TARTU ÜLIKOOL
Sotsiaalteaduste valdkond
Ühiskonnateaduste instituut
Info- ja teadmusjuhtimine õppekava

Anni Adamson

Infosüsteemi arendamise põhimõtted muutuvate nõuete korral Eesti avalikus sektoris

Magistritöö

Juhendaja: Reeni Mikkelsaar, MA
Maris Männiste, MA

Tartu 2020

SISUKORD

SISSEJUHATUS

Eesti riik on jäänud IT-süsteemide pantvangi (Reinsalu, 2018) - vajalike tarkvaraarenduste arendamine on ajamahukas. Keeruline on hinnata nende loomiseks ja ülalpidamiseks kuluvaid ressursse ning mitte alati ei ole nad kasutajasõbralikud ja efektiivsed. Iga arendusprojekti õnnestumine sõltub kolmest komponendist: aeg, maksumus ja kvaliteet (ingl *time, cost* and *quality*), mille vahel tuleb leida tasakaal (Atkinson, 1999). On ilmselge, et piiratud ajakava puhul peab olema suurem eelarve või tuleb järele anda süsteemi soovitud funktsionaalsustes. Järjest vananeva rahvastiku tulemusel ning vähenevate ressursside tingimustes, tuleb avalikul sektoril aga pakkuda üha paremaid e-teenuseid. Info- ja kommunikatsioonitehnoloogiatele (edaspidi IKT) tuginevate teenuste arendamine on hea võimalus kvaliteetselt ja kuluefektiivselt avalikke teenuseid osutada ja edasi arendada. Siinkohal ei tohi aga unustada, et IKT on vaid vahend ja e-teenuste kanal avalike teenuste toimimiseks (ATKRR, 2013). Selleks, et tagada inimestele kvaliteetse teenuse kättesaadavus sõltumata asukohast, võimalikult kliendisõbralikult ning optimaalse kulu-tulu vahekorraga, on oluline panustada infosüsteemide arendusprotsessi parendamisesse terves riigis. Riigi prioriteet peab olema arendada ajakohaseid tarkvaralahendusi selle valmimise hetke või isegi tuleviku vaates, mille saavutamiseks on oluline roll muutuvate nõuete juhtimisel.

Magistritöö **eesmärk** on muutuvate tarkvaranõuete juhtimisprotsessi analüüs Eesti avaliku sektori arendusprotsessi näitel, kaardistades võimalikud ohukohad ja noppida välja kriitilised edutegurid tagamaks riigieelarve võimalikult optimaalne kasutus ning maksimaalne kasu. Piiratud tingimuste all käsitlen nii hankepoliitikat, regulatsioone, ajaraami kui ka pidevalt muutuvat konteksti, et saavutada edu ka muutuvate tarkvaranõuete korral. Töö keskendub tarkvaraarendusprotsessi nõuete muutuste juhtimisele, muuhulgas on töös käsitletud lähteülesande püstitamine ja kogu projekti juhtimine. Selle ilmestamiseks on kirjeldatud Lennuametis (edaspidi LA) arendatavat drooniregistri arendusprotsessi, mis on küll valmimise järgus, kuid juba täna ilmneb aspekte võimalikeks muutusteks arendusprotsessi vältel. Arendatava drooniregistri puhul on oluline, et see saaks maksimaalselt paindlik ning võtaks arvesse võimalikult palju aspekte juba eos, et tagada maksimaalne koosvõimelisus lähitulevikus. Magistritöö teema haakub oluliselt minu põhitööga Lennuametis, olles peatselt algava arendusprojekti

raames loodava Lennuameti järelevalve infosüsteemi (edaspidi LAIS) peakasutaja ning kuuludes projekti juht- ja töörühma, mistõttu jagan ekspertarvamust oma kogemusest lähtuvalt. Sellest tulenevalt kasutan sekundaarandmetena enda protokollitud andmeid LA arendusprojekti juht- ja töörühmast ning teistest projekti kohtumistest.

Töö **ulatus** piirdub infosüsteemi arendamise ja selle muutuvate nõuetega, mis on tingitud peamiselt kliendist ning mis eelkõige suurendavad arendustööde mahtu. Töös olen vaadelnud arendusprotsessi juhtimismudeleid ja põhimõtteid ning lähemalt tutvunud tarkvaranõuete muutuste juhtimise protsessidega. Töö tulemusel annan ettepanekud Majandus- ja Kommunikatsiooniministeeriumi (edaspidi MKM) IT-arendusosakonnale, et neid rakendada peatselt algavas LA-s arendatava drooniregistri arenduse protsessis. Uuritava **teema aktuaalsus** seisneb infosüsteemide arendusprotsessi käigus muutuvate nõuete juhtimises, mida kinnitas ka Riigikontroll (2019) avaliku sektori tarkvaraarenduse projektide juhtimise kohta läbiviidud auditis - arendusprojektide tõhusat elluviimist takistavad muudatused, mis tehakse arenduse ajal õigusaktidesse. Lisaks sellele ei mõelda arendusprojektide algatamisel infosüsteemide hilisemale jätkusuutlikkusele ning sageli ei vasta valminud tarkvara kõikide kasutajarühmade vajadustele. Sellest lähtuvalt keskendub käesolev magistritöö tarkvaraarenduse nõuete muutuste juhtimisele, kuna nõuete muutumine on üks peamisi väljakutseid kogu tarkvaraarenduse elutsüklis (Anwer, Wen, Wang, 2019).

Uurimisküsimused:

1) Millised nõuded on kõige altimad muutuma?
2) Mis on tarkvaranõuete muutumise peamised põhjused?
3) Millised on avaliku sektori praktikad muutuvate tarkvaranõuete juhtimiseks? Milliseid protsesse ja tehnikaid selleks kasutatakse ning mis on peamised väljakutsed?

Eesmärgini jõudmiseks olen kasutanud kvalitatiivset metoodikat - viinud läbi poolstruktureeritud intervjuud ekspertidega ning teinud kvalitatiivset sisuanalüüsi dokumentide osas, mille pinnalt olen teinud järeldused ja ettepanekud avaliku sektori infosüsteemide arendusprotsessi parendamiseks. Samuti, olles LAIS peakasutaja, mille üheks osaks on drooniregister, on minu isiklik huvi olla targa tellijana maksimaalselt teadlik, et juhtida süsteemi arenduskäiku efektiivselt õiges suunas. Magistritöö jaguneb neljaks peatükiks, mis tutvustavad infosüsteemi arendamise põhimõtteid, tarkvaranõuete olemust, nende muutumist ja seda, kuidas muutustega toime tulla; ülevaadet ja näiteid Eesti avaliku

sektori tarkvaraarenduste kohta; kirjeldust meetodist ja valimist; intervjuude analüüsi, näidisjuhtumi kirjeldust ja ülevaadet ning järeldusi ja diskussiooni, mille ühe osana on toodud ettepanekud.

MÕISTED

AirMap - maailma juhtiv õhuruumiteenuste platvorm mehitamata õhusõidukite tarbeks.

EASA (ingl *European Aviation Safety Agency*) - Euroopa Lennundusohutusamet.

Euroopa Liidu Struktuurifondid (SF) - Euroopa Liidu regionaalpoliitika elluviimiseks ja arendamiseks ette nähtud toetusfondid.

IEEE (ingl *Institute of Electrical and Electronics Engineers*) on tunnustatud institutsioon terves maailmas, mis toodab üle 30% kogu maailma kirjandusest elektri- ja elektroonikaseadmete ning arvutiteaduste alal (IEEExplore, 2020).

Kasutajalugu (ingl *user story*) ehk stsenaarium, mida kasutaja võib kogeda.

LA - Lennuamet (ingl *Estonian Civil Aviation Administration ECAA*).

LAIS - LA järelevalve infosüsteem, mille üks osa on käesolevas töös näitena toodud drooniregister, hõlmates endas drooni registreerimist, selle käitaja registreerimist ning pädevuste haldust.

Mehitamata õhusõiduk (ingl *Unmanned Aircraft System UAS*) ehk droon on õhusõiduk, mille pardal ei ole pilooti ning mille juhtimine toimub tehniliste abivahendite vahendusel või eelnevalt programmeeritud lennuna ilma piloodi juhtimiseta autonoomselt (LA koduleht ecaa.ee).

Nõue (ingl *requirement*), mis defineerib kasutaja vajadused ja eesmärgid, arendatava süsteemi tingimused ja omadused, näiteks organisatsiooni vajadused, seadus ja standardi (Pohl, 2010).

Sidusrühm (ingl *stakeholder*) ehk huvirühm - isikud või organisatsioonid, kellel on huvi arendatava süsteemi vastu (Pohl, 2010).

Soovilogi (ingl *backlog*) ehk mahajäämus, lõpetamata tööde või asjade kogumine, mis vajavad lahendamist (Agilealliance, 2020).

UTM (ingl *Unmanned Traffic Management UTM*) mehitamata õhusõidukite liikluskorraldussüsteem.

1. INFOSÜSTEEMI ARENDUSPROTSESS JA MUUTUVAD NÕUDED

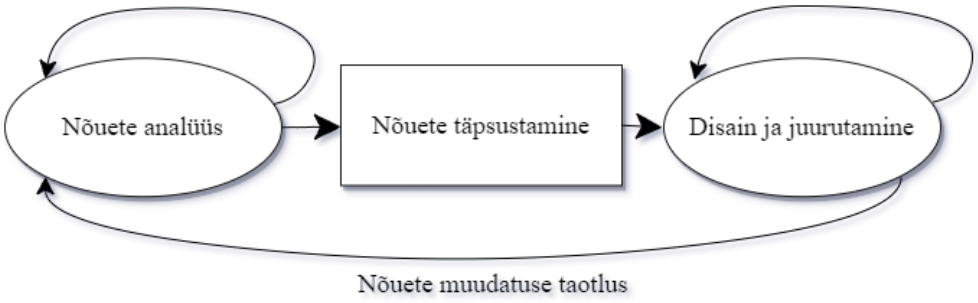
Tarkvara on olnud osa ühiskonnast juba rohkem kui 50 aastat, mil on kasutusel olnud väga palju erinevaid tarkvara arendamise metoodikaid. Mõnel organisatsioonil on enda jaoks kohandatud täiesti oma meetod, kuid peamiselt räägitakse kahest: **traditsiooniline tarkvara arendamise viis** (ingl *heavyweight*) (näiteks *Waterfall, Unified Process, Spiral Model* jne) ja **agiilne tarkvara arendamise metoodika** ehk kerge tarkvaraarendus (ingl *lightweight*) (Awad, 2005). Mõlemal meetodil on omad tugevused ja nõrkused, kuid peamiselt on meetodi valikul määravaks projekti suurus, inimesed ja riskitegurid (Awad, 2005: 35). Muuhulgas eristavad nimetatud kahte metoodikat muutustega toimetulek, dokumenteerimise praktikad, arendustsüklite arv jne (vt Tabel 1).

Tabel 1. Erinevused agiilse ja traditsioonilise meetodi vahel

	Agiilne tarkvaraarendus	**Traditsiooniline tarkvaraarendus**
Lähenemisviis	Kohanemisvõimeline, kergesti muudetav	Ennustav, kindlalt määratletud
Edu mõõdik	Äriväärtus	Plaanile vastavus
Kvaliteedi tagamine	Pidev kontroll	Planeerimine ja range kontroll
Projekti sobiv suurus	Väike	Suur
Testimine	Arenduse käigus, pidevalt	Pärast koodi valmimist
Juhtimisstiil	Koostööl põhinev	Käskiv ning kontrolliv
Nõuded	Pidevalt muutuvad	Stabiilsed, projekti alguses teada
Muutuse perspektiiv	Muutuse suhtes kohanemisvõimeline	Muutust alalhoidev
Dokumentatsioon	Vähene ja nõrk	Kohmakas ja massiivne
Rõhuasetus	Inimestele orienteeritud	Protsessile orienteeritud
Tsüklid	Arvukalt	Limiteeritud
Esmane planeerimine	Minimaalne	Kõikehõlmav
Kliendi kaasatus	Kõrge, klient on osa meeskonnast	Madal, klient näeb valmistoodet
Investeeringu tasuvus	Projekti alguses	Projekti lõpus
Meeskonna suurus	Väike/ loominguline	Suur
Esmane eesmärk	Kiire tulemus	Kõrge turvalisus

(Allikas: Awad, 2005; Kivimaa, 2014)

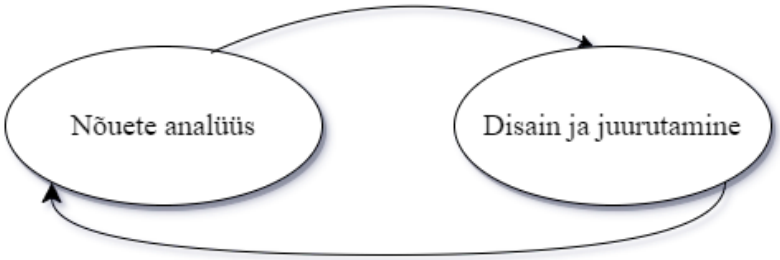
Traditsiooniline tarkvara arendamise metodoloogia on olnud kasutuses juba pikka aega - alates laialt levinud koskmudelist (Royce, 1970), mida on edukalt kasutatud nii suuremate kui väiksemate tarkvaraprojektide puhul. Vaatamata edule on sellel ka puudusi: näiteks lineaarsus, paindumatus/jäikus muutuvates nõuetes ja formaalsete protsesside järgimine sõltumata projekti mahust (Awad, 2002: 35). Traditsiooniline lähenemine on aga juhitud etapiliselt ja plaani põhiselt (vt joonis 1), kus arendusprojekt algab kõikide nõuete komplekti määratlemise ja dokumenteerimisega, millele järgneb lahenduste loomine (ingl *solution building*), testimine (ingl *testing*) ja juurutamine (ingl *deployment*), mistõttu on vajalik nõuete kogumi määratlus ja dokumentatsioon kohe projekti alguses (Awad, 2005: 3). Kui süsteemi nõuded muutuvad, tuleb ennatlikult olla valmis olemasoleva tarkvara spetsifikatsiooni ja disaini ümber tegemiseks (Sommerville, 2016: 75), misjärel kordub kogu protsess uuesti. Traditsioonilise lähenemise puhul võib olla muudatuste sisseviimine raskendatud, kuna kogu tegevus toimub kinnitatud plaani alusel ning käsu ja kontrollimise põhimõttel. Mitmed praktikud on leidnud, et traditsioonilise lähenemise juhitus ja rangus, projektipõhisus ning formaalsus (Sommerville, 2016) tekitavad raskusi algnõuete muudatuste korral (Awad, 2005), mistõttu võib olla tõenäoline projekti ebaõnnestumine.



Joonis 1. planeerimisel põhinev arendusprotsess (Allikas: Sommerville, 2016: 74).

Agiilne tarkvara arendamise metodoloogia on kiirem ja väledam kui traditsiooniline mudel, mis keskendub tarkvarale endale, lubades tegeleda tarkvara spetsifikatsiooni, arenduse ja tarnimisega järkjärguliselt (Sommerville, 2016: 74) eesmärgiga vähendada bürokraatiat ning suurendada kliendikesksust (Beck jt, 2001), vt joonis 2. Agiilsele metodoloogiale on iseloomulik inimestele orienteeritus, kohanemisvõimelisus, tegelikkusele vastavus, tasakaalustatud paindlikkus ja planeerimine, empiirilisus, detsentraliseeritus, lihtsus, koostöö ja väikesed iseorganiseeruvad meeskonnad (Awad, 2005). Täpsemad põhimõtted ja lähenemisviisid on määratud kindlaks rahvusvaheliselt tuntud dokumendiga „Agiilse Tarkvaraarenduse Manifest“ (Beck jt, 2001). Agiilsed

metoodikad sobivad paremini pidevalt muutuvate nõuetega tarkvara arendusprojektide puhul (Sommerville, 2016: 75), keskendudes arendaja ja kliendi koostööle ning toetades varajast tarnimist (Awad, 2005). Nõuded jagatakse väikesteks kasutajalugudeks (Sommerville, 2016: 80), misjärel hakatakse neid ühekaupa realiseerima, mis annab võimaluse uuenduste sissetoomiseks ka keset arendust, sest iga iteratsiooni nõudeid saab käsitleda eraldiseisvalt (Sommerville, 2016: 80).



Joonis 2. Agiilne tarkvaraarendus (Allikas: Sommerville, 2011: 74).

Tänapäeval on üha levinud agiilsed metoodikad, sest keskkond on kiires muutumises ning tarkvaraarenduses on järjest keerulisem ennustada arenduse käigus ette tulevaid probleeme ja muutuseid. Muudatusi vajavate tõrgete korral on neid lihtsam parandada kiireid meetodeid kasutades, kuna probleemidele saab kohe lahendust otsima hakata see sama väike meeskond, kes projektiga tegeleb (Kivimaa, 2014).

