks HP *(Hewlett-Packard)* teatas, et nende tootlikkus on tõusnud 6%-lt 40%-ni tänu tarkvara korduvkasutamisele [1].

Samuti tuleb silmas pidada, et korduvkasutatavad komponendid tõenäoliselt omavad vähem defekte kui need, mis said hiljuti valmis ja olid rakendatud ainult üks kord [1]. Selline tulemus on tingitud asjaolust, et paljud arendajad jõudsid selle komponendi koodi mitu korda juba läbi käia (*debug*) ja vigu kõrvaldada. Sellisel juhul on kindlasti nõutud komponentide pidev hooldus ja haldamine [3]. Mida rohkem komponendi korduvkasutatakse, seda usaldusväärsem on see tarkvarasüsteemi kasutamiseks. Põhjalikult läbi testitud komponent tõstab võimalust koodivea avastamiseks, mistõttu suurendab ka töökindlust [1].

Korduvkasutamise protsessile investeeritud aeg on väärt sellele panustatud aega. Kuna korduvkasutatavat komponendi on võimalik kasutada samaaegselt mitmetes süsteemides korraga, siis selline investeering on majanduslikult õigustatum, juhul kui sama komponendi välja arendati ja kasutati ainult üks kord. Lisaks investeeritud aeg kompenseerib ennast toodete kiirema turule toomisega. Näiteks HP organisatsioon teatas, et neil õnnestus turule jõudmise aega kiirendada 3 korda [1].

2.3 Tarkvara korduvkasutamise takistused

Kuigi tarkvara korduvkasutamisel on palju eeliseid, ettevõtted ei kiirusta seda metoodikat enda kontsernis juurutada. Selle takistusteks on tehnilised, organisatsioonilised, majanduslikud või juhtimistegurid [7].

2.3.1 Organisatsioonilised takistused

Tarkvara korduvkasutamise protsessi taga seisab põhjalikult läbimõeldud organisatsioonistruktuur. Kõigepealt tuleb arvestada erinevate vajadustega, mis võivad suureulatusliku korduvkasutamise korral tekkida. Näiteks tuleb moodustada üht meeskonda, kus töötajad vastutavad komponentide arendamise, hooldamise ja sertifitseerimise eest [7]. Juhul kui ettevõttes on puudu pidevalt komponentide arendav meeskond, siis tarkvara korduvkasutamise protsess muutub raskemaks ning sellest on vähem kasu [8].

2.3.2 Tehnilised takistused

Tarkvara komponentide korduvkasutamiseks on tarvis leida efektiivseid meetodeid nende leiutamiseks ja taastamiseks. Hästi struktureeritud ja kõiki komponente sisaldav hoidla oleks ettevõttele abiks. Samuti korduvkasutamise varad peaksid olema hästi

dokumenteeritud, kujundatud ja rakendatud, et neid saaks vajadusel kohandada ja täiendada. See omakorda võib osutuda kallim kui komponente otse nullist arendada [7].

Tehnoloogiate ja raamistikute uuendamisega on oluliseks takistuseks saanud ka vananenud komponendid, kuna nende koodi on edaspidi raske töös rakendada. Siinkohal tuleb arvesse võtta kulud ja ressursid, mida läheb tarvis korduvkasutamise komponentide eksportimiseks vananenud süsteemist. Samuti on muudatuste kohandamise puhul väga raske leida täpselt sama lahendust, mis vastaks kaasaegsetele nõuetele [7].

2.3.3 Majanduslikud takistused

Korduvkasutamine on hea viis ressursside kokku hoidmiseks, kuid sellega kaasnevad ka tõsised kulud. Kõige kulukam on toodete välja arendamine korduvkasutatuna ning selle valmimisprotsessi määratlemine ja rakendamine. Selleks on tarvis investeerida raha töötajate välja koolitamise ja töökorralduse muutumise jaoks, lisaks ka tehnoloogiatesse ja metodoloogiasse. Antud olukorras on raske ennustada, kas korduvkasutatuna ette valmistatud komponente läheb tulevikus vaja. Seega investeeringud on õigustatud ainult juhul, kui komponente korduvkasutatakse ka edaspidi [7].

2.3.4 Juhtimistakistused

Kuigi tarkvara korduvkasutamise põhieesmärk seisneb kulude ning turule toomise aja vähendamises, tihti ei arvestata metoodika välja arendamisele kulutatud aja kui ka ressurssidega. Hulk kuludest, mida on tarvis enne varade arendamist ära katta on töötajate välja koolitamine, ettevõtte ümberkorraldamine ning tehnoloogiate tugi. Kuigi mitte kõik ettevõtted on nõus varases staadiumis suuri summasid investeerima [8].

Lisaks tulude ja kulude eelhinnangu puudus on üks põhjus vältida korduvkasutamist varasemal etapil. Esiteks korduvkasutatava vara arendamise kulud on otseselt seotud personali tehnilistest oskustest ja pädevusest. Tulud omakorda sõltuvad toodete valikust, mille jaoks läheb korduvkasutamise vara tarvis [8].

2.4 Tarkvara korduvkasutuse tüübid

Tänu sellele, et tänapäeval korduvkasutamine saab ettevõtete seas suure hoogu juurde, juhid püüavad veelgi rohkem tarkvara vahendeid uuesti kasutada. Korduvkasutamise hierarhia ei hõlma endasse ainult koodi, vaid sh ka ettevõtte arhitektuuri, mustrite, raamistiku, artifaktide, komponentide, mallide jm varade uuesti kasutamist. Tihti võib juhtuda, et üksikul komponendi korduvkasutamisel genereeritakse ka teisi varasid tänu omavahelistele seostele [3].

Arhitektuurilise korduvkasutamise mõiste taga on suur protsess, mis hõlmab endasse suuremahuliste varade tuvastamist, arendamist ja toetamist ettevõtte arhitektuuri kaudu. Ettevõtte arhitektuur defineerib korduvkasutatavaid domeeni mudeleid ning omavahel seotud domeeni või äriklasside kogumeid [6].

Mustrite korduvkasutamine on dokumenteeritud lahenduste kasutamine üldiste probleemide lahendamiseks. Eristatakse analüüsi-, kujundus- ja arhitektuurmustreid [6]. Tavaliselt sisaldavad mustrid probleemi kirjeldust, mida saab kasutada erinevates olukordades. Mustrite korduvkasutus võimaldab tarkvara arendamise protsessi kiirendada tänu sellele, et parandab koodi loetavust, millega välditakse võimalikke probleeme [9].

Raamistiku korduvkasutamine hõlmab klasside kogude kasutamist, mis koos rakendavad domeenide põhifunktsionaalsust [6]. Raamistiku komponentide korrektne toimimine nõuab ka teiste komponentide liikmete olemasolu ja omavahelist koostööd. Sellisel juhul korduvkasutamise arendajad saavad näiteks pärida rakenduse kujunduse, mis on loodud teiste arendajate poolt ning keskenduda ainult rakenduse funktsionaalsusele [3].

Projekti alustamisel on tihti tarvis arendamise käigus varem loodud artifakte, mis võivad olla UML (*Unified Modeling Language*) kasutusjuhud, klassidiagrammid, mudelid, nõuded ja juhised. Artifaktid võimaldavad kirjeldada rakenduse arhitektuuri, aga mõnikord kasutatakse, et genereerida uusi ideid järgmiste etappide jaoks [6]. Artifaktid võimaldavad arendajatel ja projektijuhtidel tulevikus süveneda projekti detailidesse. Olenemata projekti seisust, annavad artifaktid meeskonna liikmetele ja klientidele selget pilti projekti eesmärkidest ja selle loomise ning hooldamise probleemidest [10].

Mallide korduvkasutamine fokuseerib dokumentide kasutamisele, mis enamasti sisaldavad süsteemi kirjeldust [6]. Dokumentatsioon on tarkvara süsteemis väga oluline detail, kus tuleb eristada protsess- ja tootedokumentatsiooni. Tootedokumentatsioon kirjeldab kuidas süsteemi kasutada ja kuidas see on rakendatud, protsessidokumentatsioon aga iseloomustab protsessi loomist. Kuna iga komponent sisaldab oma dokumentatsiooni, siis korduvkasutamise käigus tuleb arvestada mõlemat [3].

Lähtekood on praegu kõige levinum korduvkasutamise vahend, mida kasutatakse taas ühe või mitme rakenduse lõikes. Tavaliselt koodi korduvkasutamine tähendab klasside või funktsioonide omavahelist jagamist. Samuti saab koodi lihtsalt kopeerida ja oma rakendusse viia, aga see võib tuua endaga kaasa potentsiaalselt tõsiseid tagajärgi [6]. Funktsioonid on lähtekoodi populaarseim korduvkasutamise vorm. On olemas hulk erinevaid funktsioonide teeke, alates standardsetest kuni domeeni-spetsiifilisteni. Klasside teegid on omakorda objektorienteeritud funktsiooniteekide vorm. Nende eelis funktsioonide ees on parem modifitseerimine ja kohanemisvõime [3].

2.5 Komponendipõhine tarkvara korduvkasutamine

Komponendipõhine tarkvara arendus (*component-based software development*, CBSD) on tänapäeval kõige efektiivsem ja kiirem meetod keeruliste süsteemide ehitamiseks [11]. Süsteem kui tervik on korduvkasutamiseks palju eelistatum, kui see on struktureeritud üksteisest sõltumata komponentidest. Esiteks, neid komponente on väga lihtne arendajate vahel jagada ja uude süsteemi integreerida, eriti kui elementide seosed on hästi dokumenteeritud ja on viidud miinimumini. Teiseks, ekspluateerimine on palju lihtsam, kui muudatusi saab lokaalselt teha ilma kogu süsteemi rikkumiseta [3]. Seega komponentide alusel süsteeme on palju kergem ja odavam kokku panna kui neid, mis on tehtud eraldi osadest [11].

CBSD korduvkasutamisel fokuseeritakse eelkõige komponentidele. Tarkvarakomponent on selge funktsionaalsusega ja täpselt määratletud liidesega element, mida on võimalik teistes tarkvarasüsteemides korduvkasutada [12]. Komponenti saab vaadelda nagu konkreetseid funktsioone täitvat süsteemi, mis võib koosneda ka teistest komponentidest [13]. Tarkvara komponendid võivad olla klassid, teegid, moodulid, funktsioonid, laiendused või plaginad [14].

CBSD põhineb nii ettevõtte siseselt ehitatud kui ka turult hangitud tarkvara komponentide kohandamisel. Lisaks CBSD metoodika fokuseerib nullist tehtud korduvkasutatavate komponentide välja arendamisele, mida saab kohe erinevates tarkvarasüsteemides kasutada [15].

2.5.1 Olemasolevad tarkvara komponendid

Olemasolevaid komponente tavaliselt jaotatakse kaheks tüübiks: komponendi, mida saab muuta ehk valge-kasti korduvkasutus (*white-box reuse*) ja ilma muudatusteta musta-kasti korduvkasutus (*black-box reuse*) [16].