1.1. Nõuded ja nõuete analüüs

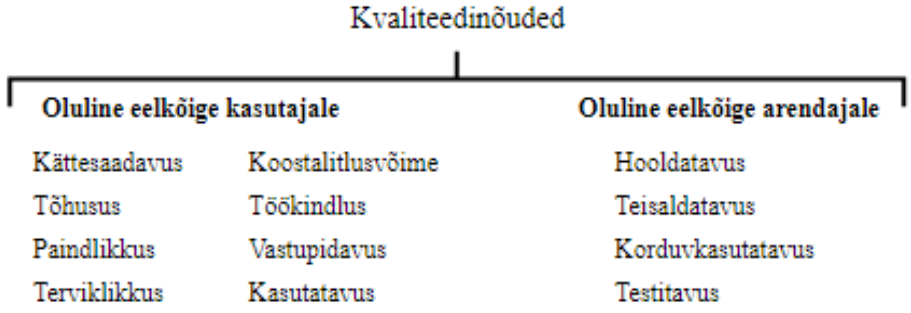
Nõue on oluline faktor iga projekti arenduses, mis defineerib sidusrühmade vajadused ja selle, kuidas süsteem nende vajadusi täidab, loomulikus keelelises vormis, et seda oleks kõigile lihtne mõista (Pohl, 2010).

IEEE standard nr 729 on defineerib nõude järgmiselt: /…/ …kui tingimus või võime, mida kasutaja vajab probleemi lahendamiseks või eesmärgi saavutamiseks; süsteemi või süsteemi komponendi kirjeldus, et süsteem vastaks lepingule, standardile, tehnilisele kirjeldusele või muule ametlikult kinnitatud dokumendile /…/ (IEEExplore, 2020). See seab paika kriteeriumid, mis vastaksid lepingule, standardile, tehnilisele kirjeldusele või muule ametlikult kehtestatud dokumendile; ka selle, milline peab olema nende esituse dokumenteeritus (Balaban, Strum, 2018). Nõuded on justkui valmiva lahenduse kasutusjuhend, et selgitada lahti, mida ja kuidas süsteem tegutsema hakkab.

Nõuded aitavad analüütikul aru saada, missugused elemendid ja funktsioonid on konkreetse projekti arenduses olulised, olles sisendiks kavandamise (ingl *design*), juurutamise (ingl *implementation*) ja valideerimise (ingl *validation*) faasides (Pohl, 2010). Nõuded jagunevad kolme tüüpi: funktsionaalsed nõuded (ingl *functional requirements*); mittefunktsionaalsed nõuded (ingl *non-functional requirements*), mis on tuntud ka kui kvaliteedinõuded (ingl *quality requirements*) (Lauesen, 2002; Bourque ja Fairley, 2014); ja kitsendused (ingl *constraints*) (Pohl, 2010).

Funktsionaalsed nõuded täpsustavad süsteemi funktsionaalsuse, vastates küsimusele, mida peab infosüsteem tegema ehk väljakirjutis süsteemi teenustest, mida süsteem pakub, kuidas süsteem reageerib antud sisendile ja kuidas käitub teatud situatsioonides, näiteks: ärinõuded, administratiivsed funktsioonid, autentimine, välised liidesed, aruandlusnõuded, arhiivinõuded ning juriidilised või regulatiivsed nõuded (Pohl, 2010: 18). Tavaliselt on funktsionaalsed nõuded dokumenteeritud kolmes perspektiivis: defineerides süsteemiandmed, -funktsioonid ja -käitumised (Pohl, 2010: 18).

Mittefunktsionaalsed nõuded on kirjeldus sellest, kuidas peab süsteem vajalikke funktsioone täitma, määratledes süsteemi omadused - töökindlus, reageerimiskiirus, andmete hoiustamise nõuded, jne (Sajjad, Hanif, 2010). Mittefunktsionaalsed nõuded on tuntud ka kui kvaliteedinõuded (Bourque, Fairley, 2014), mis jagunevad mitmeks eri tüübiks: süsteemi suutlikkus (ingl *performance*), -usaldusväärsus (ingl *reliability*), -stabiilsus (ingl *stability*) (Pohl, 2010), -hooldatavus (ingl *maintainability*), -ohutus (ingl *safety*), -turvalisus (ingl *security*), ja -koostalitlusvõime (ingl *interoperability*) (Bourque ja Fairley, 2014). Kuid neid võib jaotada ka lähtuvalt sihtrühma huvist, näiteks Wiegers & Beatty (2014) on jaotanud mittefunktsionaalsed nõuded kasutaja arendaja perspektiivist (vt joonis 3).



Joonis 3. Mittefunktsionaalsete ehk kvaliteedinõuete taksonoomia (Wiegers, Beatty, 2014).

Kvaliteedinõudeid võib defineerida konkreetse teenuse, funktsiooni või süsteemi kvaliteedi atribuutidena (Pohl, 2010), mis piiravad lahendust (Bourque, Fairley, 2014). Mittefunktsionaalse nõudena võidakse täpsustada ka näiteks konkreetse süsteemi programmeerimiskeelt või arendusmeetodit, mis võivad olla isegi funktsionaalsetest nõuetest olulisemad, kuid kui neid ei täideta, on süsteem kasutu (Sajjad, Hanif, 2010).

Kitsendusi tuntakse ka kui pseudonõudeid, mille kehtestab klient, keskkond või kontekst, milles süsteem töötab (Lauesen, 2002; Pohl, 2010: 22). Need võivad olla sisend-/ väljundseadme võimalused ja süsteemi esindatuse keskkond (Lauesen, 2002).

Nõuete analüüs (ingl *requirement engineering*) on oluline faas arendusprotsessis, et tuua esile nõuded, analüüsida neid ja esindada neid analüüsimudelites, mis on sisend arhitektuurilisele disainile (ingl *architecture design*) (Al-Saiyd, Zriqat, 2015), olles justkui teenuste loomine kliendile ehk kirjeldus sellest, mida klient süsteemilt nõuab, mille alusel see töötab ja mida arendatakse. Nõuete analüüsi protsess hõlmab endas nõuete kaardistamist (ingl *requirement elicitation*), nõuete täpsustamist (ingl *requirement specification*), nõuete kontrollimist ja kinnitamist (ingl *requirement verification and validation*) ning nõuete haldamist (ingl *requirement management*) (Pohl, 2010). See on järkjärguline ja korduv protsess, mida viiakse läbi paralleelselt muude tarkvara arendustegevustega nagu projekteerimine, juurutamine, testimine ja dokumenteerimine (Hussain, Kamal ja Mkpojiogu, 2016). Nõuete analüüs rõhutab infosüsteemi arendamise eesmärki, viidates täpsele spetsifikatsioonile, mis loob aluse nõuete **analüüsimiseks** (ingl *analyzing*); **valideerimiseks** (ingl *validating*), et nõuded tõepoolest kirjeldavad seda, mida sidusrühmad soovivad; nõuete **määratlemiseks** (ingl *defining*), mida arendajad peavad ehitama ning **kontrollides** (ingl *verifying*), kas nad on tarninud seda õigesti (Hussain jt, 2016).

1.1.1. Nõuete kaardistamine, täpsustamine, valideerimine ja jälgitavus

Nõuete kaardistamine on nõuete analüüsi kõige esimene etapp, kus kaardistatakse tööprotsess, et mõista mida sidusrühmad teevad ning kuidas loodavat süsteemi saab selle töö toetamiseks kasutada. Sommerville (2011) on nõuete kaardistamise veel omakorda jaotanud järgmiselt alaprotsessideks - nõuete avastamine ja mõistmine; -klassifitseerimine; -prioritiseerimine ning viimaseks - dokumenteerimine. Tabel 2 võtab kokku võimalikud märkused süsteeminõuete kirjeldamiseks. Võimalikud viisid nõuete kaardistamiseks on näiteks intervjuu, inimeste jälgimine (ingl *ethnography*),

lugude ja stsenaariumite (ingl *stories and scenarios*) sõnastamine ehk täpne selgitus sellest, kuidas klient tarkvara kasutab (Sommerville, 2011: 112-120).

Nõuete spetsifikatsioon (ingl *requirement specification*) ehk kasutaja- ja süsteeminõuete dokumenteerimine nõuete dokumendis (Sommerville, 2011: 120-129).

Tabel 2. Märkused süsteeminõuete kirjeldamiseks

Loomulikus keeles laused	Nõuded kirjutatakse loomulikus keeles nummerdatud lausete abil. Iga lause peaks väljendama ühte nõuet.
Struktureerituse kontseptsioon	Nõuded on kirjeldatud tüüpvormi või malli kasutades, kus iga väli annab teavet konkreetse nõude ühe tüübi kohta.
Graafilised täiendused	Graafiliste mudelite kasutamine, millele on lisatud teksti vormis märkused funktsionaalsete nõuete kohta, näiteks UML (ingl *unified modeling language*) ja protsessidiagrammide kasutamine.
Matemaatilised spetsifikatsioonid	Need tähistused põhinevad matemaatilistel mõistetel, näiteks piiratud olek (ingl *finite-state machines or sets*). Kuigi need ühemõttelised spetsifikatsioonid võivad osalt vähendada ebamäärasust nõuete dokumendis ning enamik kliente ei pruugi selle tähendust mõista. Samuti ei ole võimalik kontrollida kas märgitud täpsustused ikka tähendavad seda, mida klient tahab. Täpsemalt on Sommerville (2011, lk 285-305) käsitlenud seda süsteemi töökindluse juures.

(Allikas: Sommerville, 2011: 120-129)

Nõuete valideerimine (ingl *requirement validation*) ehk nõuete kontrollimine, mis tagab nõuete vastavuse tellija vajadustele (Pohl, 2010). Vähendamaks võimalikke tekkivaid vaidlusi tellija ja arendaja vahel on oluline, et nõuded oleksid kirjeldatud selliselt, et neid oleks võimalik kontrollida nö efektiivsus-kriteeriumid, mille põhjal saab öelda, et tarnitud süsteem vastab kõigile määratletud nõuetele. Valideerimine sisaldab endas kehtivuse- (ingl *validity*), järjepidevuse- (ingl *consistency*),

täielikkuse- (ingl *completeness*) ja reaalsuse kontrolli (ingl *realism check*) ning kontrollitavust (ingl *verifiability*) (Sommerville, 2011: 129-130).

Tarkvaranõuete jälgitavus (ingl *traceability*) on arendusprotsessis vajalik komponent (Mäder, Egyep, 2015), mis määratleb tarkvaranõude eluea - alates selle tekkest, arenemise ja täpsustamise, nõude edasise kasutuselevõtu, ning pidevalt kogu täiustamise perioodil ükskõik millises arendusetapis (Gotel, jt, 2012). Näiteks Mäder'i ja Egyep'i (2015) avastused näitavad, et jälgitavusvõimega subjektid sooritasid etteantud ülesande 24% kiiremini ja lõid keskmisel 50% õigemaid lahendusi, mis viitab sellele, et jälgitavus mitte ainult ei säästa pingutusi vaid võib oluliselt parandada ka hoolduse kvaliteeti.

1.1.2. Nõuete roll tarkvaraprojekti õnnestumisel või ebaõnnestumisel

Kamumi (2019) on defineerinud edukat tarkvaraprojekti selliselt, mis valmib õigeaegselt eraldatud eelarve piires ning millel on kõik algselt määratletud tunnused ja funktsionaalsused. Vastupidiselt, ebaõnnestunud projekt on jällegi see, mis võib olla küll lõpule viidud, kuid aja ja eelarve ületamisega ning vähemate tunnuste ja funktsionaalsustega kui algselt määratletud, mis katkestatakse või tühistatakse enne selle valmimist (Kamuni, 2015). Samas, kuivõrd täpne on lugeda ebaõnnestunuks projektiks ainult projekti mitte mahtumist ajakavasse, eelarvesse või funktsionaalsusesse, kui seejuures siiski tarniti töötav arendus? Kütt'i (2019) sõnul ei saa olla edukas projekt see, mis tarnib viis aastat tagasi kabinetivaikuses defineeritud nõudeid võimalkult odavamal viisil juhuslikult valitud kuupäevaks, kuna puudu on nii kasutaja, vahepeal toimunud muutused kui hoolduskulud. Uuringutest (Bano, Imtiaz, Ikram, Niazi ja Usman, 2012) selgub, et tarkvaraprojekti tõrked on peamiselt tingitud puudulikest nõuetest (ingl *inadequate requirements*), muutuvatest nõuetest (ingl *changing requirements*), kehvadest/puuduvatest nõuetest (ingl *poor requirements*) või teostamatust ootustest (ingl *impracticable expectations*) (Hussain jt, 2016). Õnnestunud projekti tarnimine projekti lõpus eeldab tõhusat nõuete analüüsi ja nõuete juhtimist kogu protsessi käigus. Al-Saiyd ja Zriqat (2015) on jaotanud nõuete muutumise põhjused järgmistesse kategooriatesse: inimfaktor; protsess; toode; ja organisatsioon.

Inimfaktor jaguneb omakorda kolmeks: kasutaja/klient; projekti meeskond ja kolmas osapool (Al-Saiyd, Zrigat, 2015). Kasutaja/kliendi perspektiivis on ülimalt oluline just kaasatus, teadmised, pühendumus, koostöö ja valmisolek, sest üks kõige levinum probleem on **kliendi ebamäärane**

ettekujutus sellest, mida ta tegelikult vajab, mistõttu nõuded jäävad tähelepanuta (Bigelow, 2019). Nõrk tellija ei suuda jälgida projekti kulgemist, tal puudub piisav teadlikkus infosüsteemide arendamisest või soovib ta muuta põhifunktsioone projekti kestel, mis omakorda tingib vajaduse muuta paljusid teisi funktsioone või konfigureerida kogu süsteem. Kogemus kasvatab teadlikkust ehk iga arendusprojekti käigus areneb tellija teadlikumaks, mis on efektiivse kommunikatsiooni nurgakivi. Projekti tiimi perspektiivist on ülimalt olulised oskused, teadmised, kogemused, motivatsioon ja pühendumine. Kommunikatsioon on aga igakülgselt esmatähtis, sest tihti on **puuduseks just selge ja avatud kommunikatsioon tellija ja arendaja vahel** kogu arenduse vältel (Bigelow, 2019), mis võib põhjustada segadust ja teineteise vääriti mõistmist. Näiteks erinevatel huvigruppidel (süsteemi kasutajad, projekti meeskond, kliendi esindaja(d) ja kolmas osapool) võivad puududa teadmised tehnoloogia ja funktsionaalsuse keerukuse mõistmiseks, mistõttu nad ei mõista muudatuse tõsisust ja seda, mida selle muudatuse rakendamine endas kujutab (Grebennikov, 2018). Siin on oluline roll projektijuhil, et tagada mõlema osapoole teineteise mõistmine saavutatavast eesmärgist ja selle saavutamiseks vajaminevatest ülesannetest, et puuduks võimalus uute defektide tekkimiseks. Riskide realiseerumise vältimiseks on oluline, et kõik sihtrühmad mõistavad projekti nõuete muutmise olulisust ja rakendavad vastavaid meetmeid.

Kategooria **protsessi** on Al-Saiyd ja Zrigat (2015) jaotanud kaheks: muutuse identifitseerimine ja planeerimine. Kui muutuste kaardistamine keskendub pigem muudatuse põhjusele, tüübile ja allikale, siis planeerimine omakorda pingutusele, selle hinnangulisele maksumusele ning aja planeerimisele. On ilmselge, et nõuded ei ole kaitstud muutuste ees (Alsanad, Chikh ja Mirza, 2019: 1) ning **projekti esmases spetsifikatsioonis toimuvad muutused** on paratamatus. Tihtilugu on muutused tarkvaranõuetes küll ebameeldivad ja protsessi pärssivad, kuid annavad olulise panuse lõpptoote kvaliteedile ja funktsionaalsusele. Üldise probleemina võib tuua välja ka **ebamõistliku ajakava aktsepteerimise** ilma aruteluta ja enne detailanalüüsi läbiviimist (Sommerville, 2011). Projekti ulatus ning selleks vajaminevate ressursside mõistmine annab tervikpildi tervest arenduse protsessist. Oluline roll on kindlasti planeerimisel ning algsetel kokkulepetel, et kuidas toimub protsess, kui algses plaanis esinevad probleemid, on vaja teha vastavaid parandusi või hoopis tekib muudatustega vajadus algselt pakutust teistsuguse lahenduse järgi, mis aga kahtlemata loob väljakutseid arendajale.

Tarkvaratootega seotud põhjused jagunevad omakorda veel tarkvara (ingl *Software*) ning platvormi vahel (Al-Saiyd, Zrigat, 2015). Toote tarkvaraga seotud põhjused võivad olla seotud näiteks lähtekoodi, arhitektuuri, tööriistade, dokumentatsiooni, integratsiooni, liideste jt.

Viimaseks kategooriaks on **organisatsiooniga** seotud põhjused, hõlmates peamiselt strateegilise planeerimise - sealhulgas juhtpõhimõtted ning eesmärgid. Kui arendajal lasub oluline vastutus, et **mõista ja üheselt aru saada tellija organisatsiooni poliitikast,** siis samas peab ka tellija olema valmis muutusteks organisatsiooni sees või seda ümbritsevas keskkonnas (Bigelow, 2019).

Iga muudatusega kaasneb arendusprojekti raames risk arendusprotsessi kulude suurenemiseks, võides tähendada juba tehtud töö ümbertegemist, mille tulemusena võib olla vajalik süsteemi ümberkujundamine, uute nõuete esitamine, olemasolevate programmide muutmine ja uuesti süsteemi testimine (Sommerville, 2011: 61). Seega nõuete muudatuste juhtimine (ingl *requirement change management* - *RCM*) ja varajane tarkvaranõuete muutuste ennetamine võib aidata kokku hoida kulusid, vähendada üldist arenduse aega ja suurendada projekti edukust (Bano jt, 2012). Nõuete analüüs on iga eduka infosüsteemi arendusprojekti alus, millest sõltub arendusprotsessi edu või ebaõnnestumine. Tõrgetel võib olla mitmeid erinevaid põhjuseid, kuid ebapiisav nõuete analüüs aitab suuresti kaasa tarkvaraprojekti ebaõnnestumisele (Hussain jt, 2016). Pidev muutus on reaalsus ja edukad arendusmeeskonnad peavad saama hakkama dünaamilises keskkonnas.

1.2. Muutuvate nõuete juhtimine

Muudatus (ingl *change*) on arendusprotsessile loomuomane omadus - keeruline on kõiki tarkvaranõudeid täpsustada kohe projekti alguses, sest asjaolud ja stsenaariumid võivad ajas muutuda (Jayatilleke, Lai, 2017). Tarkvaraarendus on dünaamiline protsess ning nõuete muutumine selles on vältimatu (Answer, Wen, Wang, 2019: 2), mistõttu tarkvaranõuete juhtimisel on oluline roll lõpplahenduse õigeaegsel ja eesmärgipärasel tarnimisel, sest selle puudumine võib tekitada täiendavaid riske ja kahjusid. Ka kõige rangemate piirangute ja fikseeritud tingimuste korral on projektid “elusad”, mis nõuavad muutusi. Samas väga fikseeritud projektide puhul võivad muutuvad nõuded aga väga ohtlikult mõjuda, kui neid piisavalt ei juhita (Grebennikov, 2018). Muutused tarkvaranõuetes ei ohusta ainult lõpplahenduse edukat tarnimist, vaid sellest sõltub ka võimalik lõpplahenduse kasutatavus ja väärtuslikkus (McGee, Grees, 2012). Tarkvara ja selle kontekst on

omavahel tihedalt seotud - süsteem võib põhjustada muutusi keskkonnas ja samal ajal võivad muutused keskkonnas mõjutada ka süsteemi funktsioneerimist (Pohl, 2010).

Iga nõude muudatuse elutsükkel saab alguse selle teadvustamisest ehk muudatustaotluse vajadusest, mis võib olla tingitud nii sisemistest kui ka välistest allikatest. Kuna nõuete muudatuste juhtimine on võrdlemisi toores teema, siis tihti ei ole see teadlikult juhitud. Igale päringule peaks järgnema mõjuanalüüs ja selle tulemusel muudatustaotluse valideerimine (Alsanad jt, 2019: 3), kusjuures on ülioluline arvestada kõikide mõjutatud ja sõltuvate nõuetega, millele järgneb muudatuse autoriseerimisteade ning muudatus juurutatakse. Kõik projekti väljundid värskendatakse sisseviidud muudatusega, nii konfiguratsiooni dokumendid kui jälgitavuse lingid (ingl *traceability links*) (Alsanad jt, 2019: 3). Kui taotlus lükatakse tagasi, siis teisisõnu on muudatuse taotlus tehtud teatavaks (ingl *notified*), kuid seda ei juurutata, kuid igal juhul peab olema kogu info salvestatud muudatuste andmebaasi (Alsanad jt, 2019: 3).

Nõuete muudatuste juhtimine kannatab tihtilugu ühtsete terminite ja arusaamade puuduses (Alsanad jt, 2019: 1), mis viitab sellele, et mitte alati pole kokku lepitud ühtseid sõnakasutuse põhimõtteid, millest tulenevalt esineb kommunikatsiooni vigu ja vääriti mõistmist. Uuringud (Alsanad jt, 2019) on tõestanud, et nõuete muudatuste juhtimist on võimalik toetada ühtsete ontoloogiate kasutamisega . **Nõuete muudatuste ontoloogia** (ingl *requirement change ontology* - *RCO*), mis teisisõnu on teadmuse kirjeldamiseks ja esitamiseks ühtlustatud mõistete hierarhiad ja seosed nende taaskasutamiseks. See ülesehituslik protsess koosneb kolmest sammust, mis ehitab ühtse teadmuse keskkonnas, et hõlbustada kommunikatsiooni kogu tarkvaraprojekti raames arendusmeeskonna ja sidusrühmade vahel:

1) **Valdkonna kohta teadmiste omandamine ehk määratlemine** (ingl *specification*), mis sisaldab endas selle eesmärki ja käsitlusala; üldist kirjeldust, kompetentsiküsimusi (vt Tabel 3) ning ontoloogia detailsust ja tüüpi (Alsanad jt, 2019: 3-4). Kompetentsiküsimused aitavad selgitada nõude muutumise olemust, et võimalikult vara suuta ette näha kogu muudatuse olemust.

Tabel 3. Nõuete muudatuste kompetentsi küsimused

KK	Kompetentsi küsimus
KK1	Miks nõue muutus?
KK2	Kes taotleb muudatust?
KK3	Kes on vastutav muudatuse juhtimise eest?
KK4	Mis peamiselt muutub?
KK5	Kuna muudatus on taotletud?
KK6	Kes otsustab muudatuse aktsepteerimise või tagasi lükkamise üle?
KK7	Mis on tegevused, kui muudatustaotlus aktsepteritakse või lükatakse tagasi?
KK8	Kuidas viiakse läbi läbirääkimise protsess?
KK9	Kuidas muudatus on kontrollitud ja kinnitatud?
KK10	Kus on muudatustaotlus salvestatud?

(Allikas: Alsanad jt, 2019: 3)

2) **Teadmuse organiseerimine ja struktureerimine** ehk kontseptualiseerimine (ingl *conceptualization*), mis sisaldab endas domeeni/valdkonna kontseptuaalse mudeli koostamist (läbi klasside, atribuutide ja seoste), mis loob aluse ontoloogia terminisõnastiku koostamiseks, hõlmates samal ajal ka muid üldistatud mõisteid ja spetsialiseeritud tehnikaid (Alsanad jt, 2019: 3-4). Lisaks sellele määratletakse iga kontseptsiooni juhtumid tabelisse, mis sisaldab endas juhtumi nimetust (ingl *the name of instance*); kontseptsiooni, kuhu see kuulub (ingl *the concept it belongs to*); atribuute ning nende väärtuseid (Alsanad jt, 2019: 3-4).
3) **Juurutamine** (ingl *implementation*) (Alsanad jt, 2019: 3-4).

Pidev koostöö ja -suhtlus sihtrühmade vahel tagavad, et muudatused, uued tunnused ja parandused saavad organiseeritud vastavalt huvirühma huvidele (Bigelow, 2019) ning arvestatud muudatuste kriitilisuse ja selle mõjuga projektile (Grebennikov, 2018). Äärmiselt oluline on koos kliendiga analüüsida ja arutada kõike - uurida soovitud muudatusi, modelleerida ja visualiseerida võimalikke muudatusi, võrrelda toote praegust ja kavandatud funktsionaalsust, hinnata tähtaegu ja eelarvet.

1.2.1. Muutuvate nõuetega toimetulek

Iga arendusprojekti eesmärk on töötada välja soovidele vastav lahendus, mis täidab maksimaalselt mõjusalt ja tõhusalt eesmärki. Tihti esineb olukordi, kus klient ei suuda esialgses nõuete spetsifikatsioonis kirjeldada kõiki soovitud nõudeid, mistõttu on oht, et arendajal puudub piisav sisend soovitud lahenduse väljatöötamiseks. Samuti võivad kliendid lõpplahenduse ajakohastamise

eesmärgil oma esialgseid ideid ümber mõelda, muuta kontseptsioone või muuta prioriteete. See võib olla häiriv eelkõige arendajale, aga on enamasti vältimatu, kuna sõltub rohkem meeskonna võimekusest tulla toime muudatustega kui muudatuse olemusest või mahust. Muutmistaotlus on standardprotseduur, mis võimaldab teha muudatusi esialgses nõuete määratluses, lisada funktsioone või vähendada funktsionaalsust (Grebennikov, 2018). Tarkvaraprojekti õnnestumine sõltub suuresti sellest, kuidas reageerime muutuvatesse nõuetesse (Bano jt, 2012: 1).

On ilmselge, et iga hea projekt vajab juhti, kes vastutab kogu protsessi arengu eest, organiseerib ressursse ja koordineerib muutusi - arendusprojekti puhul on selleks reeglina projektijuht. Selge rollijaotus ning kokkulepitud käsuliin toetab arendusprotsessi kulgu oluliselt. Kogu muudatuste haldus ja nende taotlemine peaks toimuma läbi projektijuhi, mitte üksikute arendajate, sest omavoliline suhtlemine võib viia dubleeriva tööni ja takistada tõhusat andmevahetust (Bigelow, 2019).

Järjest laiemalt on levinud **välearenduse** (ingl *agile development*) **põhimõtete rakendamine**, mida integreerides erinevate lähenemistega on võimalik tagada maksimaalne kliendi rahulolu läbi pideva koostöö, kiire reageerimise ning muutuvate oludega arvestamise. Nagu esimeses peatükis (ptk 1) on välja toodud, on muudatuste suhtes paindlikum agiilne metoodika, kuna muutuvate nõuete korral võib arendusprotsess koosneda väga paljudest iteratsioonidest võrreldes traditsioonilise arendusprotsessiga. Agiilne lähenemine aktsepteerib muudatusi ja juhib muutuvaid nõudeid "omaksvõtmise" (ingl *embracing*), mitte "kontrollimise" (ingl *controlling*) teel (Bano jt, 2012). Kindlasti toetab muutuvate nõuetega toimetulekut **pikaajalise planeerimise vähendamine**, et keskenduda rohkem reaalsele hetkevajadusele, kui plaani täitmisele. Prioritiseerimine ja planeerimine muudavad küll iteratsiooni protsessi *sprintimise* lihtsamaks, aga pidev nõuete muutmine võib plaane väga tugevalt muuta ning juhtida protsessi arengut vales suunas (Bigelow, 2019). Mõistlik on vähendada üksikdetailide planeerimisele kuluvat aega, et keskenduda iteratsiooni kõige kriitilisematele ülesannetele. Samuti toetab muudatustega toimetulekut **nõuete jagamine väikesteks kasutajalugudeks**, mis annab võimaluse lihtsamalt mõista nõude sisu ja võimalusi seostamiseks rohkem kui tavapärane nõuete dokumendi lugemine. Nõuete ühekaupa realiseerimine annab võimaluse uuenduste sissetoomiseks ka keset arendust, sest igal kasutajalool on omad nõuded, mida on võimalik eraldiseisvalt käsitleda (Sommerville, 2011: 80), et tagada eraldi iga nõude jälgitavus. Kuid siinkohal ei tohi unustada kasutajalugude täielikku kattumist - on keeruline otsustada, kas üks

lugu annab tõese pildi tegevuse üle, sest kogenud kasutajad on oma tööga sageli tuttavad, jättes selle kirjeldamisel olulisi asju katmata.

Samuti on muudatuste korral abiks kitsaskohtade **pidev ülevaatamine ja valmisolek vajadusel protsessi optimeerida** (Bigelow, 2019), sest arendusprotsessi õnnestumise huvides peab alati olema võimekus olla kaasas viimaste muudatuste ja arengusuundumustega, mida on vaja pidevalt ajakohastada. Riigikontrolli auditi raportis (2019) tuuakse välja, et kui arendusprojekti vahetulemusi ei seirata ega analüüsita, tarkvaraarenduse ulatuse kindlaksmääramine ja muutmine ei toimu läbimõeldult, arenduse teostatavust ei hinnata või ei looda koosvõimet, siis on kannatajaks projekti tulemusel loodav infosüsteem (Riigikontroll, 2019: 10). Hoolimata arendusmetoodikast - kas tegemist on agiilse või traditsioonilise lähenemisega - tuleb igakülgselt kasuks **ühtse muudatuste ontoloogia kasutamine**. Üks võimalikke viise on erinevate ontoloogiate omavaheline integreerimine, näiteks vastavuse analüüs erinevate mõistete vahel, et parendada kommunikatsiooni, tagada korrektsus ja ühtlustada muudatuste taotlemise protsessi kaasatud sidusrühmade omavahelist keelekasutust (Alsanad jt, 2019: 4). Kokkuvõttes taandub kõik **asjakohasele kommunikatsioonile**, mis aitab võimalike muutuvate nõuete põhjustega tegelda või neid isegi vältida (Alsanad jt, 2019: 2), sest tarkvara nõuete teemaline kommunikatsioon ei saa kunagi olla ühekordne ja ühesuunaline. Arendusprojektid nõuavad vestlusi reaalajas, et arendajad ja sidusrühmad saaksid nõudeid ja nende muudatusi arutada (Bigelow, 2019). Grebennikov (2018) toob oma artiklis välja, et parim lahendus on arendaja poolt antud üksikasjalikud hinnangud ning konkreetsed selgitused kliendile ja muidugi pidev kliendi harimine, et ta päriselt mõistaks, kuidas taotletud muudatused süsteemi mõjutavad ja miks. Samuti on ülioluline roll projektijuhil kommunikatsiooni juhtimise osas, et hoida pidevalt kõiki üheses infovoos - otsuse tegijaid kursis projekti arenguga, näidata kliendile seoseid funktsionaalsuste ja projekti arendusetappide vahel, ning juhtida tähelepanu, kui muudatuste tegemiseks on veel ruumi (Grebennikov, 2018). Selge ja häirevaba suhtluse toimimiseks on ülioluline leppida kommunikatsiooni põhimõtted kokku juba projekti alguses, et kõik osapooled seda üheselt mõistaksid. Muudatuste eesmärk on siiski rahuldada sidusrühmade vajadusi paindlikult ja ajakava raames (Bigelow, 2019).

Süsteemi prototüüpimine (ingl *system prototyping*) ehk uue kavandatud lahenduse mittetäielik esialgne teostamine prototüübina, mis on loodud kontseptsiooni või protsessi testimiseks (NSPCC Digital Team, 2018) on hea viis kontrollimaks kliendi nõudeid ja proovimaks disaini võimalusi (ingl

try out design options) (Sommerville, 2011: 62). See on kindlasti lahendus muutuste ennetamiseks, mis aitab prognoosida vajalikke ja võimalikke muudatusi ning samas aitab ka kliendil märgata tugevusi ja potentsiaalseid ohukohti. Näiteks nõuete spetsifikatsioonis kirjeldatud funktsioon võib tunduda kliendile arusaadav ja lihtne, kuid kui see on kombineeritud teiste funktsioonidega, võib kasutaja leida, et see esialgselt kirjeldatud vaade spetsifikatsioonis ei ole täielik. Prototüüpimine võimaldab süsteemi katsetada enne süsteemi valmimist ning seega täpsustada süsteemi nõudeid, mis tõenäoliselt võib vähendada muudatusettepanekute arvu pärast süsteemi tarnimist (Sommerville, 2011: 62). Samas on prototüüpimisel omad puudused. See võib kujuneda kohati liiga kalliks - võib juhtuda, et kulutatakse raha prototüübile, mis ei õnnestu, on liialt keeruline ning aeganõudev (NSPCC Digital Team, 2018). Lisaks puudustele on sellel aga ka hulgaliselt kasulikke omadusi: võimalus saada kiiresti tagasisidet, tuvastada lünkasid, hallata ootusi (Sommerville, 2011: 62). Kokkuvõttes võib prototüüpimine aidata raha hoopis kokku hoida, kuna mida varasemas faasis on võimalik tuvastada puudused, seda suurem tõenäosus on projekt lõpetada õnnestunult ning tarnida paindlik ja toimiv süsteem. Küll aga tuleb prototüüpimisega seotud eesmärgid kohe protsessi alguses kokku leppida: kui mitu prototüüpi, missuguse funktsiooni kohta ja kellele suunatud, sest iga prototüüpimise viimane etapp on prototüübi hindamine, kus potentsiaalsed kasutajad kasutavad süsteemi tavapärasel moel ning annavad selle kohta hinnangu.

Järkjärguline tarnimine (ingl *incremental delivery*) seisneb keskendumises loodavale väärtusele, et tarnida kliendile lisandväärtust samm-sammuliselt, kommenteerimiseks ja katsetamiseks, mis on testimise viisi poolest omane pigem agiilsele metoodikale. Järkjärgulise arendusprotsessi puhul puudub süsteemi spetsifikatsioon, mida väline testrühm saab testimiseks kasutada (Sommerville, 2011: 81).