Valge-kasti korduvkasutus on lähenemisviis, kui koodi muutmine on lubatud. Arendaja saab muuta koodi vastavalt oma vajadustele ning kohandada olemasolevaid komponente uue projekti nõuetele. Koodi muutmine nõuab põhjaliku detailide tundmist, mis on võimalik ainult juhul, kui kood on välja arendatud ettevõtte sees või sellega kaasneb suurepärane dokumentatsioon [16]. Muudatustega komponendipõhine korduvkasutus võib omakorda jagada kaheks tüübiks: ettevõtte olemasolevate komponentide (mis esineb 60-80% juhtumites) ja avatud lähtekoodiga ehk vabavara korduvkasutamine [17]. Vabavara on avatud lähtekoodiga tarkvara, mis on kättesaadav laiema ringi kasutajate jaoks tasuta kasutamiseks. Seda saab alati ilma piiranguteta oma süsteemides rakendada, vastavalt oma soovidele muuta ja jagada arendajate vahel. Vabavara põhieeliseks on turvalisuse tagamine. Kuna paljud kasutavad seda lahendust pidevalt, siis arendajad leiavad üles uusi vigu, parandavad neid ja jagavad uuendatud versioone teistega. Sellisel juhul on lahendus stabiilselt pidevas hoolduses tänu suure arendajate ühingule. Kõige suuremad vabavara hoidlad on *Sonatype* ja *SourceForge* [18].

Musta-kasti korduvkasutamine eeldab tasuliste komponentide eksportimist uude süsteemi ilma mingite muudatusteta. Selliseid komponente nimetatakse kommertsi komponentideks (*commerical off-the-shelf, COTS*), mis on saadaval Interneti turul korduvkasutamise eesmärgiks [17]. Arendaja peab süvenema ainult komponendi funktsionaalsusse, ilma selle sisemist loogika ja koodi muutmata. Tavaliste COTS toodetega tulevad kaasa ainult dokumentatsioon ja juhendid. Selline lähenemisviis võib vähendada kulusid ning vältida raskusi komponendi muutmisel. Juhul kui tulevikus tuleb välja, et arendajal ei lähe kommertsi toodet enam vaja või ta soovib seda modifitseerida, siis komponent on selle koha pealt raisatud kulu, mille tõttu musta-kasti korduvkasutamine on tänapäeval vähem kasutatav [16].

2.5.2 Korduvkasutatavate tarkvara komponentide arendamine

Korduvkasutatavate komponentide välja arendamine on võrreldes olemasolevate komponentide kohandamisega palju kulukam protsess. Komponendid peaksid olema programmeeritud üldistataval kujul, et neid saaks erinevates süsteemides hõlpsasti kasutada [19]. Praktika näitab, et selline strateegia on üsna riskantne, kuna tihti on raske ennustada, kas sellist korduvkasutatavat komponenti läheb teine kord arendusprotsessi käigus vaja. Kui tuleb välja, et arendajatele selline komponent ei sobi, siis seda tuleb ümber teha, mis on lisakulu [6].

2.5.3 Tarkvara komponentide hoidla

Korduvkasutatavate komponentide mahu kasvamisega on vaja hoidlat, kus saaks neid ülal pidada ja vajadusel otsida või taastada [20]. Komponentide hoidla on andmebaas, mis hoiab endas vajalikku infot komponentide kohta nagu kasutamise ajalugu, koostoime teiste komponentidega, klassifikatsioon ja dokumentatsioon. Hoidla on arendajatele vajalik eelkõige selleks, et olla kursis kõikide ettevõtte komponentide olemasolu ja nende andmetega [3]. Hoidla efektiivseks toimimiseks on vaja otsingumehhanismi, mis vastaks üles laetud komponentide nõuetele [21]. Otsingupäring peaks toimima nii tavalise teksti sisemisel kui ka märksõnade järgi. Päringu tulemustele võib vastata mitu komponente, aga valitakse kõige sobivamat. CBSD protsessi käigus hoidla haldamine ja komponentide hoidlast kätte saamine on ettevõtete jaoks kõige suurim väljakutse [22].

3 Metoodika

ADM Interactive OÜ probleemi parima mõistmiseks oli autori poolt teostatud äriprotsesside analüüs. Kõigepealt analüüsiti osakondade sisemist ja vahelist mõõdet, et välja selgitada tarkvara korduvkasutamise hetke olukorda ja projektides kasutatavaid tehnoloogiad. Lisaks kaardistab autor olemasoleva AS-IS arenduse elutsükli, mis on aluseks TO-BE mudeli modelleerimiseks.

3.1 ADM Interactive OÜ tutvustus

ADM Interactive OÜ on Eesti kõige suurim digi- ja e-turunduslahenduste pakkuv agentuur. Teenuste hulka kuuluvad äristrateegiad, UX/UI *(User Experience/User Interface)* disain, arendus, hooldus, pilvemajutus, sisuturundus ning analüütika. ADM'il on eraldi kolm suuremat arendusüksust, milliseid sisemiselt kutsutakse HP, MRM ja Fenomen osakondadeks. Autor viis analüüsi läbi osakondade vahel, et välja selgitada nende back-end ja front-end kasutatavaid tehnoloogiad.

HP osakond on Hewlet-Packard'i arenduspartner, kes teostab erinevaid veebiarendustöid. HP back-end osa on rakendatud Custom CMS *(Content Management System)* peal, mis on ehitatud kasutades .NET ja PHP *(Hypertext Preprocessor)* ja front-end'i jaoks kasutatakse React'i, mõned lahendused on tehtud ka jQuery peal.

MRM osakond loob veebilahendusi nii Eesti kui ka välismaa turule, sh pakkudes e-kommerts teenuseid. MRM'i 90% back-end teenuste osast põhineb WordPress sisuhaldustarkvaral ja Laravel'il, front-end lahendusi arendatakse välja kasutades React, jQuery või Vue.js tehnoloogiaid.

Fenomen osakond on 18-aastase kogemusega arendusosakond, mille back-end osa põhineb Drupal sisuhaldussüsteemil ja Laravel raamistikul ning front-end lahenduste ehitamiseks kasutatakse React'i või jQuery teeke.

3.1.1 Osakondade sisene ja vaheline mõõde

Autor küsitles kõikide arendusüksuste (HP, MRM, Fenomen) arendajaid, et välja selgitada osakondade tarkvara korduvkasutamise seisu.

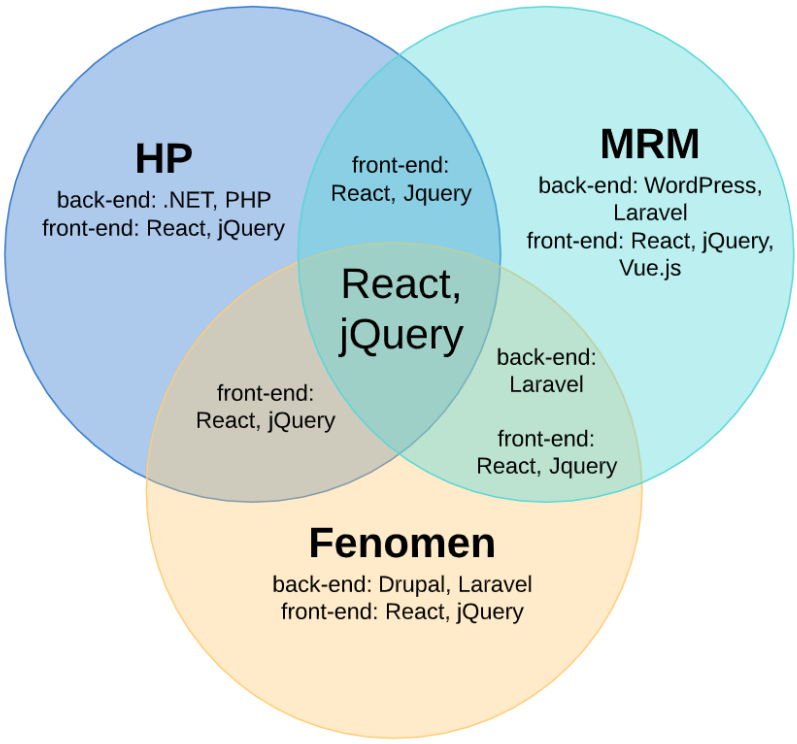
HP osakonnas otseselt tarkvara korduvkasutust väga palju ei esine. Tavaliselt HP osakonna liikmed arendavad paar suuremat projekti, mis on üksteisest päris erinevad. Projekti siseselt ehk erinevates vaadetes kasutatakse vahepeal sama koodi. Põhiprobleem on seotud hoidla puudumisega, kuhu oleks võimalik valmis tehtud koodi teegid või komponendid paigutada ja vajadusel sobiva lahenduse teises projektis kasutada. See tähendab, et tänapäeval ei ole töötajatel alati piisavalt infot, et kas on sarnast lahendust juba valmis arendatud ja kus see asub. Hetkel on lihtsam võtta aluseks eelnevast projektist mingi komponent ning hakata selle baasil täitsa uut nullist arendama.

MRM'i osakond üritab koodi korduvkasutada seal, kus on see võimalik. Tavaliselt see juhtub Laravel'i komponentide või WordPress'i plaginatega. Arendajad võtavad varem tehtud lahenduse ja kohandavad selle uutele nõuetele. Praegu aga teostavad seda ainult kogenud vanemarendajad. Seega tehakse ühe kliendi jaoks lahendust algusest peale eeldusega, et seda läheb ka kunagi mujal vaja, mis nõuaks minimaalseid muudatusi. Viimasel ajal seda on eriti tihti hakanud tegema WordPress'i plaginatega, kuna neid läheb tihti vaja. Täielikult terviklikke lahendusi, mida saaks mitmes veebilehes korraga kasutama hakata, tänapäeval ei arendata. Selleks, et komponendi universaalseks teha, on vaja rohkem arendusaega ja ressursse, mida MRM'i osakond ei suuda tänapäeval teostada. Selle takistuseks on ka protsessi puudus, mis ei ole täna mingil määral defineeritud ja organiseeritud.

Fenomen'i osakonnas on tarkvara korduvkasutamist keeruline teha, kuna igal veebilehel on üsna erinev funktsionaalsus ning on raske ühisosa leida. Drupal'i kontekstis näiteks on võimalik sisukomponente korduvkasutada mitmetes projektides korraga, aga üsna tihti on seal muud nõuded lisaks veel juures. Praegu on võimalik Fenomen'is Drupal'i lahendusi korduvkasutada ainult lihtsamal kujul. Näiteks, kui suurem osa arendusest on tehtud, siis tuleb kasutada igat koodirida, mida saab *copy-paste* (kopeerimine-kleebimine) meetodiga ühest projektist teise üle kanda. Samuti on abiks ka Drupal'i ametlik koduleht, kus vahepeal Fenomen'i arendajad kasutavad avatud lähtekoodiga lahendusi ning panustavad nende koodi kvaliteeti. Näiteks kui koodi sees leitakse vea üles, siis arendajad püüavad ise seda parandada ning selle kogukonnale tagasi anda. Vahest luuakse ka ise

Drupal'i mooduleid, mida saab sama efektiivselt teiste kasutajatega jagada. Kommertstarkvara üldiselt ei kasutata, v.a kui tõesti vaja on.

Osakondade vahelise mõõdu analüüsimiseks oli tehtud Venn diagramm (Joonis 2), mis aitab välja selgitada, millised arendustehnoloogiatest on osakondadel ühised. See omakorda annab selget pilti, milliseid back-end ja front-end tarkvara raamistiku osasid on mõistlik korduvkasutada.



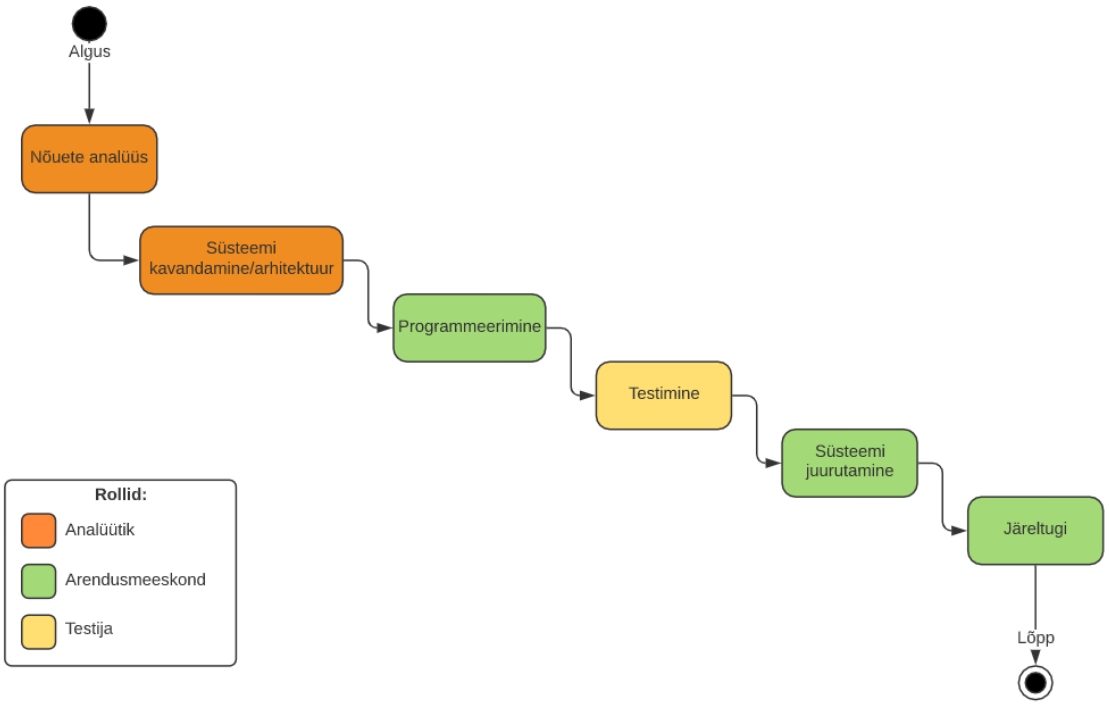
Joonis 2. Fenomen, MRM ja HP osakondade vaheline mõõde Venn diagrammi kujul.

Analüüsides ADM-i arendusüksuste tausta tuleb välja, et Fenomen ja MRM osakonnad saavad omavahel back-end lahendusi korduvkasutada, mis on rakendatud Laravel raamistikul. React ja jQuery Javascript'i teegi komponente saaks korraga jagada ja korduvkasutada nii Fenomen, MRM ja HP meeskondade vahel.

3.1.2 ADM'i süsteemi arenduse elutsükkel (AS-IS)

ADM'i arendusüksuste protsessi elutsüklid on igas osakonnas erinevad, kuna see sõltub projekti iseloomust ja arendajate kogemustest. Tänapäeval üldine süsteemiarenduse protsess mudel (Joonis 3) kujutab endast klassikalist kosemudeli (*waterfall*), mida võib laiendada ka agiilse arendusprotsessi ja SCRUM metoodikasse. Protsess koosneb nõuete

analüüsimisest, süsteemi kavandamisest, programmeerimisest, testimisest, süsteemi juurutamisest ja järeltoest.



Joonis 3. ADM'i üldine tarkvaraarenduse mudel

Tavaliselt iga uus kosemudeli tüüpi projekt algab nõuete välja selgitamisest, mida saadab klient ise. Nõuete abil saab aru, mida klient ootab tuleviku süsteemist ja mis eesmärke see peaks saavutama. Selles faasis analüütik selgitab välja nii funktsionaalseid kui ka mittefunktsionaalseid nõudeid.

Süsteemi kavandamise ehk disaini etapis koostab analüütik detailsema funktsionaalsete nõuetega dokumendi, mis on aluseks arendustööde realiseerimiseks. Dokument kirjeldab täpsemalt süsteemi arhitektuuri, mis koosneb eraldi jaotatud komponentidest.

Programmeerimise käigus ehitab arendusmeeskond loodud spetsifikatsiooni alusel tarkvarasüsteemi. Arendajate kohustuste hulka lisanduvad vajalike keskkondade paigaldamine. Kõik arendustööd toimuvad arenduskeskkonnas. Arendusmeeskond koosneb tavaliselt front-end ja back-end arendajatest.

Kui tarkvaraline arhitektuur on loodud, tuleb tarkvaratestijal arenduskeskkonnast testimiskeskkonda tarnitud funktsionaalsuse läbi käia ja vigu otsida. Selle tulemusena

koostakse testimisanalüüsi raporti, mis kirjeldab kõiki tekkinuid süsteemirikkeid. Lisaks peab testija veenduma, et süsteem vastab püstitatud nõuetele.

Peale testimist tõstetakse kõik muudatused testimiskeskkonnast kliendi toodangukeskkonda. Selles faasis tihti tellib klient ka kasutajate välja koolitamist uue süsteemiga töötamiseks.

Kui valmis tehtud tarkvara on kliendile üle antud, tuleb seda ka edaspidi hooldada. Värskelt valminud süsteemis alati tekivad vead, mida tuleb kõrvaldada. Tihtipeale tellib klient ka lisatöid, mille rakendamine nõuab eelnevate süsteemiarenduse etappide kordamist.

3.2 Tööriistad

AS-IS ja TO-BE äriprotsesside kaardistamiseks kasutab autor UML diagramme, mille joonestamiseks on kasutatud Lucidchart veebipõhine programm. Mudeli arendusprotsessid on modelleeritud tegevusdigrammi (*Activity Diagram*) põhjal. Diagramm sisaldab AS-IS ja TO-BE arenduse elutsükli etapid algusest lõppuni.

TO-BE meetodikat toetava keskkonnaks oli teostatud komponentide hoidla tarkvara nõuete analüüs ning tehtud andmemudel. Analüüs sisaldab funktsionaalseid ja mittefunktsionaalseid nõudeid, mis kirjeldavad hoidla põhiülesandeid ja eesmärke. Andmemudel on tehtud hoidla objektide seostuste ja nende atribuutide kirjeldamiseks. Lisaks on toetava keskkonna jaoks kaardistatud meeskonna kasutusjuhtude mudel, mis annab ülevaade TO-BE protsessis osalevate rollide suhtlusest ja nende kasutusjuhtudest.

4 Tulemused

4.1 Eesti IT-firmade küsitlus

Tuginedes ADM'i korduvkasutuse metoodika puudumisele oli autori poolt otsustatud läbi viia küsitluse Eesti suurte IT-ettevõtete vahel. Käesoleva küsitluse eesmärgiks on teada saada ettevõtete korduvkasutamise AS-IS olukorrast. Saadud tulemused aitavad välja selgitada, kui palju on tarkvara korduvkasutamist ettevõtetes juurutatud. Vastavalt sellele, autor saab selget pilti, kas ADM'i jaoks välja töötatud TO-BE lahendus on õigustatud, et olla tulevikus IT-turul veelgi konkurendivõimelisem.

4.1.1 Küsitluse kirjeldus

Küsitluse läbiviimiseks oli vaja saada loetelu kõikidest Eesti suurtest tarkvarafirmadest. Vastavalt soovitud tingimustele oli koostatud päring Sihtgrupid infoportaalis, mis jagas andmeid Eesti IT-firmadest. Päringu põhitingimuseks oli firma suurus, mille töötajate arv peaks olema vähemalt 21 ning aastakäive üle 100 000 eurot. Peamisteks ettevõtete tegevusaladeks olid valitud telekommunikatsioon, programmeerimine ja infoalane tegevus. Päringu tulemuseks oli saadud 144 ettevõtet Harjumaa ja Tartumaa maakondadest.

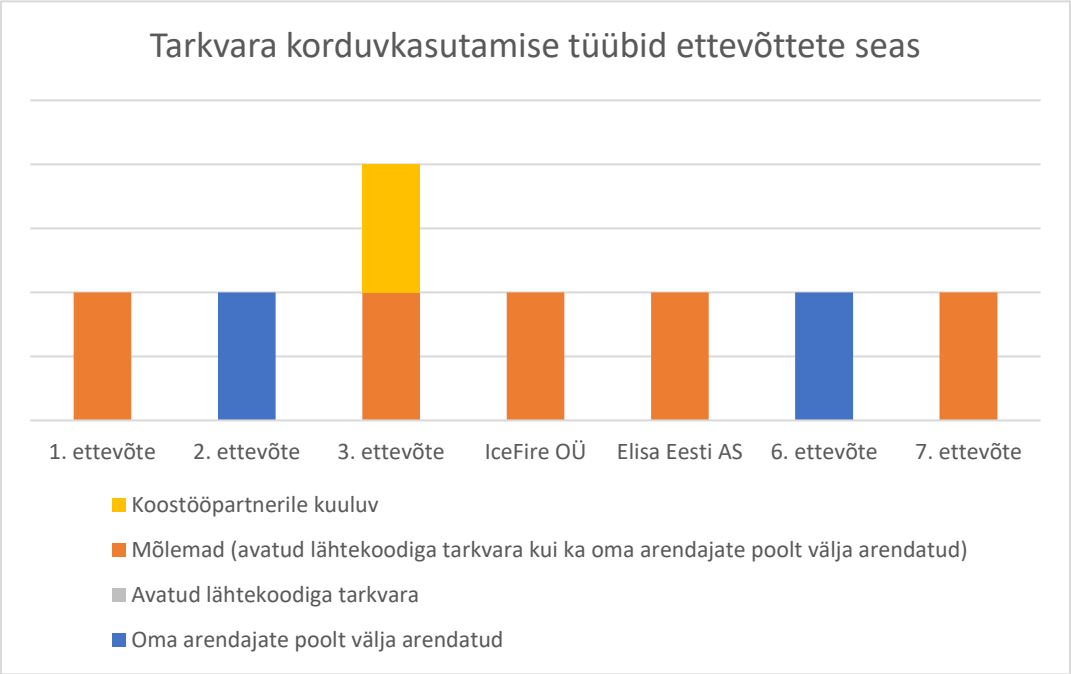
Käesolev küsimustik oli koostatud kasutades Google'i Vormid online tööriista. Firmadele oli pakutud 20 küsimust ja nendele vastamine võttis ligikaudselt 5-7 minutit aega. Küsimustik oli laiali saadetud 144 IT-firmadele, millest kokku vastajaid oli 7. Viis ettevõtet soovisid lõputöö kontekstis jääda anonüümseks ja 2 ettevõtet (Icefire OÜ ja Elisa Eesti AS) andsid loa kasutada nende ettevõtte nimesid tulemuste analüüsi eesmärgil. Enamik ettevõtete põhi tegevusvaldkondadeks on telekommunikatsioon ja programmeerimine ning nende arendusmeeskondade keskmine kogemus on 5-10 aastat. Peamised tooted, mida küsimustikus osalenud firmad arendavad on äritarkvara, veebi- ja mobiilirakendused ning elektroonikaseadmete püsivara.