Muutuste ettenägemine ja võimalik muutuste ennustamine tähendab, et tuleb suuta näha ette tarkvara ja disaini edaspidiseid muudatusi, et neid oleks lihtne rakendada (Sommerville, 2019: 81). Näiteks arendaja peab reeglina nägema ette tulevasi muutusi, lisades fikseeritud hinnale lisatasu, et vältida tarbetuid läbirääkimisi ning pakkuda paindlikkust (Grebennikov, 2018).

2. EESTI AVALIKU SEKTORI ARENDUSPRAKTIKAD

Moodne riigipidamine on andmete keskne ja riigi toimimiseks hallatakse andmeid väga suurtes kogustes (Riigi Infosüsteemi Amet). Riigi infosüsteemide haldussüsteemi (RIHA) andmetel on Eesti avalikus sektoris üle 2600 registreeritud infosüsteemi ja andmekogu, mis pakuvad erinevaid e-teenuseid nii elanikele kui ka ametnikele. Riigi infoühiskonna arengukava viimase eelarve prognoosi järgi peaks ajavahemikul 2014–2020 kuluma IKT-lahendustele kokku 223 miljonit eurot (Riigikontroll, 2019). Riigile pakuvad IKT arendamise ja haldamise teenuseid ministeeriumite haldusalas asuvad IT-asutused nagu Siseministeeriumi infotehnoloogia- ja arenduskeskus (SMIT), Registrite ja Infosüsteemide Keskus (RIK), Rahandusministeeriumi Infotehnoloogiakeskus (RMIT), Riigi Infosüsteemi Amet (RIA), Eesti Infotehnoloogia ja Telekommunikatsiooni Liit (ITL), Keskkonnaministeeriumi Infotehnoloogiakeskus (KEMIT) ning ministeeriumid ise, kuhu on koondatud avaliku sektori IKT teadmus. Peamiselt kuuluvad IT-asutuste tegevusalasse kasutajatoega, tarkvaraga- ning infotehnoloogilise taristu haldusega seotud teemad, kuid suuresti toimuvad avaliku sektori tarkvaraarendused riigi ja erasektori omavahelises koostöös.

Riigis on kehtestatud mitmeid nõudeid, näiteks IT-arhitektuuri raamistik (2007) ja infosüsteemide semantilise koosvõime raamistik (2007), et luua alus infosüsteemide koosvõimelisuseks ja luua võimekus ühtselt mõista vahetatava informatsiooni tähendust erinevates organisatsioonides. IKT arengut arvestades võib eeldada, et mõlemad nimetatud raamistikud ei pruugi enam olla täielikult asjakohased. Hiljem on veel välja antud riigi IT-arhitektide nõukogu poolt kehtestatud ristfunktsionaalsed nõuded, mida on kohustuslik järgida kõikide riigi infosüsteemide arendamisel. Muuhulgas on infosüsteemide arendamise perspektiivist oluline ka Infosüsteemide kolmeastmeline etalonturbe süsteem (edaspidi ISKE), mis on mõeldud andmekogude pidamisel kasutatavate infosüsteemide ja nendega seotud infovarade turvalisuse tagamiseks (ISKE, 2017). Lisaks kehtivad veel täiendavad toetusmeetme nõuded arendusprojektidele, mida arendatakse välisraha eest.

Üldiselt viiakse arendusprotsess ellu avaliku- ja erasektori koostöös ning kasutusel on mitmeid erinevaid arendusmetoodikaid, mida omavahel kombineerides saadakse sobiv lahendus - lisaks

koskmudelile iteratiivne arendusmudel, agiilne tarkvaraarendusmudel, *DevOps* (ingl *development and operations*) mudel jt (Riigikontroll, 2019). Samas Pardo ja Scholl (2002) toovad välja, et avaliku sektori arendusprojektide puhul on kõige levinumalt kasutusel jäik ja lineaarne järjestikune arendusprotsess, tuntud kui koskmudel, jättes tähelepanuta tehnilise arengu protsessi iteratiivse ja mitte-lineaarse olemuse, kuna keskendub lahenduse dokumenteerimisele enne selle väljatöötamist (Meso, Jain, 2006, lk 20). Selline lähenemine raskendab oluliselt mistahes muudatuste ellu rakendamist ning nagu ka mitmed näited (Riigikontroll, 2019) lähiminevikust viitavad, on mitmel juhul saanud saatuslikuks nõuete muutumine. Põhjuseid on mitmeid, aga peamiselt süvendab probleemi asjaolu, et infosüsteemi kvaliteeti hinnatakse liiga hilja - alles kogu projekti või projekti osa lõppedes, sest lõpptulemile planeeritud funktsionaalsuse saavutamiseks on vaja muudatusi teha palju varem.

Eestis on spetsiifiliste teemade puhul väga konkreetsed suurpakkujad, mis võivad oluliselt mõjutada arendusprojekte - näiteks üheltpoolt tekib neil ettevõtetel kindlasti teatav kogemus ja vilumus hangetes osalemisest, mistõttu on neil eelised võrreldes konkurentidega. Samuti võib spetsialiseerumine kitsale valdkonnale seda veel süvendada ning need ettevõtted kinnistuvad veelgi spektrisse, mistõttu on võimalik, et vähenevad ka valikud ja innovatiivsus, kuna suure tõenäosusega on ühe ettevõtte esindatavad väärtused, võtted ja lahendused potentsiaalselt ühevaatelised. Samas Riigihangete seaduse (edaspidi RHS) § 2 kohaselt on: /.../*seaduse eesmärk /.../ tagada hankija rahaliste vahendite läbipaistev, otstarbekas ja säästlik kasutamine, isikute võrdne kohtlemine ning konkurentsi efektiivne ärakasutamine riigihankel /.../. Riigihangete planeerimisel ja korraldamisel arvestatakse sotsiaalsete kaalutluste, innovatsiooni rakendamise ning keskkonnasäästlike lahendustega* /.../ (RHS). Siinkohal võib tekkida küsimus kuivõrd reaalne on isikute võrdne kohtlemine, konkurentsi efektiivne ärakasutamine või innovatsiooni rakendamine. Samuti on mainimist väärt ka tõsiasi, et mainitud suurpakkujatel on eeldused võimekuses - on tõenäoline, et väikepakkuja pakkumisdokumenti vormistavad inimesed põhitöö kõrvalt samal ajal, kui suurpakkujal võib selleks olla professionaalne strateegia ning meeskond, kes koostavad "professionaalsema pakkumise" ent kvaliteedi ja motivatsiooni näitajad võivad jääda varjatuks.

Riigikontrolli (2019) andmetel ebaõnnestusid auditeeritud üheksast tarkvaraarenduse projektist neli perioodil 2009 kuni 2018. Siin tekib küsimus, miks kipuvad arendusprojektid ebaõnnestuma? Riigikontroll (2019) soovitab siinkohal enne arenduse algust kirjeldada ja optimeerida põhitegevuse

protsessid, mis Karu (2019) sõnul kuuluvat ulme valdkonda, täiendades, et /.../ *eriti kui seda kombineerida tähelepanekuga, et "Arendusprojektide tõhusat elluviimist takistavad muudatused, mis tehakse arenduse ajal õigusaktidesse"*/.../. Sekundeerides Karu (2019) võib riigi perspektiivist ümbervaatamist vajada põhimõtteline muutus, mõistmaks ühiselt, et /.../ *kõnealused lahendused ei ole "IT-lahendused", vaid tervet organisatsiooni puudutavad lahendused, mille käigus muuhulgas ning pikema aja jooksul optimeeritakse ka põhitegevuse protsessid koostöös arendaja, peakasutaja, lõppkasutajate ning partneritega /.../.*

Riigikontrolli (2019) auditi aruandes toodi välja veel mitmeid tähelepanekuid avaliku sektori tarkvaraarenduste suunal, mis demonstreerib ulatuslikke ning põhimõttelisi probleeme Eesti avaliku sektori IT-süsteemide arendamise koordineerimisel. Ülesanne, kuidas riigi IT-süsteemide arendust olulisel määral paremuse poole liigutada, lasub MKM-il (Karu, 2019). Täpsemalt on MKM-i ülesanne läbi enda haldusalas asuva Riigi Infosüsteemi Ameti (RIA) panustada turvalisse andmevahetusse, milleks on tänaseks juba loodud X-tee; riigi infosüsteemide andmete tervik-ülevaatesse (RIHA); ühtsesse elektroonilise identiteedi kasutusse; riigi internetivõrgu toimimisse ja kaitsesse; e-valimiste korraldamisesse; riigiportaali eesti.ee tegevusse; ning ID-kaardi kasutajatoe töösse (Riigi Infosüsteemi Amet).

2.1. Uurimisprobleem ja uurimisküsimused

Magistritöö teema aktuaalsus seisneb probleemis, et infosüsteemide arendusprotsessi käigus ei toimu teadlikult muutuvate nõuete juhtimist, mida kinnitas ka Riigikontroll (2019) läbiviidud auditis avaliku sektori tarkvaraarenduse projektide juhtimise kohta. Muuhulgas ei mõelda infosüsteemi loomisel ja projekti idee tõstatamisel infosüsteemi hilisemale jätkusuutlikkusele, mistõttu ei vasta arendatud tarkvara kõikide kasutajarühmade vajadusele. Nagu töö esimesest osast (ptk 1) selgus, on tarkvaranõuete muutumine projekti vältel üks kõige sagedamini esinev väljakutse kogu arendusprotsessis. Sellest lähtuvalt olen tõstatanud järgmised uurimisküsimused:

1) Millised nõuded on kõige altimad muutuma?
2) Mis on tarkvaranõuete muutumise peamised põhjused?
3) Millised on avaliku sektori praktikad muutuvate tarkvaranõuete juhtimiseks? Milliseid protsesse ja tehnikaid selleks kasutatakse ning mis on peamised väljakutsed?

3. VALIM JA MEETOD

Käesolev peatükk annab ülevaate uurimistöö läbiviimiseks kasutatud valimist ja meetodist ning uurija reflektsioonist antud töö suhtes.

3.1. Valim

Andmete kogumiseks viisin läbi 5 poolstruktureeritud intervjuud ekspertidega perioodil 07.04.2020 kuni 24.05.2020 ning tegin kvalitatiivset sisuanalüüsi lähtuvalt uurimisküsimustest valikuliselt järgmiste dokumentide osas, mis juurdepääsupiirangute tõttu ei ole tööle lisatud:

- LA teenuste register ja teenuste-protsesside maatriks (edaspidi DOK 1);
- LA UAS protseduur ja protsessijoonis (edaspidi DOK 2);
- EV siseriikliku seadusloome muutmise eelnõu (edaspidi DOK 3);
- LAIS lähteülesande dokument koos lisadega (edaspidi DOK 4);
- Euroopa Regulatsioon 2019/947 ning muud kõrgetasemelised regulatiivsed raamistikud ja seisukohad (ingl *high-level regulatory framework and opinions*) (edaspidi DOK 5);
- enda protokollitud koosolekud, sealhulgas projekti juhtrühma koosolekud (2), töörühma koosolekud (4) ning üks-ühele vestlused konkreetsete teemade osas spetsialistidega (5) (edaspidi DOK 6).

Muutuste juhtimine on võrdlemisi uus valdkond, mistõttu olen valinud ka uurimismetoodika - uuendusliku valdkonna uurimiseks on soovituslik just kvalitatiivne uurimismeetod ringja ja spiraalse uurimisprotsessi, üksikjuhtumite uurimise ning sihipäraselt koostatud väikese valimi tõttu (Baur, 2019). See tähendab, et kvalitatiivne metoodika sobib peamiselt andmete struktureerimata olemuse jaoks, et saada kokku terviklik pilt teemast, kus toimub pidev tsükliline avastamine, kus iga tsükliga saab uurija uut informatsiooni ja teadmisi. Kõik intervjuud viisin läbi virtuaalselt videokõne vormis, kasutades selleks Skype'i ja Skype'i ärirakendust.

Antud töös on oluline väärtus ekspertarvamusel. Konkreetsed eksperdid olen valinud sihipärase valimi (ingl *purposive sample*) alusel ehk */.../ eksperdirollis uurija valib uuritavad, püüdes leida populatsiooni kõige tüüpilisemaid esindajaid sõltuvalt uurimiseesmärgist /.../* (Rämmer, 2014). Antud uurimuses pean ekspertideks eelkõige isikuid, kes on avaliku sektori arendusprotsessis vastutavatel ning olulistel positsioonidel. Näiteks projektijuht, peakasutaja ning arendaja esindaja(d), kelle ekspertiis seisneb peamiselt kogemustel erinevate avaliku sektori arendusprotsessidega. Antud teema, milleks on muutuvate nõuete juhtimine ja nendega toimetulek eelkõige avalikus sektoris, on väga uuenduslik ja ajas muutuv valdkond, mistõttu puuduvad täpsed kirjutatud praktikad ja ekspertarvamuse kogumine on ainuõige võimalus asjakohase info saamiseks. Ekspertarvamuse kogumiseks olen läbi viinud poolstruktureeritud intervjuud valdkonna ekspertidega. Eksperdid tutvustavad erinevaid kasutusel olevaid praktikaid nii avaliku aga ka erasektori perspektiivist, et koguda informatsiooni tarkvaraarenduse protsessi raames muutuvate nõuete juhtimise kohta. Intervjuud on läbi viidud järgmiste isikutega:

- Avaliku sektori ekspert, MKM projektijuht (edaspidi INT 1);
- Erasektori ekspert, tegevjuht ja arendaja (edaspidi INT 2), kellel on märkimisväärne kogemus alates eelmisest sajandist erinevate tarkvaraarendusprojektidega nii avalikus- kui ka erasektoris;
- Avaliku sektori ekspert, MKM esindaja (edaspidi INT 3), kellel on pikaaegne Eesti avaliku sektori kogemus;
- Avaliku sektori ekspert, (edaspidi INT 4), kellel on tänaseks märkimisväärselt kogemusi nii rahvusvahelisest IT suurorganisatsioonist, *StartUp* maastikult kui ka Eesti avalikust sektorist.
- Erasektori ekspert, arendaja (edaspidi INT 5), kellel on pikaajaline kogemus avaliku sektori tarkvaraarenduses.

3.2. Meetod

Ekspertarvamuse kogumiseks viisin läbi poolstruktureeritud intervjuud. Intervjuude läbiviimise toetamiseks koostasin intervjuu kava (vt Lisa 1), mis sisaldas endas teemasid ning küsimuste valikut, mida intervjuu läbiviimise ajal vastavalt vajadusele kombineerisin - ühest küljest oli teada intervjuu eesmärk ja vajadus, kuid jätsin võimaluse küsimuste järjekorra muutmiseks vastavalt teemadele ja

olukorrale. Soov oli pakkuda intervjueeritavale vabadust teemade kulgemise osas loomulikult ning küsida vajadusel täpsustavaid küsimusi. Kõik intervjuud on läbi viidud individuaalselt, et igal intervjueeritaval oleks võimalik privaatselt ning omas tempos ekspertarvamust jagada. Kõik intervjuud salvestasin transkribeerimise eesmärgil. Intervjuu kava jagunes neljaks osaks. Esimeses sissejuhatavas osas annab intervjueeritav loa tulemuste kasutamiseks, ülevaate valdkonnas tegutsemise staaži ja erinevate praktikate osas üldiselt. Teises osas on küsimused muutuvate nõuete olemuse kohta üldiselt - missugused nõuded on altimad muutuma ning muutumise peamised põhjused. Kolmandas osas käsitleme lähemalt praktikaid muutuste juhtimiseks - teadlikud ja mitteteadlikud praktikad. Neljas ehk viimane plokk keskendub üldiselt tarkvaraarendamise põhimõtetele, et uurida ekspertarvamust üldisemalt tarkvara arendamise põhimõtete kohta avalikus sektoris. Viimane osa oli rohkem sekundaarne, kuid oluline tausta avamiseks ja intervjuu kokkuvõtmise perspektiivist.

Antud magistritöös olen esitanud tsitaadid intervjueeritavate kogemustest, arvamustest ja teadmistest. Transkribeeritud intervjuusid haldasin veebirakenduses QCAmap, mis on kvalitatiivse sisuanalüüsi tarkvara teadusprojektide süsteemseks analüüsimiseks, et vähendada paljusõnalist teksti mahult paremini hallatavateks andmeteks, tuvastada mustreid ja saada terviklik ülevaade. Intervjuude kodeerimisel kasutasin suunatud kodeerimist, keskendudes magistritöö uurimisküsimustele. Kodeerimise tulemusel moodustasin kategooriad, et koondada sisult lähedased ning pealkirjastasin koondnimetused, mille tulemusel valmis intervjuude analüüs (vt ptk 3.3). Seejärel kõrvutasin ekspertintervjuude analüüsitulemused isiklike kogemuste ning teoreetiliste lähtepunktidega.