4.1.2 Küsitluse tulemused

1. *Kas projekti alustamisel kaalute tarkvara korduvkasutamist?*

Kõik 7 küsitluses osalenud firmadest vastasid, et korduvkasutavad tarkvara oma arendusprotsessides. Antud küsimus oli üks tähtsamaid, kuna see andis ülevaate tarkvara korduvkasutamise hetkeolukorrast suurte IT-firmade vahel.

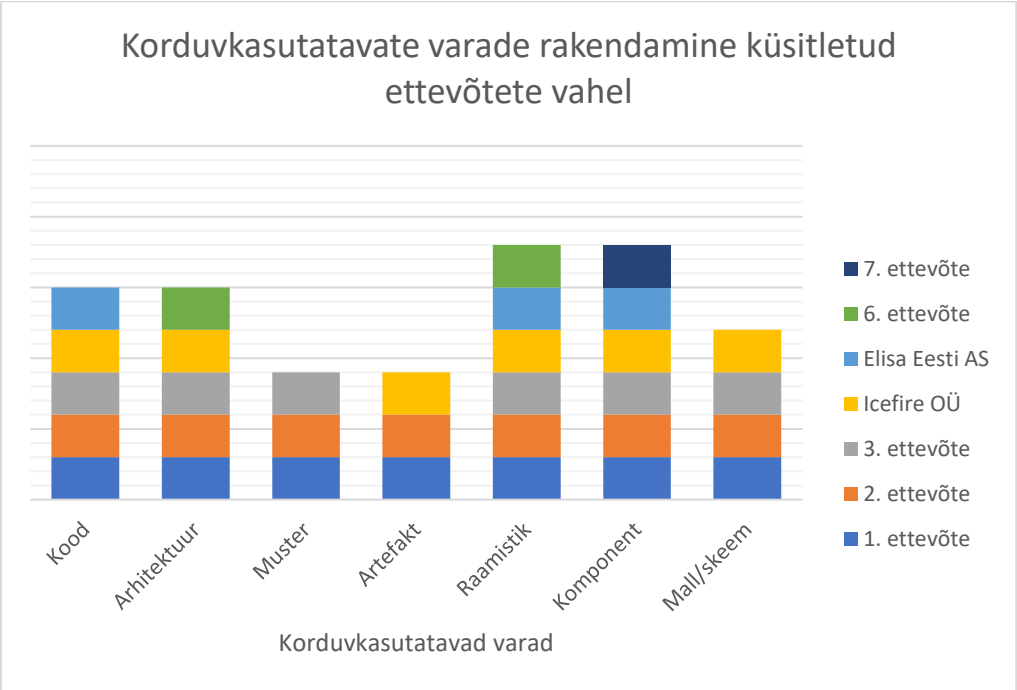
2. *Millist tüüpi tarkvara korduvkasutate?*



Joonis 4. Tarkvara korduvkasutuse tüübid.

Antud küsimuse puhul oli eesmärgiks välja selgitada, kas ettevõtted korduvkasutavad olemasolevaid varasid või pigem eelistavad hankida välistest allikatest vabavara ja kommertslahendusi. 2 ettevõtet kasutavad oma valmis tehtud lahendusi, siis kui 5 ettevõtet eelistavad rakendada nii oma kui ka kolmanda osapoole tarkvara (Joonis 4). Juriidilisest aspektist lähtuvalt tähendab see, et tänapäeval on võimalik sama efektiivselt kasutada nii väliste allikate varasid kui ka oma lahendusi, ilma autori õiguseid rikkumata.

3. *Milliseid korduvkasutatavaid vahendeid Teie asutuses kasutatakse taas?*

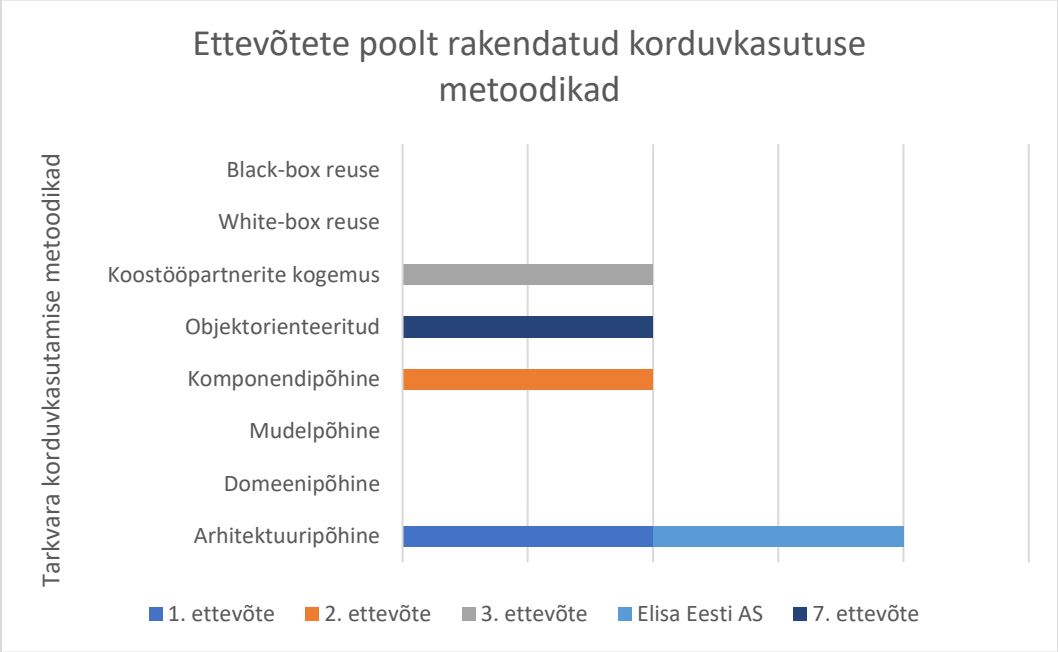


Joonis 5. Korduvkasutatavate varade rakendamine küsitletud ettevõtete vahel.

Vastajatele oli pakutud valida kõigi enam kasutatavate varade seast 7 varianti, mida nad endal korduvkasutavad. Need on lähtekood, arhitektuur, muster, artifakt, raamistik komponent ja mall või skeem.

Kõige rohkem korduvkasutatakse raamistikke ja komponente, mida mõlemat valisid 6 erinevat ettevõtet (Joonis 5). Seejärel 5 erinevat firmat on valinud, et korduvkasutavad koodi ning arhitektuuri. Kõige vähem ettevõtted korduvkasutatavad mustreid ja artifakte. Antud küsimuse tulemusest selgub, et kõik küsitletud ettevõtted rakendavad vähemalt 3 korduvkasutatavat vara. Antud tulemusele on kirjanduse ülevaates toodud välja ka kinnitus, et komponent on täna kõige lihtsaim ja populaarseim arendajate seas vara, mida kõige enam korduvkasutatakse.

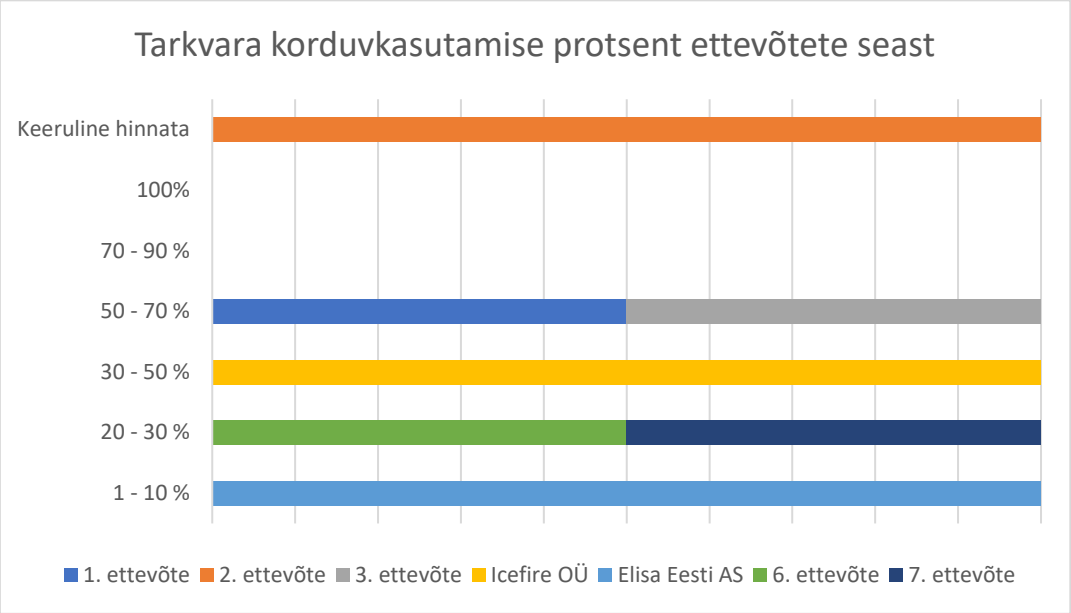
4. *Kas Teil on kindel metoodika/tehnika korduvkasutamise võimaldamiseks? Kui "Jah", palun valige milline metoodika on Teie ettevõttes kasutusel?*



Joonis 6. Ettevõtete poolt rakendatud korduvkasutuse metoodikad.

7-st küsitletud ettevõttest 5 vastasid, et neil on kindel metoodika tarkvara korduvkasutamiseks olemas. Joonisel 6 on toodud välja viie ettevõtte poolt kasutatavad metoodikad. Küsimuse eesmärk oli välja selgitada, mitu ettevõtet kasutavad kindlat metoodikat korduvkasutamise käigus ja millist tüüpi. Kaks ettevõtet valisid, et kasutavad arhitektuuripõhist metoodikat, üks kasutab objektorienteeritud tehnikat, veel üks komponendipõhist ja üks lisas, et metoodika valik baseerub meeskonnaliikmete ja koostööpartnerite kogemusel. 4. küsimuse tulemuste põhjal saab väita, et kuigi tarkvara korduvkasutuse metoodikaid on palju, Eesti IT-firmad kasutavad ainult osa sellest ning kõige rohkem kasutatakse arhitektuuripõhist metoodikat.

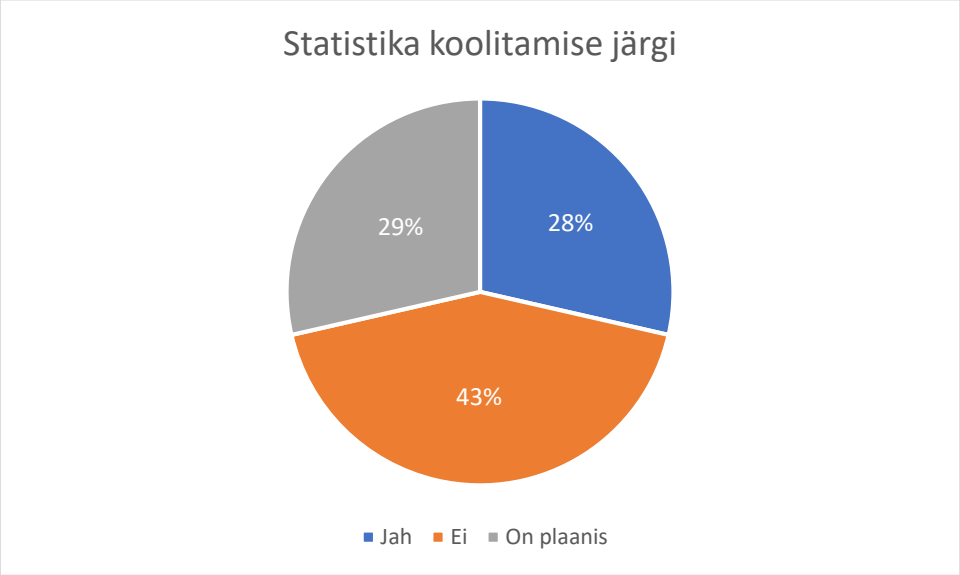
5. *Milline on Teie toodete korduvkasutamise protsent?*



Joonis 7. Tarkvara korduvkasutamise protsent küsitletud ettevõtete seast.

Antud diagrammi näitel (Joonis 7) on näha, milline on ettevõtete toodete korduvkasutamise protsent, mis varieerub 0-100 %-ni. 2 ettevõtet valisid, et nende tooted koosnevad 50-70% korduvkasutatavatest elementidest ning sama palju ettevõtet valisid, et 20-30%. Peaaegu pool Icefire OÜ toodetest (30-50%) koosneb korduvkasutatavetest osadest. Elisa Eesti AS andmetel, antud ettevõte peaaegu üldse ei korduvkasuta. Saadud tulemuste järgi võib teha järelduse, et firmad otsustavad tarkvara korduvkasutada projekti alustamisel päris tihti.

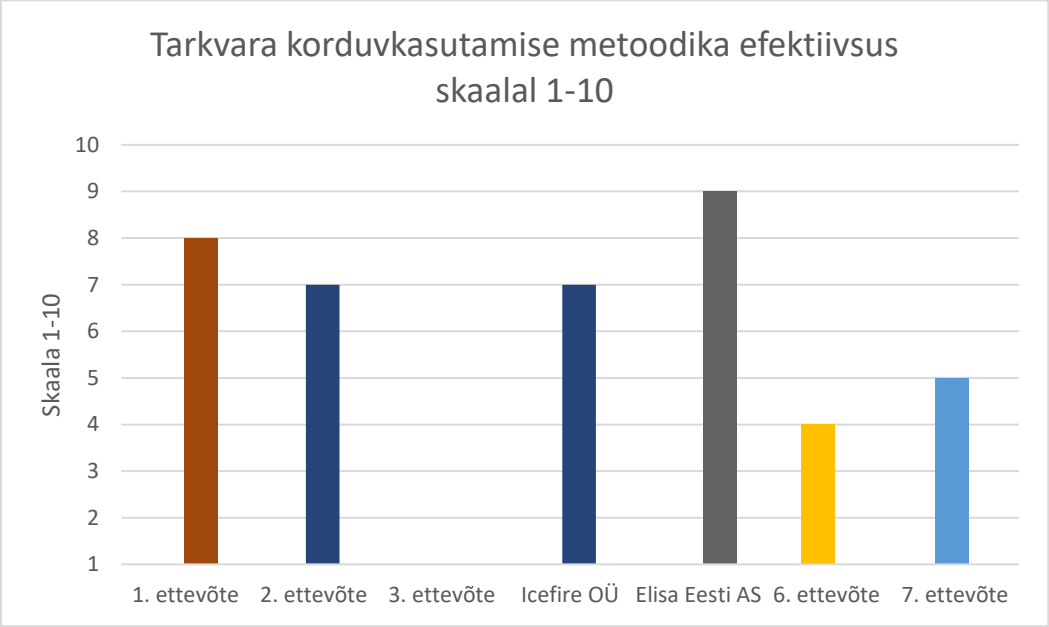
6. *Kas Teie asutuses koolitatakse töötajaid vastavalt korduvkasutamise metoodikale?*



Joonis 8. Koolitamise statistika.

Antud küsimus oli esitatud eesmärgiga, et uurida välja, kas küsitletud firmades on võimalus oma arendajaid vastavalt korduvkasutuse metoodikale välja koolitada. 3 ettevõtet vastas, et ei praktiseerita endal koolitamist (Joonis 8). Ülejäänud 2 firmat, millest üks oli Icefire OÜ, vastasid, et neil on plaanis sellist meetodid juurutada ning veel 2 ettevõtet, millest üks oli Elisa Eesti AS, tegelevad juba sellega tänapäeval. Kokkuvõttes antud küsimuse järgi saab öelda, et kuigi suurem osa ettevõttest praegu veel koolitamist ei praktiseeri, osad näevad selle juurutamise perspektiivi juba praegu.

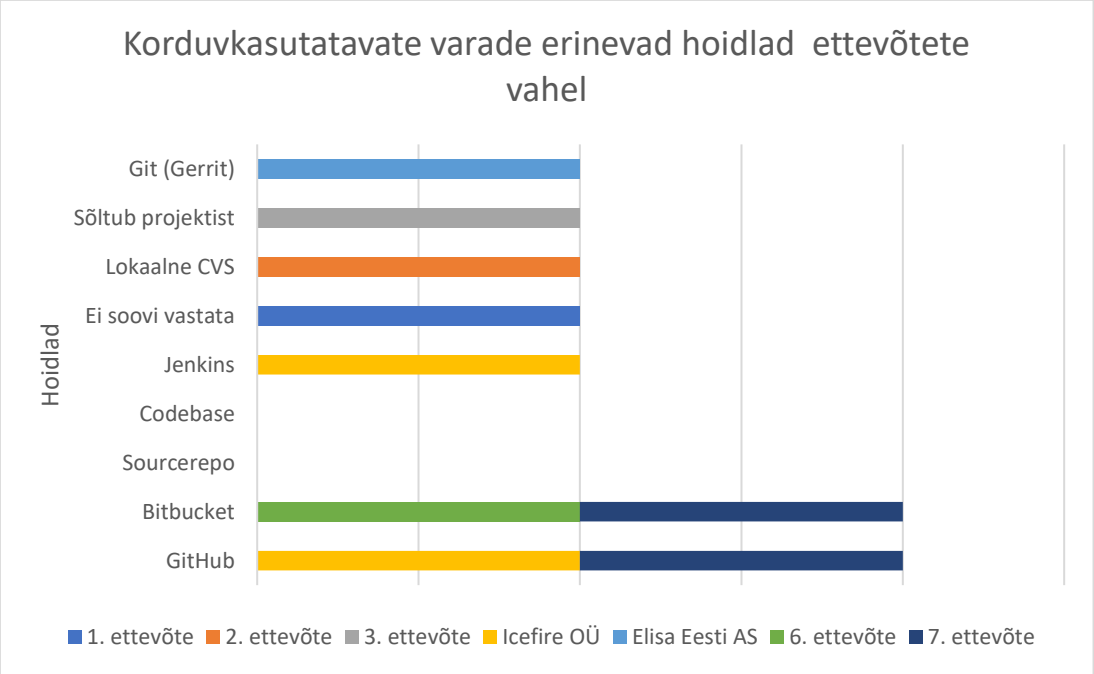
7. *Palun hinnake skaalal 1-10 korduvkasutamise metoodika efektiivsust?*



Joonis 9. Korduvkasutamise efektiivsus skaalal 1-10.

Antud küsimusega tahtis autor saada teada, kui hästi on tarkvara korduvkasutamise metoodika mõjunud ettevõtete tavapärastele arendusprotsessidele. Vastused varieeruvad skaalal 4-st kuni 9-ni efektiivsuse järgi (Joonis 9). Neli ettevõtet märkisid, et nende korduvkasutamise metoodika on tõstnud nende arendusprotsesside efektiivust. Ainult kaks ettevõtet hindasid oma efektiivsust keskmisest madalam. See annab usku, et firmade tootlikus on kasvanud tänu sellele, et kaob vajadus kõik tooteid otse nullist arendada.

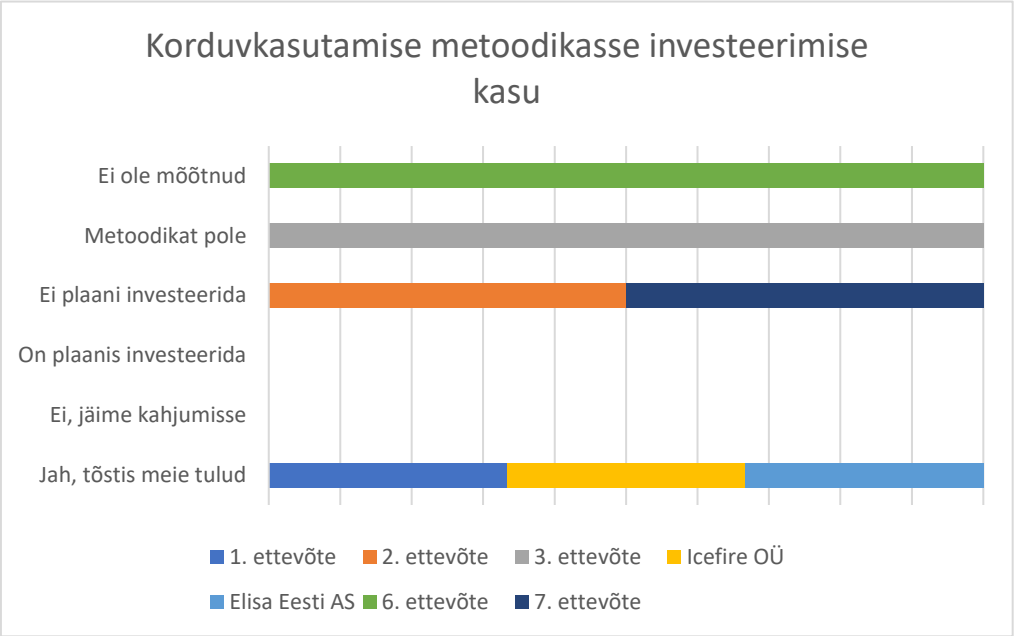
8. *Kas Teie asutuses on kindel koht korduvkasutatavate varade hoidmiseks? Kui Jah, palun valige vastust.*



Joonis 10. Ettevõtete hoidlad.

Enamikul IT-firmadel on oma hoidla, kus nad hoiavad alles oma tehtud lahendusi. Joonisel 10 on toodud välja diagramm, mis kirjeldab küsitletud IT-asutustes korduvkasutatavate varade kasutatavaid hoidlaid. GitHub'i ning Bitbucket'i kasutavad kokku 3 firmat, kus üks ettevõte kasutab mõlemaid (Joonis 10). Ülejäänud üksikud firmad kasutavad Jenkins, lokaalset CVS *(Concurrent Versions System)* süsteemi ja Gerrit'i. Järelduseks võib tuua välja, et enamik firmasid kasutavad kolmanda osapoolte lahendusi ning enda hoidlasse investeerimist tänapäeval pole vajadust.

9. *Kas investeering korduvkasutamise metoodikasse on majanduslikult õigustatud?*



Joonis 11. Korduvkasutamisse investeerimise statistika.

Korduvkasutamise metoodika nõuab varasemal etapil väga palju ressursse selle juurutamiseks. Antud küsimuse eesmärk oli välja selgitada, kas esmased investeeringud on tulevikus õigustatud või mitte (Joonis 11). 3 ettevõtet vastasid, et korduvkasutamise meetodikasse investeerimine tõstis nende tulud. 2 ettevõtet lähitulevikus ei plaani metoodikasse investeerida.