3.3. Uurija reflektsioon

Kvalitatiivse uurimismetoodika puhul tuleb arvestada uurija võimaliku mõjuga, sest */.../ kvalitatiivne sisuanalüüs loob uurijale võimaluse valikulise tõendusmaterjali kogumiseks, mis toimub sageli mitteteadlikult, uurijale meelepäraste hüpoteeside kinnitamiseks /.../* (Kalmus, Masso, Linno, 2015). Käesoleva töö puhul võivad olla minu kui autori võimalikud eelarvamused teemapüstituse, andmekogumise ja analüüsi osas peamiselt tellija esindajana ehk arendatava süsteemi peakasutaja rollist lähtuvalt. Läbivalt terve intervjuu vältel pidin teadvustama oma rolli ehk missugune on minu ja intervjueeritava omavaheline suhe omavahelises koostöös - kas antud hetkel rääkis intervjueeritav minule kui uurijale, kolleegile või hoopis tellijale. Mitme intervjueeritavaga esines olukordi, kus pidin täiendavalt selgitama, et minu magistritöö ja uurimuse eesmärk on keskenduda erapooletult üleüldiselt

valdkonna kaardistamisele ning protsessi võimalikule parendamisele kui, et lahata teemat negatiivses tähenduses.

Tulenevalt eriolukorrast viisin kõik intervjuud läbi videosilla vahendusel, mis abistas ühelt küll kohtumiseks sobiva aja leidmisel, kuid virtuaalne ei ole reaalne. Intervjuude käigus võis jääda varjatuks teatud emotsioone ja informatsiooni, mida pole võimalik sõnadega väljendada. Samuti kadus ühendusprobleemide tõttu mitmeid minuteid väärtuslikku intervjuuks planeeritud aega. Uuritava valdkonna uudsusest lähtuvalt oli oluline koguda fakte, mistõttu lähenesin struktureeritult teemade lõikes lähtuvalt uurimisküsimuste järjekorrast: esmalt muutuvatest tarkvaranõuetest ja nende muutumise põhjustest; ning seejärel nende juhtimisest avaliku sektori näitel. Viimaseks plokiks jätsin üldised küsimused arendusmetoodika kohta, et koguda taustainformatsiooni üldiselt.

Minu teadlikkus kattuvatest rollidest ning paralleelne eneseanalüüs on olnud peamine võimalus rollikonflikti vältimiseks ning tulemuste kallutamise vähendamiseks. Selle tarbeks küsisin võimalikult avatud küsimusi ning andsin ruumi intervjueeritavale vastata oma sõnadega, vältisin oma kommentaaride ja kogemuse jagamist, et maksimaalselt suurendada intervjueeritava mõtete esitamist. Samuti andmeanalüüsi tehes keskendusin eelkõige uurimisküsimustele ning kodeerimisel ja grupeerimisel lähtusin intervjueeritavate vastustest, mitte sellest mida isiklikult ootasin ja arvasin.

4. TULEMUSED

Käesolevas neljandas peatükis annan ülevaate magistritöö analüüsi tulemustest püstitatud uurimisküsimustest lähtuvalt. Esimeses alapeatükis (ptk 4.1) toon välja ekspertide arvamused, mida olen ilmestanud ka otse tsitaatidega. Teises alapeatükis (ptk 4.2) kirjeldan näidisjuhtumit LAst.

4.1. Muutuvad nõuded, peamised põhjused ja avaliku sektori praktikad

Järgnevalt analüüsin töös püstitatud uurimisküsimusi ekspertintervjuude põhjal: muutuvate nõuete olemust, nõuete muutumise peamisi põhjuseid ning erinevaid praktikaid avaliku sektori tarkvaraarenduste pinnalt.

4.1.1. Muutuvad nõuded

Käesoleva uuringu jaoks läbi viidud intervjuud kinnitasid sarnaselt teooriale (Jayatilleke, Lai, 2017), et nõuete muutumine arendusprotsessi vältel on paratamatu, kuna elame pidevalt muutuvas keskkonnas ning eesmärk ei ole muutusi ära hoida, vaid sobivaid meetodeid kasutades leida parimad võimalused lahenduste elluviimiseks.

/.../ Siin on muidugi põhimõtteline küsimus - kas me tahame muudatusi ära hoida? /.../ ...muudatused on loomulik osa - me peame suutma need ellu viia /.../ (INT 3)

Samas tõid mõned intervjueeritavad (INT 4) välja, et nõuete muutumise kõige suurem probleem võib olla illusioon sellest, et nõuded ei muutu või hoopis see, kui püüda tekitada olukord, kus nõuded ei muutu - näiteks jäädakse kinni lähteülesande kirjeldusse ning eitatakse muudatusi. Pigem on eesmärk toimiva lahenduse arendamine, kus rõhk on paindliku süsteemi arendamisel, mis oleks maksimaalselt kasutajasõbralik ning efektiivne.

/.../...see on natukene selline idamaine zen-filosoofia, et sa aktsepteerid kasutaja nõuete muutumist ja alles siis hakkad mõtlema rakendamisele /.../ (INT 4)

Kui kliendi nõue on töö käigus põhjendatud, siis viiakse see tavaliselt ikka ellu (INT 2). Teatud faasis võib olla vajalik teha otsus, et mõni muudatus tuleb viia jätkuprojekti mastaapi, aga kindlasti ei tähenda aktsepteerimine seda, et kõik muudatused vastu võetakse (INT 4).

Intervjuudest selgus, et kõige altimad on muutuma arendusprotsessi käigus just funktsionaalsed nõuded, mis kirjeldavad nõudeid funktsioonide täitmiseks, määratledes täpsemalt süsteemi omadused, näiteks kuidas tuleb mingit kooskõlastust teha või kuidas peaks väline kooskõlastaja toimima. Infosüsteem on toetav vahend organisatsioonile tööprotsesside automatiseerimiseks ja efektiivsemaks muutmiseks - kui organisatsiooni protsessid või tegevusala muutub, on see ilmselge sisend funktsionaalsete nõuete muutumiseks (INT 1).

/.../...Mittefunktsionaalsed nõuded eriti ei muutu - need on IT paika pannud ja nüüd oleme raiunud need igasugustesse juhenditesse ja lasknud ministril, kantsleril käe alla panna /.../ (INT 3)

Nii nagu iga arendusprojekt on omamoodi, on igal muudatusel omad spetsiifilised nüansid ja põhjused. Ometi saab kokkuvõtvalt öelda, et ekspertide kogemusele toetudes on Eesti avaliku sektori tarkvaraprojektides kõige altimad siiski muutuma funktsionaalsed nõuded, täpsemalt ärinõuded, väliste liidestega seotud nõuded (sealhulgas kolmas osapool ja raportid) ning juriidilised ja regulatiivsed nõuded.

4.1.2. Nõuete muutumise peamised põhjused

Intervjuudest selgus mitmeid erinevaid nõuete muutumise põhjuseid. Olen need jaotanud vastavalt kategooriatesse, mida järgmiselt lähemalt tutvustan: inimfaktorist põhjustatud muutused, tööprotsesside ja organisatsiooniga seotud muutused, keskkonnaga seotud muutused ning tehnoloogilised ja arendusprotsessiga seotud muudatused.

Ühe peamise põhjusena said kõikides intervjuudes kajastust inimfaktorist tulenevad muudatused. Näiteks kõik intervjueeritavad märkisid inimeste harjumuslikku käitumist. Küsimusele, kuidas midagi parendada, on inimlik parendusettepanek stiilis “See nupp võiks olla natuke paremal pool ja teist värvi” (INT 4), mis iseloomustab olukorda, kus inimesed eelistavad näha kõike iseendale tuttavast perspektiivist. Seetõttu uued ideed ning mõtted võivad olla harjumatud.

/.../...Klassikaline näide on Fordi tsitaat, kui küsida kasutajalt, et mida nad vajavad siis oleks vastus, et kiiremaid hobuseid. Teatud hulga otsuseid pead sa eelnevalt kasutaja eest ära tegema. Eriti, kui Sa teed väga radikaalset muudatust /.../ (INT 4)

Intervjuud kinnitavad, et üheks oluliseks muutusi tingivaks põhjuseks on ka nõuete ununemine inimeste poolt, inimesed ei saa aru või nõuded jäävad hoopis tähelepanuta, mistõttu puudub füüsiline töövoo kirjeldus. Ühe intervjueeritava kogemusele tuginedes kiputakse analüüsidokumenti liialt pealiskaudselt lugema - nad lappavad kõik leheküljed kohusetundlikult läbi, aga kas nad sellest ka aru saavad? Testid näitavad, et kui inimeselt küsida, et mida sa eelmiselt leheküljelt lugesid või mida see diagramm tähendas, siis tuleb vastuseks midagi sellist, et "a no jah, ma neid diagramme ei vaadanud" (INT 4).

/.../... nõuded olid olemas aga keegi ei teadnud neist. Alles pärast esimest installatsioon tuli see ilmsiks. Püüame võimalikult vara " jalad alla saada", et klient saaks midagi näppida. See on juhul, kui tegemist on enam-vähem teadliku kliendiga /.../ (INT 2)

Lisaks märkisid eksperdid (INT 2 ja 3) inimfaktoriga seotud muutuseks ka arendusprojekti raames teadmuse kadumist, mis võib olla tingitud inimeste vahetumisega positsioonidel. Iga arendusprojekti perspektiivis olulise võtmeinimese vahetumine mõjutab projekti õnnestumist oluliselt (INT 3).

/.../...Näiteks poole arenduse pealt on tellija osakonnajuhataja vahetunud, tulnud uus ja öelnud, et see mida arendate on jama ja tuleb teha teisiti /.../ (INT 3)

Intervjuudest selgus, et inimeste nõrk protsessiteadlikkus ehk puuduv arusaamine protsesside juhtimise metoodikast võib tekitada olukorra, kus analüüs on tehniliselt õige, aga tegelikult vale (INT 2). Samas selgus, et enamasti on detailanalüüs tehtud avaliku sektori projektide puhul peamiselt arendaja analüütikute poolt, kelle teadlikkus protsessi metoodikast on ilmselgelt parem kui tellijal, kuid seevastu jällegi valdkonna tundmine tellijast nõrgem. Tellija esindajad ei ole harjunud lugema protsessidiagrammi, mida ei saa neile ette heita, sest miks disainer teeb kasutajatele sellist tööriista, mida ta pole harjunud kasutama? (INT 4) Siinkohal on paslik märkida ka eksperdi (INT 5) arvamus sellest, miks ei ole avaliku sektori tellija võimekas, teadlik ja tark. Ekspert (INT 5) toob analoogse näite ehitusvaldkonnast, sest omakasupüüdlik arendaja võib suunata tellijat, kellel puudub protsessi- ja IT-teadlikkus, tegema arendajale kasulikke valikuid.

/.../ näiteks ehitusvaldkonnas on ehitusjärelevalve, kes on kliendi huvide eest väljas /... / eesmärgiga vähendada riske ning kes teostavad tehnilist kontrolli /.../ (INT 5).

Inimfaktorist põhjustatud muutuseks saab lugeda ka kliendi arenemist arenduse raames (INT 2) - klient saab reeglina IT-teadlikumaks ning suudab juba peagi pärast arenduse käimalükkamist küsida ja näha rohkem funktsionaalsusi. Aga areneb ka arendaja, kes saab teadlikumaks kliendi valdkonnast (INT 2). Intervjuudest selgus, et inimeste arenguhüpet toetab tõsiasi, et arenduse käigus küsivad nii projektijuht, arendaja või juriidilise taustaga inimesed selliseid küsimusi, millele tellija ei ole harjunud mõtlema, ning tavaliselt vastavad, et *"oi, selle peale ma polegi mõelnud"* - kõik sellised teemad tulebki üheskoos püüda lahendada, mis on oluline ühtsele arusaamisele ning annab olulise panuse lõpptoote õnnestumiseks (INT 4).

/.../ ...alguses teeb ta kohutavalt käsitööd ning tal pole aimugi, kuidas seda vähendada - tihti on selleks lausa palgatud põhikohaga inimene. Meie, kes me oleme samm või kaks sellest mõtteviisist eespool püüame kliendile näidata, et neid asju saab teha ka nii /.../ kui klient on süsteemi juba natuke näppinud siis ta ütleb, et ta tahaks sama loogikat veel sinna ja sinna ka /.../ kliendis toimub arenguhüpe /.../ (INT 2)

Üheks oluliseks intervjueeritavate poolt välja toodud muudatuseks on ka organisatsiooni ja selle tööprotsessidega seotud muutused, mis võivad samuti tingida funktsionaalsete nõuete muudatusi olles suuresti seotud ka inimfaktoriga. Nagu tõi välja INT 4, isegi kui protsessid on nõuetekohaselt kaardistatud ja dokumenteeritud, siis veelgi olulisem on nende optimeerimine, efektiivsemaks muutmine ja lihtsustamine - mis unustatakse tihti ära.

/.../ ...tihti unustatakse ära üks äärmiselt oluline samm - protsesside lihtsustamine /.../ (INT4)

Seega võib intervjueeritavate sõnul tekkida olukord, kus analüüs on suurepärane, aga päris elus on vana taagaga seltskond, kellel puudub valmisolek uuendusteks ja kes näevad probleemi selles, et ei saa enam vana käsitööd teha ehk vanas süsteemis on kasutajad harjunud muutma teatud andmeid, mida uues süsteemis enam käsitsi ja korduvalt muuta ei saa ning seetõttu ei ole nad nõus lahendust vastu võtma (INT 2). Samas arendaja esindaja (INT 5) toob välja, et see on peamiselt organisatsiooni väärtuste küsimus - kui organisatsioonikultuur on paigas ja valdav töömeetod ei ole salajane vigade parandamine, siis see ei tohiks probleem olla.

/.../ ...kui valdav töömeetod ja organisatsioonikultuur on salajane praagi varjamine, siis organisatsiooni käitumuslik probleem võib olla hoopis suurem ja sügavam ning sellega peab tegelema väljaspool arendusprojekti käsitlusala /.../ (INT 5)

Keskkonnaga seotud muudatuse põhjuste osas tõid intervjueeritavad välja enim nimetatud avaliku sektori piiranguid, näiteks eelkõige rahastus, mis on seotud SF nõuete ning hankepoliitikaga. Ekspertide kogemusel võib selleks hetkeks, kui arendus pihta hakkab, esialgne idee olla juba aegunud ning arenduse valmimise tulemuseks võib olla seega toode, mis vastab kolm aastat tagasi seatud nõuetele, mitte käesoleva hetke tegelikele vajadustele. Mitu intervjueeritavat viitasid, et see võib olla seotud klient puuduva suutlikkusega oma lõppvajadust väljendada kohe alguses ning see tingib suure tõenäosusega muutusi arendusprotsessi vältel. See omakorda tingib põhjusi seadusemuudatustest - eriti sellised seadusemuudatused, mis jõustuvad arenduse lõppfaasis ning mõjutavad arendatavat süsteemi oluliselt (INT 3). Intervjueeritavate sõnul ei jõustu seadus ootamatult, seda valmistatakse ette ja paralleelselt sellele tehakse IT-arendus, mis omakorda seab IT-le väga suure vastutuse.

/.../ Näide ehitisregistrist - olime süsteemi arendamas ja teinud juba mingid asjad valmis ja siis tuli välja, et poliitilist kokkulepet õigusakti osas ei saavutatud ning mingid olulised asjad jäid kinnitamata, mistõttu tuli süsteemis palju asju ümber teha /.../ (INT 3)

Näiteks intervjuust (INT 3) selgus, et lisaks tavapärasele projektijuhtimise praktikale, kus luuakse nii töö- kui ka juhtrühm, moodustatakse ka juriidiline töörühm. Konkreetse eksperdi kogemustel on väga tõenäoline, et juba eelnõu faasis ringleb parandus- /muudatusettepanekuid - ei arendaja esindajal ega ka peakasutajal ei ole tõenäoliselt aega ega vastavat pädevust, et juriidilisi teemasid mõista.

/.../ Juriidiline töörühm annab tagasisidet, oskab juriidilises keeles mõtestada peakasutaja ideid ning oskab näha, kuidas konkreetset vajadust seadusesse kirjutada ning läbi suruda /.../ (INT 3).

Arendusprotsessi ja selle juhtimisega seotud põhjuste osas märkisid eksperdid samuti mitmeid teemasid. Seoses sellega tõid eksperdid välja peamiselt, et tarkavaranõuete muutused mõjutavad kogu projekti, sh ajagraafikut, eelarvet, tegevuskava, kvaliteeti ja eesmärke mistõttu tekitavad enam raskusi just avalikus ja mitte erasektoris, kus on peamiselt on kasutusel agiilne metoodika (INT 5), mistõttu muudatuste integreerimine arendusprotsessi rohkem loomulik protsess (INT 3). Ekspertide (INT 1 ja 3) arvamusel on see ilmselt tingitud peamiselt rahastusallikate erinevusest, kuna avaliku sektori arendusprojekte rahastatakse peamiselt SF toetustest - Struktuuritoetused aitavad tasakaalustada ja

ühtlustada Euroopa Liidu (edaspidi EL) liikmesriikide arengut ning tõsta seeläbi EL-i kui tervikliku majanduspiirkonna konkurentsivõimet maailmaturul (Struktuurifondid).