4.2 ADM'i tarkvara komponentide korduvkasutuse metoodika TO-BE lahendus

4.2.1 TO-BE lahenduse metoodika valimine

Enne kui tarkvara korduvkasutamist ADM'i sees juurutada, tuleb läbi mõelda millised varasid on selle jaoks vaja ja kust kohast neid hankida. Analüüsides ADM'i osakondade vahelist ja sisemist mõõdet tuli välja, et kõige sobivam korduvkasutamiseks vara on komponent. Põhjuseks on ühised front-end kasutatavad tööriistad nagu React ja jQuery JavaScript'i teegid. Teekide arhitektuur võimaldab tooteid arendada, kasutades konkreetseid komponente. Lisaks ka Wordpress ja Drupal sisuhaldussüsteemid, mis on

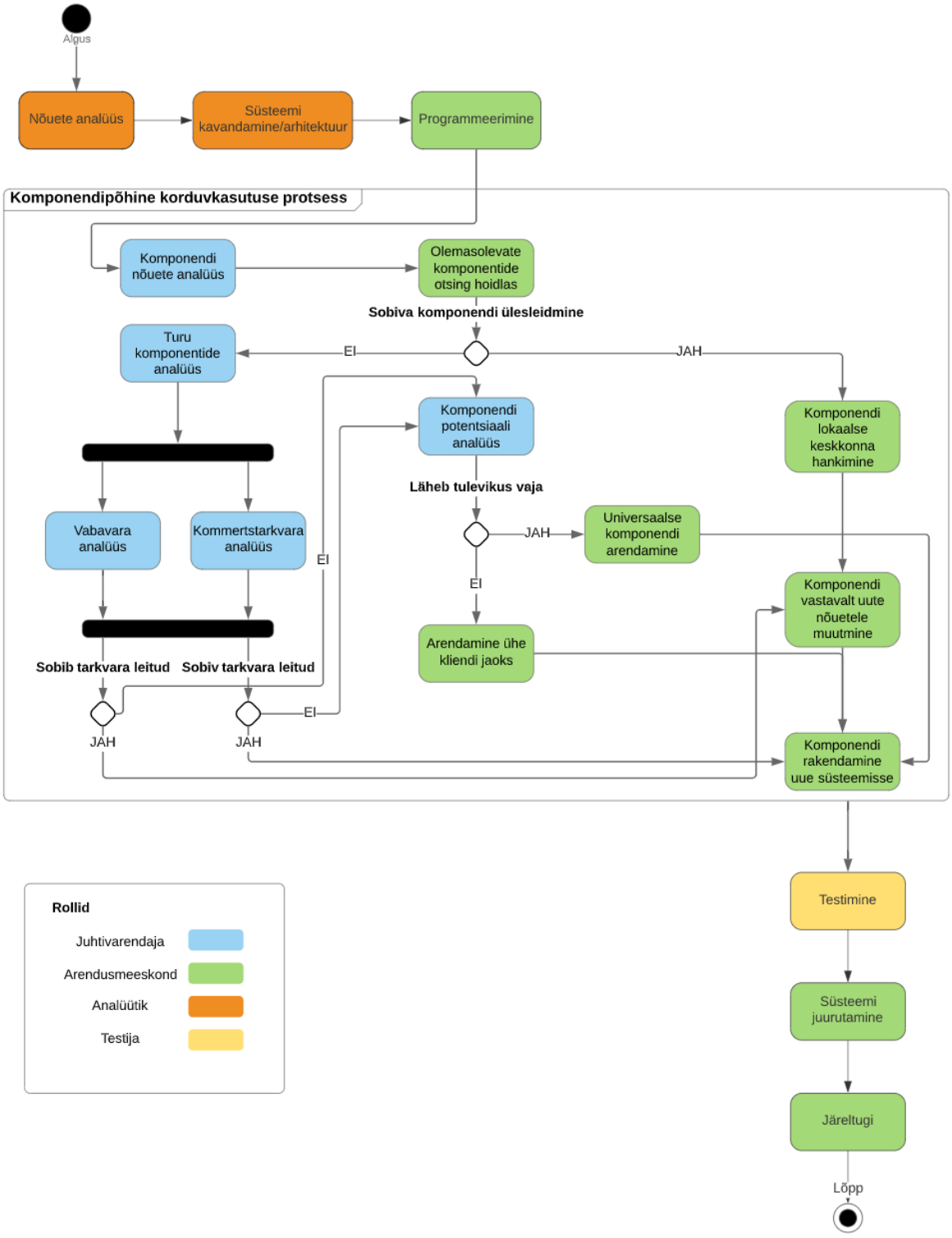
kasutusel back-end lahendusena Fenomen'is ja MRM'is, suuremas osas koosnevad ka erinevatest komponentidest ja moodulitest.

Lähtudes kirjanduse ülevaatest ja ADM'i AS-IS korduvkasutamise seisust on mitu võimalust edukaks komponentide korduvkasutamiseks. Esiteks saab ettevõte sees rakendada välja arendatud varasid, mis on juba ammu oma arendajate vahel kasutusel. Teine võimalus on komponente hankida otse välistest allikatest. Siit tuleb eristada avatud lähtekoodiga varasid ja kommertslahendusi. Lisaks saab komponente arendada välja ise, et nad oleksid juba alguses korduvkasutatavad. Selline strateegia aga on palju keerulisem ja ajakulukam, kuna nõuab lisainvesteeringuid arendusmeeskonna loomiseks.

4.2.2 TO-BE lahenduse kirjeldus

TO-BE mudeli eesmärk on lihtsustada hetkel olemasolevat programmeerimisetappi tänu komponendipõhise korduvkasutamise alamprotsessile. Käesoleva protsessi käigus lisandub ka juhtivarendaja roll, kes vastutab turu komponentide analüüsimise, tarkvara nõuete välja selgitamise ja korduvkasutatava komponendi potentsiaali hindamise eest. Arendusmeeskonnal tuleb nüüd kõigepealt oma hoidlas või turu olemasolevaid lahendusi kohandada. Kui sobiva komponendi ei leia ja sellel on tulevikus potentsiaali, arendajad ehitavad korduvkasutatava tarkvara algusest peale.

Käesoleva mudeli käigus (Joonis 12) juhtivarendaja kõigepealt otsustab, kas süsteemi komponente saab uuesti kasutada olemasolevatest lahendustest ning esmasel faasil analüüsib ja korjab kokku komponendi nõudeid. Kui nõuded on kirja pandud, siis neid edastatakse arendusmeeskonnale, mis omakorda kontrollib kõigepealt ettevõtete sees varem tehtud lahenduste peale. Arendusmeeskond läheb komponentide hoidlasse ning otsib sobiva lahendust filtreid kasutades. Kui sobiv komponent on üles leitud, siis arendajad hankivad seda endale lokaalsesse keskkonda. Seejärel arendajad muudavad komponendi vastavalt kliendi poolt edastatud funktsionaalsetele nõuetele. Kui komponent on valmis, arendusmeeskond integreerib seda uue süsteemi sisse.



Joonis 12. Tarkvara korduvkasutuse protsessi TO-BE mudel.

Juhul kui sobiva komponendi hoidlast üles leidmine ei õnnestu, tuleb juhtivarendajal analüüsida turu lahendusi. Selles faasis on võimalik hankida nii avatud lähtekoodiga kui ka kommertslahendusi. Kui vajalik vabavara on leitud, saab seda ilma probleemita vastavalt uute nõuetele kohandada ja rakendusse juurutada. Kommertstarkvara peamine

takistus on seotud muudatuste piiranguga, mis tähendab, et komponendi arhitektuuri on keelatud muuta. Seega enne kui tasulisi lahendusi hankida, tuleb neid põhjalikult läbi analüüsida ja veenduda nõuete vastavuses.

Eeldades, et sobiv vaba- ja kommertstarkvara ei ole leitud, juhtivarendaja otsustab analüüsida komponendi potentsiaali. Kui analüüsi käigus tuleb välja, et komponent sobib üldise funktsionaalsuse alla, siis seda läheb kindlasti ka tulevikus vaja ja arendusmeeskond arendab lahenduse korduvkasutatuna. Sellisel juhul võib komponendi kui vabavara arendajate kogukonnale jagada. Vastasel juhul komponendi arendamine toimub kui erilahenduse valmimine ühe kliendi jaoks nagu tavalises süsteemi arendusprotsessis.

4.3 AS-IS ja TO-BE võrdlemine

TO-BE mudeli põhierinevus ADM'i praeguse süsteemiarenduse AS-IS tsükliga seisneb selles, et programmeerimise etapp on rakendatud komponendipõhise korduvkasutuse protsessi alusel. Kui nüüd praegu on ADM'is süsteemiarendus rakendatud lineaarse kosemudeli põhiselt, siis TO-BE lahenduse eesmärgiks oleks jagada tuleviku programmi eraldi komponentideks või mooduliteks ja ehitada neid kõiki paralleelselt. Kui AS-IS mudeli puhul kõik rakendused arendatakse välja algusest peale, siis TO-BE lahenduse puhul kõik süsteemiosad on iseseisvad elemendid, mida peaks võimaluse korral korduvkasutada.

Selleks, et TO-BE lahendus tulevikus efektiivselt toimiks, läheb lisaks vaja ka juhtivarendajat, kes juhiks seda protsessi. Esmasel etapil tema peab kindlaks tegema, milliseid süsteemiosi saab korduvkasutada olemasolevatest lahendustest ja milliseid tuleks juurde arendada. Seega juhtivarendaja peamine roll seisneb komponentide analüüsimises ning turulahenduste otsingus. Hoidlas olemasolevate lahenduste otsinguga suudab arendusmeeskond ise hakkama, kuna nemad paremini haldavad hoidla sisu ning tunnevad lähtekoodi. Sellisel juhul komponendi kohandamiseks läheb kordades vähem aega, kui uut osa arendada algusest peale.

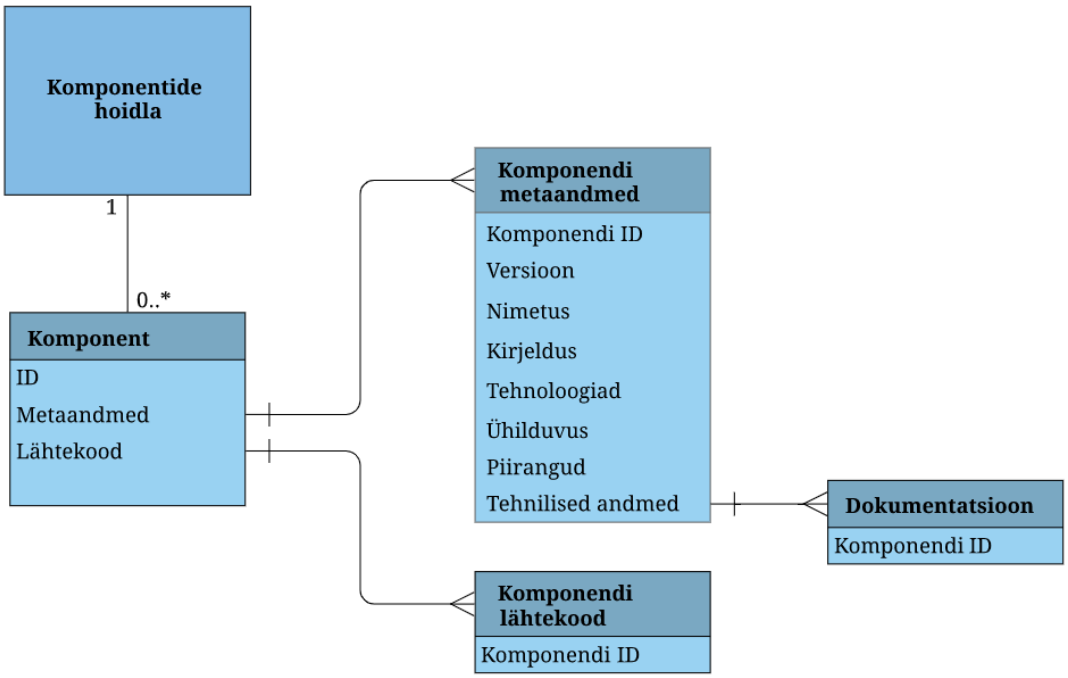
4.4 Metoodikat toetav keskkond

4.4.1 Tarkvara komponentide hoidla

Valmis tehtud komponentide hoidmiseks tuleb ettevõttel muretseda kindel keskkond, kus saab neid soovi korral alati kiiresti välja tuua, hallata ja üles laadida. Tänase seisuga kõik ADM'i lahendused asuvad Github ja BitBucket veebiteenustel. Mõlemad keskkonnad on võetud kasutusele ADM'i projektide majutamiseks.

Üksikute komponentide kätte saamiseks on vaja kindlat infosüsteemi või hoidlat, mis võimaldaks arendajal välja otsida komponente, millest valitakse kõige sobivama. Seda on vaja eelkõige sellepärast, et minimaliseerida koormust, mis võiks korduvkasutatavate komponentide mahtu kasvamisega tekkida. Komponentide hulka võivad kuuluda lähtekood sh klassid ja meetodid, teegid, moodulid või funktsioonid.

Iga komponent peaks sisaldama enda metaandmeid ja lähtekoodi (Joonis 13). Metaandmed võimaldavad arendajal tutvuda komponendi praeguse versiooniga. Metaandmed sisaldavad ka lühikirjeldust, kasutatavaid tehnoloogiaid (programmeerimiskeeled), tehnilisi ühilduvusi ja piiranguid. Lisaks on iga komponendi juurde lisatud selle dokumentatsioon.



Joonis 13. Komponentide metaandmed hoidlas.

Komponentide hoidla eesmärgi paremaks selgitamiseks oli autori poolt koostatud funktsionaalsed ja mittefunktsionaalsed nõuded. See annab selget pilti, mida peab hoidla tegema ja kuidas oma ülesandeid täita.

Funktsionaalsed nõuded:

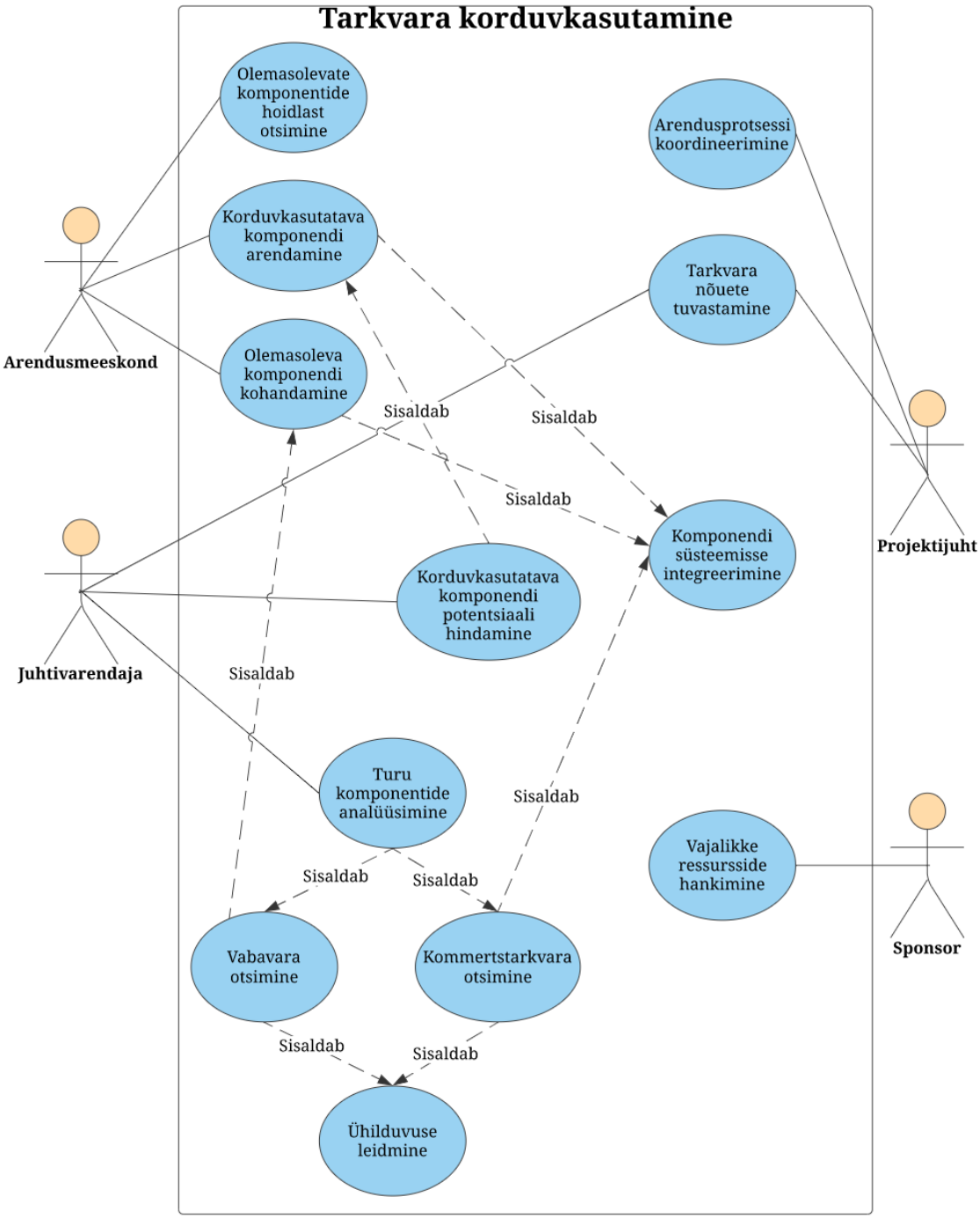
- Otsingus saab märksõnade järgi komponendi üles leida;
- Kasutaja saab komponente alla laadida ja üles laadida;
- Kasutaja saab komponente lisada ja kustutada;
- Otsingus paiknevad filtrid, mis lihtsustavad komponendi üles otsimist;
- Iga komponendi kohta on lisatud dokumentatsioon;
- Süsteem sisaldab ühe komponendi puhul erinevaid versioone.

Mittefunktsionaalsed nõuded:

- Komponendi otsing toimub kiirelt, päringu tegemine ei võta rohkem kui 5 sekundit;
- Loogiline elementide paigutus (parameetrid, nupud, valikud);
- Lihtne ja minimalistik kasutajaliides;
- Hoidlasse ligi pääsevad kõik arendajad.

4.4.2 Meeskonna kirjeldus

Eduka korduvkasutamise metoodika konstrueerimiseks on vaja konkreetset arendusmeeskonda (Joonis 14), mis oleks suunatud olemasolevate komponentide kohandamiseks või nende välja arendamiseks.



Joonis 14. Korduvkasutamise protsessi kasutusjuhtude diagramm.

Diagramm kirjeldab juhtivarendaja, sponsori, arendusmeeskonna ja projektijuhi rollide ülesandeid. Projektijuht planeerib korduvkasutamise protsessi ja koordineerib arendustöid. Lisaks aitab projektijuht juhtivarendajal korduvkasutatava komponendi nõudeid tuvastada. Sponsor on osakonna- või tegevjuht, kes otsustab korduvkasutamise komponenti investeeringu üle ning kontrollib nõutud ressursside olemasolu.

5 Analüüs

5.1 Küsimustiku tulemuste analüüs

Eesti suurte IT-ettevõtete tarkvara korduvkasutamise AS-IS olukorra küsitlusele on vastuseid andnud tegelikult vaid 7 ettevõtet (5%) eeldatavast 144-st. Küsimustiku tulemuste põhjal võib öelda, et tarkvara korduvkasutamisel on suur potentsiaal selle juurutamisse paljude ettevõtete seast. Enamik 7-st vastanutest ettevõtetest on korduvkasutamise metoodikasse investeerinud (Joonis 11) ja läbi viinud vastavaid koolitusi (Joonis 8), mis tähendab, et ettevõtted näevad tarkvara korduvkasutamist kui õiget meetodit arendusprotsesside efektiivsuse tagamiseks (Joonis 9). Seda kinnitab ka kirjanduse ülevaates toodud välja HP näide, kuidas korduvkasutamine on nende tootlikust tõstnud [1]. Kuigi kõik 7 vastanud ettevõtet kinnitasid, et nende kontsernis on tarkvara korduvkasutamine rakendatud, siis tegelikult antud tulemuse põhjal ei saa teha kokkuvõtet kõikide Eesti IT-firmade ja nende tarkvara korduvkasutamise rakendamisest, sest selleks on vaja rohkemate Eesti IT-ettevõtete vastuseid antud küsitlusele. Vaatamata sellele aga selgus, et Eesti IT-firmades tarkvara korduvkasutamine ikkagi leiab aset ja areneb edasi.

Küsimuse tulemused korduvkasutatavate varade rakendamise kohta (Joonis 5) näitavad, et komponendipõhine korduvkasutamine on üks enim kasutatavaid korduvkasutatavaid metoodikaid, mida saavad ka arendajad kõige lihtsamal viisil käsitleda. Metoodika populaarsust kinnitab ka kirjanduse analüüs, mis on üks põhjustest, miks oli antud bakalaureusetöö raames tehtud korduvkasutamise TO-BE lahendus just komponendipõhisel metoodikal tuginedes.

5.2 TO-BE lahenduse analüüs

Tarkvara korduvkasutamise metoodika on ADM'ile vaja eelkõige sellepärast, et vähendada tootmiskulusid, suurendada ettevõtte tootlikust, tõsta uute toodete turule toomisaja ja luua veelgi kvaliteetsemaid ja töökindlamaid tarkvara lahendusi. Kui komponent on pidevas korduvkasutamise tsüklis, siis tõenäoliselt seda kasutatakse mitmes süsteemis korraga. Sellest võib järeldada, et komponendis pidevalt avastatakse uusi vigu ja neid kohe kõrvaldatakse. Korduvkasutatavate komponentide defekti esinemise tõenäosus on sellisel juhul palju väiksem kui tootel, mis oli välja arendatud ja

kasutatud ainult üks kord. Mida vähem edaspidi komponendis vigu avastatakse, seda vähem kulub selle hooldamisele aega. Kvaliteetsete ja töökindlate komponentide olemasolu peaks ettevõtete tootlikust tõsta tänu sellele, et arendajatel kaob vajadus kõiki tooteid otse nullist luua. See laseb kulud ja ressursid kokku hoida tänu vähema testimise, analüüsimise ja dokumenteerimise vajadusele.