/.../ Suurim takistus on see, et riigil on arenduste jaoks vähe raha. Need vähesed projektid, mis õnnestub riigieelarvest rahastada, siis seal saab kasutada pigem paindlikku protsessi. Aga enamus projektide rahastusest tuleb SF-st /.../ (INT 3)

Analüüsist selgus ka see, et SF rahastuse taotlemisel on väga jäik süsteem - projektiga alustades peab kõik hankelepingud projekti alguses korraga esitama, mis tähendab, et seda, mida oled lubanud, pead ka valmis tegema (INT 3). Võimalus on taotleda rahastust SF-st ka jätkuprojektide, kuid toimetulemiseks SF ebaloomuliku ja ajamahuka tingimustikuga on vajalik oma eelarve paralleelne olemasolu, et oleks võimalik kasutada lisaressursse - nt projekti lõpetamiseks (INT 3). Näiteks tuli välja ühest intervjuust (INT 2) kolmandate osapoolte kaasamise vajadus - kui palju neid on, mis on süsteemi sõltuvus nendest välistest esindajatest ja vastupidi. Ühe eksperdi (INT 3) kogemusel on ka hulk muutusi olnud tingitud just kolmanda osapoole nõuetest, mille all on mõistetud raporteid ja aruandeid, näiteks partnerorganisatsioonid, kes on huvitatud teatud laadi statistikast, avalikus sektoris näiteks ministeeriumid või EL jt institutsioonid.

/.../ ...ehk oled tähele pannud lähteülesandeid, kus on loetletud kilomeetreid punkte ja ainult väike punkt pühendatud osale "statistika"/.../ (INT 2)

Eksperdid tõid veel välja mitmeid arendusprotsessist tingitud muudatusi, mis on seotud arendusmetoodika valiku ja nende kombineerimisega, projektijuhtimise, kommunikatsiooni ja koostööga. Kokkuvõtvalt võib öelda, et muutusi tingivaid põhjusi on väga palju ning kõiki ette näha ei ole võimalik. Peamiselt selgus intervjuudest, et nõuete muutustega toimetulemine sõltub suuresti projektijuhi kogemustest, võimekusest ja ettenägemise oskusest. Järgmises alapeatükis (ptk 4.1.3) toon ülevaate ekspertarvamusest lähtuvalt avaliku sektori praktikatest muutuvate nõuetega toime tulemisel.

4.1.3. Avaliku sektori praktikad muutuvate tarkvaranõuete juhtimiseks

Iga arendusprojekti planeerimine algab suurel määral sobiva metoodika valikuga. Intervjuud ekspertidega kinnitavad, et Eesti avalikus sektoris on kasutusel valik erinevaid arendusmetoodikaid,

mida komplekteerides leitakse igale projektile just see sobiv metoodika. Samuti joonistus välja, et erinevatest piirangutest tulenevalt (hankelepingud; SF rahastusega kaasnevad nõuded) on kasutusel peamiselt kohandatud projektipõhine arendusmudel - rakendame sega-metoodikat - kolmveerandkosk, mis on tänases SF õigusruumis kõige sobivam valik (INT 3).

/.../ ...Samas, kui sa tahad, et vähegi suurem projekt tooks sulle vähegi mõistliku tulemuse siis waterfalli põhimõtetel pole lihtsalt seda mõtet teha /.../ (INT 4)

Ometi selgub intervjuudest, et avalikus sektoris on kogemusi agiilsete metoodikate rakendamise osas. Näiteks on proovitud hankelepingu raames osta töötunde - esmalt seati projekti lõppeesmärk, mis jagati paarinädalasteks etappideks, millede vastuvõtmise alusel tasuti arendajale etapiliselt.

/.../ Agiilne ei tähenda ju seda, et lihtsalt trambid ringi ja vaatad, mis juhtub - sul on ikka üldine suund teada, aga ka see mis seal üldises suunas on muutlik. Sul peab olema võimalik igal hetkel teha suuremaid-väiksemaid pöördeid /.../ (INT 4)

Teatud puhkudel, peamiselt erasektoris, võib olla see intervjueeritavate arvates mõistlik, kuid vaid juhul, et lepingus ei ole määratud eelnevalt lõpptulemuse täpsed kriteeriumid (INT 3). Analüüsist selgub, et siin on vastuolu RHS-iga, mis nõuab põhjalikku süsteemi kirjeldust, et ühtsetel alustel valida sobiv pakkuja (INT 3). Samas toob ekspert (INT 3) näite, kus sellise töötundide ostmise puhul on mitmeid negatiivseid juhtumeid, kus töötunnid on küll kulutatud, kuid reaalne lõpptulemus puudub. Ta lisab, et töötundide ostmisel peab olema suutlikkus kogu aeg tööülesandeid ette anda, samal ajal arendaja, kelle käest töötunde ostetakse, ei vastuta millegi eest - selles suhtes on see meetod ohtlik (INT 3).

/.../...selline naljakas juhus, et oli vaja teha riigihange, et parim pakkuja välja selgitada. Hankes oli seatud, et kõige madalama esimese etapi maksumuse esitanud pakkuja saab kogu projekti endale. See hange lõppes nii, et parim pakkuja pakkus negatiivset hinda - algul pakkus hinnaks null eurot, siis kümne minuti pärast tegi lausa negatiivse hinnaga pakkumise. Asi läks jaburaks, hange tühistati ja tehti klassikaline hange /.../ (INT 3)

Enne lepingu sõlmimist on vajalik kindlasti leida sobiv ja usaldusväärne partner. Intervjuust (INT 3) selgus, et usaldusväärsuse testimiseks on katsetatud erinevaid metoodikaid - meeskonna nimekirjade küsimine, ülevaade tehtud töödest ja lepingutest või proovitöö. Intervjueeritavate kogemusel on RHS

läinud iga muutmisega lõdvemaks ja pakkujal on lubatud võtta oma meeskonda suvalisi inimesi, mistõttu meeskonna liikmete nimekirjal ei ole tähtsust, kuna on võimatu kontrollida, kes tegelikkuses koodi kirjutas (INT 2). Samuti puudub garantii hinnates vaid arendaja varasemaid kogemusi - ühelt poolt näitab kogemus ettevõtte kompetentsi, aga samal ajal võib inimene, kes eelmist projekti edukalt juurutas või arendas, olla ettevõttest lahkunud (INT 3). Proovitöö on kõige kindlam variant, millega kaasneb päring ajakava osas, näiteks on olnud ettevõtteid, kes pakuvad küll odavat hinda, kuid kui ajanõuet juurde ei pane, siis on reaalne, et see odav töö valmib alles nelja aasta pärast (INT 3).

/.../ Proovitöö on indikaator - kes kukub läbi proovitööga ei saa reaalselt ka hakkama, sest päriselus on see veel keerulisem /.../ (INT 3)

Samuti tuli intervjuust välja üks eeskujulik praktika, et hankele lisatakse vähemalt esimese etapi lähteülesanne, et pakkujal oleks näha, millega tuleb hakata kohe tegelema.

/.../ näiteks on proovitöö kõrval oluline ka veel see, et lähteülesandes oleks kirjeldus esimese etapi ülesandest, et pakkuja näeks kohe, millega tuleb tal esimesena tegelema hakata /.../ samuti küsime töö valmimise aega, sest on firmasid olnud, kes pakuvad küll odavalt, aga kui ajanõuet juurde ei pane siis selgub, et alles nelja aasta pärast saab see odav töö valmis /.../ (INT 3)

Arendusprojekti raamleping sõlmitakse enne projekti algust, tavapäraselt perioodiga 3-4 aastat (INT 3). Kehtib põhimõte, et see, kes arendab valmis SF-ist taotletud rahastusega projekti, teeb ka jätkuarendused (INT 3). Otseselt muudatuste juhtimise kohaseid punkte raamlepingus ei kajastata, kuid on kasutusel samaväärsuse mõiste (INT 4) - ehk paindlikkus nõuete vahetamise osas, kui tegemist on analoogse või hinnanguliselt võrdse kuluga muudatuse sissetoomisel, et maandada riski, kui mõni oluline nõue on esialgu jäänud tähelepanuta. Lisaks hankelepingule on kasutusel ka koostöö kokkulepete sõlmimine väliste oluliste osapooltega - näiteks mõni teine amet, kes klienditeekonna lihtsustamise eesmärgil kooskõlastab mingit osa menetlusest (INT 3). Kokkuleppe võib allkirjastada peadirektor või lausa kantsler, aga praktika näitab, et ka allkirjastatud koostöökokkulepe võib ebaõnnestuda (INT 3).

/.../ Kõige kindlam on, kui suudad oma eelarves vajaliku raha leida…. Sellisel juhul on tavaliselt õnnestunud. Oleme ka nii teinud, et partneri arendaja arendab, aga meie saadame garantiikirja tasumise kohta /.../ (INT 3)

On illusioon, et hankija suudab kirjeldada projekti käsitlusala täielikult kohe projekti alguses (INT 4). Pärast lepingu sõlmimist saab alustada arendustööga. Intervjuud kinnitasid kommunikatsiooni tähtsust läbi projekti - pidev suhtlus ning regulaarsed koosolekud on eduka ja eesmärgipärase lõpptoote valmimise edutegur, et kõik vajalikud muudatused saaksid rakendatud. Oluline vastutus lasub intervjueeritavate arvates kindlasti projektijuhil, peakasutajal, aga ka kõigil töörühma liikmetel. Projekti juhtimine sõltub peamiselt projektijuhi omadustest - pädevus, kogemused jne (INT 2). Iga projekt on oma projektijuhi moodi (INT 4) ning kogu muudatuste haldus peaks olema hoitud projektijuhi käes (INT 3). Samas toob üks intervjueeritud ekspertidest (INT 1) välja, et tihti on arendajal detailsed küsimused, mistõttu on kiirem ja bürokraatia-vaesem lahendus suhelda otse sisuspetsialistiga, minnes mööda projektijuhist, mis aga tekitab olukorra, kus hakkab puhta "telefonimäng". Täpne projekti skoop ja selle mõistmine vähendab kindlasti survet projektijuhile (INT 4) ning toetavaks teguriks on ka põhimõtted etapilisuse rakendamiseks (INT 3).

/.../ Ütleme, et jagame hankelepingu kolmeks, mille omakorda veel kahenädalasteks minitööülesanneteks, mida töörühm läbi töötab. Aga me ei saa teha klassikalist Scrummi, et võtame aktiga vastu ja maksame ära... /.../ (INT 3).

Intervjuust (INT 2) selgus, et arendaja, tehes tundmatu järgu arendustöid, mõtleb kliendiga kaasa ja temaga arutatakse läbi sisulisi probleeme eesmärgiga luua sisulisi lahendusi, kuid tegelikult ei ole ärilahenduse läbi mõtestamine arendaja rida - arendaja toetab seda, kuid põhiline motivatsioon ja pühendumine peab olema tellija poolt.

/.../ Ärilahenduste läbi mõtestamine ei ole tegelikult absoluutselt meie rida, aga meie arendajad on jätnud usaldusväärse mulje /.../ (INT 2).

Intervjuust (INT 4) selgus, et tavaliselt taandub probleemide lahendamine 3-4 geniaalse arendaja suutlikkusele hästi kommunikeerida ning kiiresti programmeerida.

/.../...saime esimesed kogemused, kuidas saame hakkama, kui kliendi poolt puudub sügav läbimõeldus ja hoolivus. Algusaastatel tajusime end kui kellegi teise teadmuse hoidja, kuid see kõik oli inimeste tasandil - see ei olnud otseselt meie ülesanne, meie lisaväärtus /.../ (INT 2)

Samuti toodi välja, et tugev projektijuht ja peakasutaja annavad olulise panuse arendusprojekti õnnestumisele (INT 3). On esinenud ka olukordi, kus arendaja vastutab puhtalt projektijuhtimise eest, sest klient on tugev ja tark tellija, tunneb oma äriprotsesse ning juhib oma huve (INT 2).

/.../ ...kui klient on tugev ja ta tajub, mida muudatus tegelikult tähendab, saan panna ka nooremarendajad tööle ilma, et peaksin kartma, et nad midagi käkki astuvad /.../ (INT 2)

Intervjuudest selgus, et teadlikult küll tarkvaranõuete juhtimist avaliku sektori arendusprotsessi raames ei toimu, kuid on mitmeid praktikaid, kuidas seda mõjutatakse, eelkõige lõpptoote õnnestumise eesmärgil. Näiteks ühe väljakutsena toodi ekspertide poolt välja nõuete jälgitavus.

/.../...kolm aastat hiljem lähen arendusprojekti dokumentatsiooni vaatama, kus on kirjas, et kasutatakse seda, tegelikult kasutatakse teist - keegi ei pannud Confluenci kirja, et tegelikult lõppseis on hoopis selline /.../ (INT 2)

Eksperdi sõnul on lihtne panna nullist kirja kõik nõuded, kuid esimene suur muudatus ajab süsteemi segamini - esimesed kaks väikest muudatust tekitavad tunde, et "need ju ei muutnud midagi olulist" (INT 2). Tegelikult võib-olla ei ole teada, et selle muutuse taga oli väga põhimõtteline määruse või korra muutus, mida kliendipoolne projektijuht/kontaktisik juba analüüsis ja tegi sellest tuletise, mis oli sisendiks arendajale (INT 2). Arendaja ülesanne on jõuda põhjuseni, et sünteesida lahendus probleemile (INT 2). Kliendi ülesanne on kirjeldada oma vajadus võimalikult täpselt. Kuid selge on see, et juba projekti alguses peab olema kokkulepitud põhimõtted, kuidas muutusi juhitakse, sest puudulikku nõuete juhtimist kompenseerib teinekord füüsiline käsitöö (INT 3).

/.../...kas muudatus siin tähendab ka muudatust seal, millist muudatust see äriprotsessis tähendab, missugust muudatust tarkvaratootes /.../ (INT 2)

Eksperdi (INT 4) sõnul vaeveldakse kuni tänaseni probleemiga, kus pole mitte informatsiooni puudus, vaid informatsiooni üleküllus ning selle tõeväärtus on liiga madal. Oleme loobunud illusioonist, et me suudame hoida ajakohast analüüsidokumenti - see ei ole reaalne (INT 4). Näiteks nõuete jälgitavus toimub JIRA piletitega, ning koodi dokumenteerimine *Swaggeris*, kus on kogu dokumentatsioon kirjeldatud (INT 4).

/.../...andmebaasiskeem on aga alati uuendatud - erinevad arendajad tegelevad sama andmebaasi peal siis andmebaasiskeem on alati tagantjärele ka uuendatud /.../ andmebaas ja andmebaasiskeem on omavahel suhtes /.../ (INT 4)

Samas oluliste muudatuste puhul, nt kui toimub ühe samaväärse tüki asendamine teisega, siis see on projektikoosoleku otsusena dokumenteeritud (INT 4). Kokkuvõtvalt on ühe eksperdi kogemusel lahendus see, et tuleb valida võimalikult väikesed tükid, mida suudame kontrollida ja elus hoida - need dokumendid püsivad tõesed (INT 4).

/.../ Ma ei tea mitte kedagi, kes tagantjärgi dokumenteerib - mingi aeg on püütud veel Wikis hoida elus informatsiooni aga ka sellest on loobutud /.../ ...ka suuremates tarkvara ettevõtetes /.../ (INT 4)

Kui teadliku nõuete juhtimist arendusprotsessi raames ei toimu, siis eksperdid loevad kõige tõhusamaks metoodikaks projekti tulemuslikkuse mõõtmiseks prototüüpimist. Näiteks intervjuust (INT 4) selgus, et prototüüp on efektiivne tööriist kasutaja soovide väljendamiseks ja lõpuni mõistmiseks. Kasutaja vajab arusaamist, mida talle pakud ja kasutaja mõistab, et Sina saad aru, mida Sa teed - on oluline, et klient mõistaks võimalikult kiiresti kuidas süsteem reaalselt funktsioneerima hakkab (INT 4).

/.../...esmalt tuleb prototüüpida front ja alles siis hakkad backlogi looma /.../ kui front'il on kasutajale (ja/või tellijale) enam-vähem selge ja alles seejärel järgneb backendi loomine /.../ (INT 4)

Samas kahtleb (INT 5), kas prototüüpimise tulemusena midagi ilmtingimata valmib, mida arenduse hilisemates etappides saab ära kasutada - prototüüp on suhteliselt ajamahukas ning erasektoris on palju levinum elementaarne agiilne meetod. Samas toob antud ekspert välja, et kui protoüüpimise sisuline eesmärk on mingi muu arendustöö ettepoole toomine, siis ei pruugi see üleüldist töömahtu tõsta, kuna see oleks olnud vaja teha niikuinii (INT 5).

/.../ arvan, et üldine reegel on see, et kliendilt tuleks tagasiside saada nii kiiresti ja nii väheste vahenditega kui võimalik /.../ (INT 5)

Intervjuust eksperdiga (INT 4) selgub oluline juhtprintsiip, et otsustega ei tohi kunagi venitada, aga otsused tuleb langetada võimalikult hilja. Kohe, kui mõni tükk arendusest on valmis tuleb see lükata kasutajate käsutusse, et võimalikult kiiresti toota kliendile sellega kasu ning samas saada ka tagasisidet, et positsioneerida edasist suunda (INT 4).

/.../ Analüüsi ja rakendamise faasi vahe on võimalikult lühike - see on hea ühelt seetõttu, et ärilised vajadused võimalikult vähe muutuvad ja teiseks, et on kulutatud aeg ja on tehtud investeering /.../ arendusmeeskond on teinud oma töö ära. Pole põhjust venitada sellega ja mitte anda seda hüve juba kasutajate käsutusse /.../ (INT 4)

Karu (2019) on öelnud, et MKM-i ülesanne on välja mõelda, kuidas riigi IT-süsteemide arendust parendada - praegune tõsiste puudujääkidega lähenemine vajab märkimisväärset ümberkorraldust. MKM esindaja sõnul on MKM-i ülesanne riigis läbi RIA toota universaalseid valmisarendatud tarkvaralahendusi - näiteks TARA või SIGA, mida saab uuesti ja uuesti kasutada erinevates arendusprojektides (INT 3). Sellistest valmis-komponentidest arendamine loob ühtsust ning samas aitab ka kokku hoida kulusid.

/.../ ...näiteks TARA - me ei pea enam arendama sisselogimise funktsioone igas arendusprojektis eraldi. /.../ need Euroopa ID-kaardi omanikud poleks suutnud igale infosüsteemile neid Portugali ID-kaarte juurde panema hakata /.../ (INT 3)

Samuti sai intervjuudes kahel korral märgitud ka projektipõhiselt tootepõhisele ideoloogiale üleminek. Nimelt ühelt poolt on soov liikuda projektipõhisest tootepõhiseks arenduseks, aga paratamatult rahastus käib käesoleval hetkel projektide kaudu, mistõttu on kasutusel *project-product* hübriidmudel (INT 4). Intervjuudest tuli ilmsiks ka mõningaid häid ideid, mille rakendamist alles kaalutakse ja ette valmistatakse. Näiteks üks võimalus on Archimate metoodika (INT 2).

/.../ ....sellel on tasuta tarkvaratoode nimega Archi, millega saab modelleerida kolmel tasandil - riistvaraline tase, süsteemi tarkvaraline tase ja kolmandaks ärilahendus /.../ (INT 2)

Kokkuvõtvalt võib öelda, et teadlikult tarkvaranõuete juhtimist Eesti avaliku sektori arendusprotsesside puhul ei toimu, kuid hoolimata sellest on mitmeid häid praktikaid, mis toetavad muudatuste edukat rakendamist ning samuti eksperdid tunnistavad, et tarkvaranõuete muutuste juhtimine on oluline faktor tarkvaraarenduste eduka tarnimise puhul.

4.2. Näidisjuhtumi ülevaade ja kulg

Käesolevas peatükis kirjeldan näidisjuhtumit, mille raames analüüsin konkreetse asutuse praktikat muutuvate nõuete kontekstis põhinedes avalikele dokumentidele, isiklikule kogemustele, kuid ka uurimuse raames läbiviidud dokumentide analüüsile Lennuametis arendatava LAIS-i näitel, mille üheks osaks on drooniregister. Minu peamine ülesanne peakasutajana on olla tellija esindaja ning ühenduslik lüli Lennuameti ja arendusmeeskonna vahel, et kogu funktsionaalsus ja nõuded saaksid kajastatud lähteülesandes ja detailanalüüsis ning süsteemi vajalikkus ja eesmärgid oleksid pidevalt selged kogu projekti meeskonnale.

4.2.1. Üldine taust

LA on MKM-i valitsemisalas tegutsev valitsusasutus, mis teostab riiklikku järelevalvet tsiviillennunduse üle ning kohaldab riiklikku sundi seaduses ettenähtud alustel ja ulatuses (Lennuamet). Muuhulgas kohalduvad LA järelevalve alla ka mehitamata õhusõidukid ja kõik sellega seonduv. Kuni 2019. aaasta sügiseni oli mehitamata õhusõidukite valdkond Euroopa tasandil reguleerimata ning igal liikmesriigil oli täielik vabadus siseriiklike nõuete kehtestamiseks. Mehitamata õhusõidukite kasutamine ning lennulubade menetlemine on viimastel aastatel konstantselt suurenenud, ca 50%, kuid kõik selle valdkonnaga seonduv toimub LA-s täna veel manuaalselt - *Excel*-is ja *Word*-is, infot töödeldakse käsitsi ja tõstetakse ühest dokumendist teise. Need on vaid mõned tegevused, mis raiskavad kõrgelt kvalifitseeritud lennundus-spetsiifilise riigiteenistuja aega. Aktiivsed droonilennutajad on arendanud veebikeskkonna *drooni.app*, mille kaudu on küll taotlejatel mugav edastada mehitamata õhusõiduki lennutamise või ühekordse loa taotlus LA-le, kuid iga saabunud taotlus registreeritakse ja menetletakse manuaalselt - selline kasutu töö, kus kõrgelt kvalifitseeritud lennundusspetsialist kulutab suure osa tööajast administratiivsele tegevusele, on võimalik automatiseerida ja ühtlasi luua võimekus reaalajas järelevalvetegevuseks kogu valdkonna üle. Vajadus automaatse infosüsteemi järgi on olnud alates 2015. aastast, mil LA töö automatiseerimise ja lennundusohutuse suurendamise eesmärgil alustati menetlussüsteemi arendusprotsessiga, et kiirendada ühekordse loa menetlemise protsessi ning vähendada inimesest põhjustatud vigu (nt inspektor saadab kogemata loa valele isikule jne). Antud arendustegevus peatus 2017. aastal. 13. mail 2018 vahendas ERR (2018), et koostöös MKM-iga on veel enne jaanipäeva

plaanis käivitada mobiililahendus, millega drooni omanik saab oma lennu registreerida sekunditega. Kahjuks ei ole selline lahendus tänaseni avalikkusele kättesaadav.

Täna on olukord muutunud. Lisaks LA vajadusele parendada tööprotsesside efektiivsust ning alates 2019. aastast on surve ka EL-i, täpsemalt EASA poolt, mil jõustus uus EL regulatsioon nr 2019/947, mis kehtestab ühtsed nõuded droonide lennutamiseks terves Euroopas. Regulatsioon näeb ette, et esimese sammuna peab olema igas liikmesriigis rakendatud mehitamata õhusõiduki käitajate ja mehitamata õhusõidukite register ja nende registreerimise süsteem, et ohutuse kaalutlusel toimiks mitmesuunaline info liigutamine Euroopa tasandil - LA on Eestis otsene vastutaja regulatsioonist tulenevate nõuete rakendamise eest - ka drooniregistri valmimise eest. Samas on kogu IT-tugi LA-le tagatud teenusena MKM-i poolt ning antud registri arendamise protsessi veab MKM-i IT-projektijuht. ERR (2019) on vahendanud, et riik valmistub laialdase drooniliikluse alguseks - koostööd drooniliikluse võimaldamiseks vajaliku taristu ja regulatsioonide loomise osas teevad MKM, Lennuliiklusteeninduse AS (edaspidi EANS) ja LA, olles alustanud ettevalmistusi juba 2019. aastal, et õigeaegselt käivitada infosüsteemide arendamise protsess, mis võimaldaks koordineerida mehitamata õhusõidukite ning teiste madalal lendavate õhusõidukite liiklust Eesti õhuruumis.

Kuna valdkond on innovaatiline ning nõuded pidevas muutumises, on LA-l väljakutse, kuidas arvestada peagi algavas arendusprotsessis kõikide oluliste nõuetega selliselt, et register oleks ajakohane ja koosvõimeline ka võimaliku lähituleviku stsenaariumi puhul, mil jõustuvad täiendavad nõuded ja realiseeruvad uued vajadused. Seetõttu peab juba täna esimese ehituskivi - drooniregistri arendamisel - kõikvõimalikke nõudeid suutma ette näha ja rakendada. Juba täna on teada, et registri valmimisel järgneb sellele järk-järgult jätkuarendusi, et luua toimiv süsteem ja saavutada seadusest tulenev eesmärk - suutlikkus mehitamata õhusõidukeid jälgida reaalajas ning drooni operaatoritel peab olema võimalus esitada oma lennuplaan ja -koordinaadid kooskõlastatud lennutegevuseks.

4.2.2. Menetlusprotsess ja selle kirjeldamine

Mehitamata õhusõiduki ja selle käitaja registreerimise menetlusprotsess on laias laastus standardne LA põhiprotsess, olles analoogne loamenetlusega mõnes teises valdkonnas (nt õhusõiduki või selle käitaja registreerimise taotlemine) (DOK 1). Küll aga on igas valdkonnas omad olulised nüansid. Infosüsteemi arendamise perspektiivis on oluline kogu kontseptsiooni ja põhiprotsessi kirjeldus, et

see oleks üheselt mõistetav nii arendajale, kes ehitab tellija soovi järgi abstraktset süsteemi, kui ka tellijale, kes peaks suutma ette näha kõiki võimalikke realiseeruvaid stsenaariumeid.

Mehitamata õhusõidukitega seotud menetlusprotsessis (DOK 2) on mitmeid erisusi, aga ka veel kokku leppimata aspekte, mis mõjutavad üleüldist kontseptsiooni siseriiklikul tasandil, kus seaduse muutmise eelnõud (DOK 3) on alles koostamisel. Näiteks keeruline mitmedimensiooniline opereerimistingimuste jaotus alamkategooriate järgi; käitamisala määramine kaardirakendusel ja vajadusel heakskiitmise protsess koos võimalikus koostöös EANS-iga; järelevalveprotsess ja -võimekus mehitamata õhusõidukite üle; riigilõivud ja trahvimäärad; identifitseerimine kolmandate riikide kodanike või alaealiste puhul; pädevuste andmise protsess ja võimalik koostöö Lennuakadeemiaga; ning võimalik kindlustuse kohustuslikkus mehitamata õhusõidukitele.

4.2.3. Arendusprotsess ja selle juhtimine

Täna toimub riigi infosüsteemide arendus suuresti hanke põhiselt, seda ka LA drooniregistri puhul. Drooniregister, mis hõlmab endas drooni registreerimist, selle käitaja registreerimist ning pädevuste haldust, on osa LA järelevalve infosüsteemi (edaspidi LAIS) hankest, mis sai alguse 2018. aastal, mil alustati äriprotsesside kaardistamise ja lähteülesande koostamisega ning valmis ka esimene prototüüp. LAIS saab olema uus infosüsteem olemasoleva LOIS (Lennuohutuse Infosüsteem) asemel. Tulenevalt 2019. aasta septembris jõustunud regulatsioonist (EL Reg 2019/947) on LA-l plaanis arendada drooniregister LAIS-i hanke esimeses etapis. Tegemist oli SF hankega, hankijaks MKM, kes vastavalt RHS-ile tegi kättesaadava hankedokumendi (DOK 4) koos lisadega (9), mis oli pakkumisteks avatud 28.11.2019 kuni 10.01.2020 (Riigihangete register). Hanke käsitlusalasse kuulus EL regulatsioon nr 2019/947 (DOK 5) käsitlusalas vaid registri arendamine, mis aga ei kata kogu kehtestatud nõudeid - kogu mehitamata õhusõidukite kontseptsioon peab endas hõlmama lisaks registri arendamisele ka registri liidestamist Euroopa-keskse süsteemiga, pädevuslubade ja -sertifikaatide süsteemiga ning võimaliku UTM-iga. Kõik kolm nimetatut on paralleelselt töös olevad projektid, mille valmimine on planeeritud paralleelselt ja pigem hiljem kui LA drooniregistri valmimine. Hetkel on kõige kriitilisem tagada koostalitlus UTM süsteemi projektiga, mida juhib EANS koostöös AirMap'iga, et luua lahendus droonide integreerimiseks Eesti õhuruumi madalamal kõrgusel kui toimub tsiviillennundus (E-estonia, 2019). EANS-i UTM-i projektijuht Tamm on öelnud, et droonivaldkonna kiire kasv nõuab uuenduslikke digitaalseid lahendusi, et tagada ohutu ja turvaline lennuliikluse juhtimine, mille osas

on just LA-l väga oluline roll antud mastaapse projekti õnnestumiseks, sest lähitulevikus peab LA suutma reaalajas seostada mehitamata õhusõiduk selle käitaja ja/või omanikuga, et teostada järelevalvet lennuohutuse eesmärgil.

LA drooniregistri arendusprojekti toimimiseks on loodud kaks töörühma: **juhtrühm**, kelle ülesanne on jälgida projekti ajakavast ja eelarvest kinnipidamist, fikseerida projekti ja selle etappide lõppemised ning langetada vajalikud otsused muudatuste juhtimise osas; ning **töörühm**, kes tegeleb projekti nn operatiivjuhtimisega, tööde lõpetamise ja avamisega, tööde kulgemist takistavate probleemide tuvastamise ja nende lahendamisega. Juhtrühma kuuluvad MKM-i IT-esindajad (2), kellest üks on projektijuht; LA esindajad (2) - peadirektor ja süsteemi peakasutaja; kasutaja/kliendi esindaja (1-2) ning arendaja esindaja (1-2). Juhtrühm on kinnitanud töörühma, kuhu oli võimalik kandideerida kõikidel LA teenistujatel, põhjendades iseenda motivatsiooni ning varasemaid kogemusi ja kokkupuuteid infosüsteemi arendamisega. Töörühma kuuluvad LA esindajad igast sisuosakonnast (4), dokumendi- ja teabehaldusspetsialist, õigusnõunik ning süsteemi peakasutaja; projektijuht; kliendi/kasutaja esindaja(d) (1-2); ning arendaja esindaja(d) (1-2). Positiivne on tõdeda, et nii projekti juhtrühma kui töörühma on kaasatud lisaks LA inimestele ka kliendi ehk kasutaja esindajad väljaspool LA-d.

Toimunud on mõlema rühma koosolekud, kinnitatud projekti ajakava, kommunikatsiooni kokkulepped, töörühma koosseis ning alanud esimesed platvormi seadistamised. Raamlepingu raames sõlmitud hankelepingu I etapi pikkuseks on 18 kuud, mil peab valmima hankedokumendis kirjeldatud LAIS. Hanke raamlepingu pikkuseks tervikuna on 48 kuud. 2020. a. mai lõpus on plaanitud alustada detailanalüüsiga ning esialgse plaani kohaselt peaks vastavalt funktsionaalsete nõuete kirjeldusele olema sama aasta augustiks püsti ja valmis *live*'i minema I osa LAIS-ist ehk drooniregister. Nagu eelnevalt selgunud, on tegemist väga muutliku ja veel fikseerimata valdkonnas toimuva arendustööga - siseriiklik seaduse muutmise eelnõu on koostamisel (DOK 3). Mina, LAIS peakasutajana, olles LA-poolne koordinaator, seisan lisaks kõigele selle eest, et antud LAIS arendusprojekti käigus toimuks maksimaalselt teadlik muutuste juhtimine efektiivse ja tulemusliku lõpptoote valmimise perspektiivist.

5. JÄRELDUSED JA ARUTELU

Käesolevas peatükis toon välja peamised järeldused ning diskuteerin tarkvaranõuete muutuste juhtimise teemal - mis on peamised põhjused ning missugused on avaliku sektori praktikad muutustega toime tulemisel. Lõpetuseks (vt ptk 5.1) jagan omapoolsed ettepanekud nõuete muutuste juhtimise parendamiseks arendusprotsessi vältel.

Tehnoloogia uudsus on tasapisi kadumas ning järjest enam on suurenemas organisatoorse tegevuse mõtestamine avalikus sektoris, et toimimise võimekust IKT vahendite abil veelgi parendada. Eesti riik on olnud teerajaja avalike teenuste digitaliseerimisel, mis teeb Eestist kõige digitaalsema riigi maailmas (e-Estonia), tehes nii mõnelegi suurriigile selles osas silmad ette. Ometi esineb Eesti avaliku sektori tarkvaraarendustes mitmeid väljakutseid, mille ületamine annaks olulise panuse veelgi efektiivsemaks riigi juhtimiseks, võimaldades inimestele kvaliteetse ning kliendisõbraliku avaliku teenuse kättesaadavust sõltumata asukohast. Tänapäeval on meid ümbritsev keskkond, eriti infotehnoloogiline, aktiivses muutumises.