Korduvkasutatava komponendi ehitamine on võrreldes olemasolevate lahenduste kohandamisega esmasel etapil kulukam, kuna selle saavutamine nõuab täiendavat tööd ja lisainvesteeringuid. Samas aga universaalse komponendi ehitamine kompenseerib ennast väga kiiresti, kuna komponendi pole vaja igakord modifitseerida ja arendajal kulub teine kord vähem aega selle kohandamisele, sest tema on juba selle komponendi funktsionaalsusega kursis.

5.3 Metoodikat toetava keskkonna analüüs

Komponendipõhise efektiivse korduvkasutamise põhitingimuseks on ettevõtte sisese hoidla olemasolu, mis laseb juba valmis tehtud komponente hoida ja ülal pidada. Arendajal tekib võimalus isegi suurema lahenduste hulga olemasolul neist kõige sobivama kiiresti üles leida. Korduvkasutuse metoodika juurutamisel on hoidlasse investeering algfaasis ettevõtte jaoks soovitatav, kuna selle puudumisel edaspidine komponentide hooldamine on keeruline.

Teiseks, on tarvis investeerida hästi organiseeritud meeskonda, mille koosseisus on kompetentne juhtivarendaja, kes mõistaks komponendi tehnilisi nõudeid ja suudaks kiiresti leida turult sobiva lahenduse ning arendajad, kes oskaks korduvkasutatavaid komponente luua kui ka olemasolevaid kohandada. Juhtivarendaja puhul on peamine kulu seotud olemasolevate turulahenduste otsinguga ja tarkvara nõuete analüüsimisega. Lisaks peab tema universaalse komponendi tootmisel analüüsima selle äriperspektiivi ja veenduma, et seda läheb tulevikus vaja. Universaalse arenduse kontekstis see nõuab tunduvalt rohkem aega, kui tavaline arenduse protsess, kuna arendajad peavad tagama hästi kvaliteetset ja läbi testitud koodi ning kirjutama põhjaliku dokumentatsiooni. Lisaks tuleb arvestada sellega, et klient ei hakka korduvkasutatava komponendi kinni maksma, vaid pigem maksab ettevõte oma sisearenduse eest ise. Peamine kulu on seotud ka arendajate välja koolitamisega, kes oskaks kohe korduvkasutatavaid komponente otsast peale ehitada. Varasematel etappidel sellega saavad tegeleda ainult vanemarendajad.

5.4 Potentsiaalsed riskid

Komponentide korduvkasutamisega tuleb versioonide ühilduvuse kontekstis väga tähelepanelik olla. Kuna tarkvara versioonid koguaeg muutuvad, siis võib tekkida risk, et mõni varem loodud komponent või plagin uue süsteemi ei sobi. Samas ei tähenda see, et vanu komponente ei pea alles hoidma. Äärmisel juhul saab ka neid kasutada ja kohandada uute nõuetega, aga mida vanem on komponent, seda raskemaks muutub selle ümbertegemine.

Sisuhaldussüsteemide puhul nagu Wordpress ja Drupal üks suurimaid puuduseid on see, et kuna need on avatud lähtekoodiga, siis hilisemate uuendustega kaasas käimine on üks parimaid viise, kuidas järjepidevalt turvalisust tagada. Seega kõige optimaalsem meetod oleks enne süsteemi tarkvara versiooni värskendada ja seejärel integreerida sinna kõige hilisema komponendi, et mingeid vigu ei tekiks. Kui CMS on uuendatud, arendajad peavad veenduma, et kõik asjad töötavad õigesti ja vajadusel uuendada ka komponendi järgi. Vahest võib juhtuda, et värskelt valmis tehtud komponent ei ühildu süsteemi uue versiooniga ja seda tuleks ümber teha. Selleks, et versioonide kokkusobimatust maksimaalselt vältida, on vaja komponentide arendamisel alati fokuseerida kõige viimastele CMS tarkvara versioonidele.

5.5 Tuleviku plaanid

Korduvkasutatavate komponentide välja arendamisel võib tulevikus otsustada, kas ehitada tarkvara vabavaralisena ja avalikustada kogu maailmale, või kasutada ainult ettevõtte siseselt. Drupal ja Wordpress on avatud lähtekoodiga süsteemid, mis omavad suurt arendajate kogukoda, kes lisavad pidevalt nii mooduleid kui ka komponente ühiseks kasutamiseks. ADM võiks ka tulevikus panustada korduvkasutatavate varade arengusse, jagades oma komponente teiste arendajate vahel. Sel moel saab igaüks ADM'i poolt loodud varasid läbi testida, jätta veateateid või märkuseid ning nõuda uusi lisafunktsioone. Lisaks võiksid arendajad tulevikus keskenduda mitte ainult komponentide korduvkasutamisele, vaid ka muudele varadele nagu raamistikud, mustrid, arhitektuurid jt. Loomulikult peab arvestama ka sellega, et keegi peaks komponente hooldama. Järeltugi võimaldaks ettevõttel korduvkasutatavaid komponente alati ajakohasena hoida. Samuti see peaks positiivselt ettevõtte mainele mõjuma, mille tulemusena oleks ADM'i varus veelgi rohkem kliente.

Komponente on kõige lihtsam alati ajakohasena ja uuendatuna hoida, kui ettevõttes oleks kindel meeskonnaroll, mille peamiseks ülesandeks on komponentide toetamine. Kui nüüd praegu arendajad uuendavad ja kohandavad olemasolevaid komponente ise, siis selle hooldamise protsessi taga võiks olla korduvkasutamise insener, kes vastutaks lisaks olemasolevate komponentide hoidlas jooksva värskendamise, edasise arendamise ja kohandamise eest. Uue rolli olemasolu laseks ADM'i osakondade arendajatel oma aega kokku hoida kui ka ettevõtte kulusid vähendada. TO-BE lahenduse kontekstis sellist rolli ei ole välja toodud, kuna piisab ka sellest, et arendajad arvestavad teatavate uuendusvajadustega ise ning pärast lisavad uuendatud komponendi hoidlasse, mida saaks ka tulevikus kasutada.

6 Kokkuvõte

Käesoleva lõputöö eesmärgiks oli luua tarkvara komponentide korduvkasutamise metoodika, mis võimaldaks ADM Interactive OÜ osakondadel lühendada uute toodete programmeerimisel kuluvat aja. Antud töö oli teostatud ettevõte nõusolekul, kuna tarkvara korduvkasutamise protsessi pole varem analüüsitud ega praktiseeritud.

Eesmärgi saavutamiseks oli teostatud kirjanduse ülevaade tarkvara korduvkasutamisest ning läbi viidud analüüs ADM'i osakondade vahel, et välja selgitada nende tarkvara korduvkasutamise seisu ja leida ühist tehnoloogiat, mida saaks omavahel kasutada. Parima tulemuse saamiks oli läbi viidud küsitlus Eesti suurte tarkvara firmade vahel, et teada saada nende AS-IS tarkvara korduvkasutamise olukorrast.

Vastavalt saadud tulemustele oli modelleeritud komponendipõhine korduvkasutamise metoodika protsess, mida võrreldi olemasoleva arenduse elutsükliga. Lisaks oli kirjeldatud TO-BE protsessi metoodikat toetava keskkonna nõuded. ADM'i tarkvara korduvkasutamise kontekstis peamine probleem oli seotud protsessi puudumisega, mida välja töötatud TO-BE mudel lahendab ära.

Arvestades olemasoleva korduvkasutamise seisuga ADM Interactive OÜ's, ettevõte oli välja pakutud lahendusega rahul ja kaalub antud metoodikat tulevikus juurutada. Veendumaks, et metoodika on efektiivne ja täidab oma eesmärke, ADM'i juhtkonnal tuleb antud lahendust integreerida tööprotsessidesse. Lisaks on võimalik läbi viia simulatsiooni AS-IS ja TO-BE protsesside vahel ning tuginedes saadud tulemustele võib teha otsust, kas korduvkasutamise lähenemisviis on ennast õigustanud.

7 Kasutatud kirjandus

[1] „What is Software Reuse?,“ The Lombard Hill Group, [Võrgumaterjal]. Available: http://softwarereuse.x10host.com/Article/. [Kasutatud 11 03 2020].

[2] „ADM,“ ADM Interactive OÜ, [Võrgumaterjal]. Available: https://www.adm.ee/kes-me-oleme/. [Kasutatud 04 03 2020].

[3] J. Sametinger, Software Engineering with Reusable Components, 1997, pp. 31-178.

[4] W. Frakes ja K. Kang, „Software Reuse Research: Status and Future,“ IEEE Xplore, 2005.

[5] H. Nakano, „A METHODOLOGY FOR SOFTWARE REUSE,“ UNIVERSITY OF WISCONSIN-LA CROSSE, 2006.

[6] „PMI Disciplined Agile,“ Project Management Institute, Inc, [Võrgumaterjal]. Available: https://www.pmi.org/disciplined-agile/process/reuse-engineering. [Kasutatud 10 04 2020].

[7] E. S. De Almeida, V. Cardoso Garcia, V. Burégio, A. Alvaro, J. C. Cordeiro Pires Mascena, L. Nascimento ja S. Meira, C.R.U.I.S.E Component Reuse in Software Engineering, C.E.S.A.R e-book, 2007, pp. 20-21.

[8] K. Sherif ja A. Vinze, „Barriers to adoption of software reuse: A qualitative study,“ ScienceDirect, 2003.

[9] „Source Making,“ [Võrgumaterjal]. Available: https://sourcemaking.com/design_patterns. [Kasutatud 04 04 2020].

[10] E. K, „Linux Academy,“ 2016. [Võrgumaterjal]. Available: https://linuxacademy.com/blog/devops/development-artifacts/.

[11] D. BOSE, „COMPONENT BASED DEVELOPMENT,“ Arxiv, 2010.

[12] C. McClure, Software Reuse: A Standards-based Guide, New York: IEEE Xplore, 2001.

[13] P. R.S., Software Engineering – A Practitioner’s Approach. Seventh Edition, Saudi Electronic University, 2001.

[14] A. C. Wills, D. D. Souza ja I. Computing, „Rigorous Component-Based Development,“ Semantic Scholar; Components, pp. 1-28, 1997.

[15] F. Imeri ja L. Antovski, „An Analytical View on the Software Reuse,“ ResearchGate, 2012.

[16] T. Ravichandran ja M. A. Rothenberger, „Software reuse strategies and component markets,“ ResearchGate, 2003.

[17] W. Kim, „On Issues with Component-Based Software Reuse,“ Journal of Object Technology, 2005.

[18] „OpenSource,“ Red Hat, Inc., [Võrgumaterjal]. Available: https://opensource.com/resources/what-open-source. [Kasutatud 24 04 2020].

[19] M. A. Rothenberger, K. J. Dooley, U. R. Kulkarni ja N. Nada, „Characteristics of Software Reuse Strategies: A Taxonomy of Implementations Patterns,“ Management Information Systems Research Center, 2002.

[20] L. Antovski ja F. Imeri, „Review of Software Reuse Processes,“ ResearchGate, 2013.

[21] W. N. Robinson ja H. G. Woo, „Finding reusable UML Sequience Diagrams Automatically,“ Institute of Electrical and Electronics Engineer, 2004.

[22] R. Nath ja H. Kumar, „Issues in Software reuse,“ *National Conference on Futuristic Trends in Engineering and Technology*, Haryana, 2007.