Võitja on see, kes suudab pidevate muutustega sammu pidada. Intervjuud kinnitavad teoreetilist käsitlust, et tarkvaranõuete muudatused on arendusprotsessile loomuomased (Jayatilleke, jt, 2017), sest ilmvõimatu on kõiki tarkvaranõudeid briljantselt juba projekti alguses ette näha, kuna keskkond, asjaolud ja stsenaariumid on ajas muutuvad. Samuti kinnitasid intervjuud ka Ennuse (2017) sõnu, et arendusprojekti alguses planeeritakse paratamatult üpris palju ebaolulisi funktsionaalsusi, mida on võimalik arenduse vältel välistada koostöös lõppkasutajatega (Pau, 2017). Määravalt oluline on selge eesmärk - tähtajast ja eelarvest kinnipidamine nõuab detailides paindlikkust. Tarkvaraprojektide puhul on võimalik fikseerida projekti ärilised eesmärgid ja vajadused, aga konkreetseid tehnilisi lahendusi (funktsionaalseid omadusi) tuleb pidevalt projekti käigus mingil määral lõppkasutajatega koostöös katsetada ja täiendada (Pau, 2017). Samuti kinnitasid eksperdid, Bano jt (2012) väidet, et tarkvaraprojekti õnnestumine sõltub suuresti hoopis sellest, kuivõrd hästi suudab meeskond muutusi aktsepteerida ning nendega toime tulla.

Igaühel meist on kujunenud harjumused, uskumused ja vaated elule, kuid tööprotsesside puhul peab olema primaarne valmisolek jätta kõrvale enda isiklikud väärtused ning keskenduda organisatsiooni väärtustele. Vastavalt intervjuude analüüsitulemustele on üheks muudatuste peamiseks põhjuseks just inimfaktor - unustamine, mitte arusaamine, puudulik motivatsioon ja pühendumine, harjumustesse kinni jäämine jne. Inimeste kitsa silmavaate tagajärjel võib realiseeruda stsenaarium, kus valmiv infosüsteem on sama stagnaatiline nagu varasem, lihtsalt digitaalses vormis. Ka Aavik (2018) on oma intervjuus välja toonud, et Eesti riik kannab tohutut digitaalset taaka - sajad tuhanded protsessid on kahekümne aasta jooksul jõudnud paberilt veebivormi täpselt samasugusena, nagu olid paberil või minimaalsete muudatustega. Nagu ka intervjuudest selgus, on selle põhjuseks ebapiisav protsessi optimeerimine ja lihtsustamine, mis omakorda on tingitud väga erinevatest põhjustest. Näiteks ressursipuudus, mistõttu inimesed on sunnitud keskenduma vaid esmastele tööülesannetele ning kõik, mis puudutab (p)arendamist, on teisejärguline. Siinkohal on asjakohane sekundeerida Karu (2019), kes on öeldnud, *et /.../ lisaks sellele, et otsida paremaid vastuseid küsimusele "kuidas?", tuleb rohkem tähelepanu pöörata küsimustele "miks?" ning "kelle jaoks?" - ükski IT-arendus ei ole väärtus iseeneses /.../*. Samuti leian, et organisatsiooni äriprotsessid ja infosüsteemi kontseptsioon peab olema paigas juba enne arendusprotsessi algust. Ühe ohukohana LA LAIS arenduse puhul näen, et oleme küll alustanud teenuspõhise juhtimise juurutamisega, seostanud teenusvaate protsessipõhise vaatega, kuid meil ei ole veel 100% kaardistatud kõik andmehulgad iga teenuse kohta, mis võib olla üheks komistuskiviks detailanalüüsi koostamisel. Samuti LA kogemusel näen, et inimesed ei teadvusta olulist põhimõtet, et infosüsteem teeb vaid seda, ning kogub vaid sellised andmeid, mida spetsialist ise süsteemile sisendiks annab.

Ohukohana näen ka seda, et inimesed armastavad kasutada iseendale tuttavaid sõnu ning erinevuste puhul esmalt eitada sarnasust ja standardseid vaateid, isegi kui protsessivaade on menetlustel ühtne. Näiteks loamenetlus ja õhusõiduki registreerimine on protsessi vaates üpris standardsed menetlused, mõlema väljundiks on sertifikaat, mis ühes spetsiifikas nimetatakse *pädevust tõendav luba* ja teisel juhul *õhusõiduki tunnistus* (DOK 1). Siinjuures on kindlasti abiks ühtsete terminite ja ontoloogia kasutuselevõtt, nagu tõi välja ka Alsanad jt (2019), et mitte alati pole kokku lepitud ühtseid ontoloogilisi põhimõtteid, millest tulenevalt võib esineda vääriti mõistmist ning infokadu.

Järjest enam on populaarsust kogumas ka dünaamiliste süsteemide arendamine, kus vigade tekkimise võimalust tõstab erinevate lahenduste rohkus, mistõttu on ka muudatused suurema tõenäosusega

tingitud just sellest. Dünaamiliste süsteemide arendamise näidet ilmestab järgmine oletus, kui esitaksime maja ehitamiseks samasuguse tellimuse, nagu esitatakse tellimusi infosüsteemidele - näiteks Rahvusringhääling tahab endale uut maja ja esitatakse tellimus järgmiselt: *„Tahame, et meie majal oleks talviti 12 korrust ja suviti viis korrust, oleks 3–5 liftišahti ja vastavalt vajadusele sõidaks liftišahtides 2–7 lifti ja stuudiod oleksid tööpäeviti hoovi pool ja pühapäeviti tänava pool.''* - Eesti IT-süsteemide arhitekt Priit Raspeli sõnul infosüsteeme niimoodi tellitaksegi (Himma, 2018).

Samuti, olles LA juhtimissüsteemi eest vastutaja, olen täheldanud, et inimestele ei ole tuttavad elementaarsed protsessijuhtimise põhimõtted, mida kinnitasid ka intervjuud. Protsesside kirjeldamisel kipuvad spetsialistid unustama lihtsaid ja nende jaoks elementaarseid detaile, kirjeldavad protsessi segases järjekorras ning eba-struktuurselt. Minu ülesanne LA-s on olla kvaliteedikontroll - luua tegevustele kronoloogia ja selge struktuur, ühtlustada sõnakasutust ning mitte-spetsialistina kontrollida protsessi arusaadavust. Avaliku sektori organisatsioonide põhiprotsesside olemus võib olla väga erinev, alates suhteliselt abstraktsetest tegevustest nagu poliitikakujundamise toetamine või majandustegevuse reguleerimine, kuni väga konkreetsete teenuste otsustamiseni - vajadus pakkuda kodanikele/klientidele ja teistele huvirühmadele üha rohkem väärtust ning suurendada tõhusust on protsesside arendamise ja uuendamise kaks peamist stiimulit (Eesti Kvaliteediühing). Ometi ei ole see avalikus sektoris veel harjumuspärane, kuid siiski sunnib pidev areng ja muutused organisatsioone oma protsesse pidevalt parendama.

Henry Fordi väide */.../ Kui ma oleks inimestelt küsinud, mida nad tahavad, oleks nad palunud kiiremaid hobuseid /.../* ilmestab hästi inimeste suutmatust näha olukorda uuest vaatevinklist. Innovatsioon on kõige üldisemas tähenduses millegi uut moodi tegemine, mis tihti on keeruline ülesanne. Ometi võiks olla kogu projektimeeskonnal selge arusaam organisatsiooni eesmärgist, sest reeglina on igal arendusprotsessil väga selge eesmärk - arendada tööprotsessi toetav infosüsteem. Siinkohal on kindlasti suunavaks jõuks projektijuht, kelle huvi on seista organisatsiooni väärtuste eest, viia ellu projekt vastavalt eesmärgile, näha suurt pilti ning suunata kasutajaid mõtlema parendavate lahenduste peale. See on inimlik, et inimesed kardavad kaotada oma tähendust ja positsiooni professionaalses elus, mis võib nende silmis olla tingitud protsesside digitaliseerimisest. Seetõttu on teinekord lihtsam töötada vastu just põhjusel, et mitte jääda ilma oma töökohast aga samal ajal ei suuda mõelda nad sellele, et ehk tööprotsessi automatiseerimine annab võimaluse regulaarselt panustada juhtimissüsteemi arendamisesse ja tööprotsesside regulaarsesse ülevaatamisse, misläbi

tõuseb inimese kvalifikatsioon veelgi. Regulaarne protsesside ülevaatamine harib ja paneb inimest märkama uudsust ja samas toob kasu ka organisatsioonile - isegi kui protsessis midagi ei muutu, siis meid ümbritsev keskkond areneb pidevalt ning uuendustega kaasas käimine on kindlasti üks oluline edutegur.

Eksperdid kinnitasid ka Al-Saiyd ja Zriqat (2015) fakti, et inimfaktor on peamiseks nõuete muutuste põhjustajaks, aga samas on just inimesed need, kes arenevad arendusprotsessi käigus. Kui kliendi arenguhüpe seisneb peamiselt tema protsessi- või IT-teadlikkuse kasvus, siis arendaja väärtus jällegi konkreetse organisatsiooni või valdkonna tundmises - siit saab sündida targa tellija ja hea arendaja sümbioos, mis on samuti üks väga oluline faktor toimivaks kommunikatsiooniks, mis on teadagi üheks võimaluseks panustada projekti õnnestumisesse. Muidugi ülimalt oluline on eelkõige tellija enda pühendumine, motivatsioon ja valmisolek kirjeldada, parendada ja juurutada uuendusi oma tööprotsessidesse, et tuvastada raiskavaid tegevusi, mis väärtust ei loo ning neist vabaneda. Näiteks LA näitel oli LAIS arendusprotsessi töörühma moodustamisel märksõnaks just motivatsioon ja kogemus, mida rõhutas ka meie arenduspartner, et inimesed, kes kuuluvad projekti meeskonda peavad olema maksimaalselt motiveeritud. Selleks korraldasime LA-s konkurssi, tõestamaks iga kandidaadi motivatsiooni ja valmisolekut panustada loodava süsteemi arendusse. Üllatuseks leidsime konkurssi tulemusel igast sisuosakonnast märkimisväärse kogemusega ja pühendunud esindaja. Samuti kinnitavad Al-Saiyd ja Zriqat (2015) intervjuude tulemust, et üheks inimfaktoriga seotud muudatuse põhjuseks võib olla kolmandast osapoolest tingitud muudatused. Iga projekti puhul on äärmiselt oluline teada, kes on kolmas osapool, kui palju neid on ning missugune on süsteemi sõltuvus nendest välistest esindajatest ja vastupidi. Siinkohal järeldan, et tuleb investeerida inimestesse ja organisatsiooni arendamisesse. See eeldab pidevat õppimist, et arendada uusi teadmisi eelkõige protsessi- ja IT teadlikkuse osas, et tellijal oleks juba enne hankesse minemist kirjeldatud ja optimeeritud põhiprotsessid, mille pinnalt on arendajal võimalik mõista tellija vajadust.

Teatavasti on üks avaliku sektori ülesanne kehtestada seaduseid, mistõttu on avaliku sektori tarkvaraarendused suuresti mõjutatud ka seadusemuudatustest, mis on Riigikontrolli (2019) auditi tulemuste põhjal üheks peamiseks ohuteguriks. Riigikontroll (2019) tõi auditis välja, et üheks oluliseks nõuete muutumise põhjuseks on seadusemuudatused - arendusprotsessi vältel jõustuvad uued seadused, mis võivad viia kogu protsessi rivist välja. Täpselt analoogne näide on ka käesolevas töös mainitud LA drooniregistri puhul, kus nõuded on küll kehtestatud EL-i tasandil, kuid siseriiklik

õigusloome alles loomisel, mis võib tingida infosüsteemi arenedes ebakõlasid. Sekundeerin Aavikule (2018), et seadusloome ei tohi takistada digitaalse ühiskonna arengut. Juba seaduseelnõude koostamise etapis tuleb arvestada asjaoluga, et tõenäoliselt nõuab seadusemuudatus infosüsteemide loomist või täiendamist. Arvan, et intervjuus (INT 3) toodud näide moodustada lisaks juht- ja töörühmale ka juriidiline töörühm, kelle ülesanne on tegeleda juriidilise poolega, on igati asjakohane. Näiteks LA näitel, ku meie tervikkoosseis on võrdlemisi õhuke ja valdkonna spetsialiste napib, oleme olukorra lahendanud selliselt, et töörühma koosseisu kuulub ka õigusnõunik.

Kui erasektoris toimub arendustegevus peamiselt erakapitalil põhinevalt, mille puhul on lihtne mõõta ja hinnata loodava väärtuse mõju, siis avalikus sektoris on väärtusloome pisut abstraktsem, mistõttu lõppkasutaja ei pruugi alati tajuda kohest tulemust, mis raskendab kasutaja võimekust projekti panustada. Siinkohal on kindlasti üheks töötavaks lahenduseks prototüüpimine, mis Sommerville (2011) sõnul on lahendus muutuste ennetamiseks, et prognoosida vajalikke ja võimalikke muudatusi ning samas aidata kliendil märgata tugevusi ja potentsiaalseid ohukohti, et vähendada muudatusettepanekute arvu ka pärast süsteemi tarnimist. Kuigi sageli tuuakse välja, et prototüüpimine ei ole veel avaliku sektori tarkvaraarendusprojektide puhul väga sage valik, siis käesoleva magistritöö ekspertide ja konkreetse töö taustnäitel seda siiski tehakse. Avaliku sektori arendusprojektide hind on enamasti fikseeritud ning arendaja jaoks on prototüüpimine lisakulu - ometi võib see vältida mitmeid kulutusi projekti hilisemates faasides. Prototüübi tegemine võtab arendajalt palju aega, mistõttu ta on kulukas - seega ei pruugi olla see arendajale kasulik kui tellija seda hankes eraldi nõudnud pole. Intervjuudest selgus, et prototüüpimine on üks tõhus meetod pidevaks tulemuslikkuse mõõtmiseks läbi projekti. Ometi toob Sommerville (2011), et prototüüpimisega seotud eesmärgid tuleb kokku leppida kohe arendusprotsessi alguses. Jällegi sekundeerin Ennust, et suurte projektide puhul tuleb arendusprotsessi kaasata tellija poolt väga erineva profiiliga inimesi: juhte, spetsialiste, kuid kindlasti ka lõppkasutajaid väljaspoolt organisatsiooni, näiteks edukate arendusprojektide puhul on tema hinnangul tavaline, et arendusprojekti kaasatakse vajadusel ka lõppkasutajate testgrupp (Pau, 2017). Prototüüpimise juures tuleb kindlasti silmas pidada seda, et inimesed ei vaja prototüübi kasutamiseks juhendamist. Eksperdi sõnul (INT 4) on prototüüp eelkõige indikaator, mis näitab kasutajamugavust. Tulenevalt eelnevast järeldan, et avaliku sektori arendusprotsesside õnnestumise huvides on oluline suurendada prototüüpimise tähtsust, kuna see on üks hea võimalus panna lõpplahendus juba enne tarnimist kliendiga suhtlema, et saada tagasisidet ning sisendit võimalike ettetulevate muudatusvajaduste kohta.

Riigikontroll (2019) on välja toonud, et kui arenduskordades on küll paika pandud rollid ja vastutajad, mida kinnitasid ka intervjueeritud eksperdid, siis muudatuste haldamine on enamasti reguleerimata - ei ole kuigi täpselt määratletud, kuidas toimub arenduse ulatuse hindamine, laiendamine ja tähtaegade pikendamine (Riigikontroll, 2019: 13). Samuti muudatuste dokumenteerimine, mis võimaldab jälgida muutuste kulgemist kuni algpõhjuseni, mõjuala ning seoseid teiste nõuetega, mida täna avaliku sektori arendusprojektides teadlikult ei rakendata. Muudatused peavad olema dokumenteeritud ja jälgitud, ning sõltuvused analüüsitud. Nagu intervjuudest selgus, võib olla muudatuse taotlemine põhjustatud ka lihtsalt ebaselgusest, mille lahenduseks on tihti lihtne selgitustöö. Mõnel juhul mainiti ka, et muudatusettepaneku registreerimine on tihti jäänud arendaja hallata, mis tegelikult peaks olema selgelt projektijuhi ülesanne. Näiteks Mäder ja Egyed (2015) on avastanud, et jälgitavusvõimega subjektid sooritasid etteantud ülesande 24% kiiremini ja lõid keskmisel 50% õigemaid lahendusi, mis viitab sellele, et jälgitavus mitte ainult ei säästa pingutusi vaid võib oluliselt parandada ka hoolduse kvaliteeti. Kuna avaliku sektori tarkvaraprojektid on eelkõige hinnatundlikud, tasub muudatustaotluse analüüsimisel pöörata tähelepanu kindlasti ka muudatuse võimalikule maksumusele - nii aja kui ka raha perspektiivist. Samas intervjuudest selgus, et mitte alati ei ole muudatuste elutsükkel jälgitud, mis tingib tulevikus mõttetut lisatööd. Samuti aitab olukorda ohjata ning klienti distsiplineerida kokkulepitud eesmärgid iga etapi jaoks ning määratleda piirnormid, millest alates pole teatud liiki muudatused lubatud - näiteks keelata suured muudatused kui moodul on täitnud juba 75% kogumahust. Kõik muudatused, mis on suuremahulisemad või hoopis ilmnevad liiga hilja, tuleks koondada jätkuarenduseks ning arendada eraldi. Lisaks on oluline veenduda, et muudatustaotluse analüüsi oleks kaasatud antud valdkonnas/ teemas vajalikud isikud ning projekti ajakava ja dokumentatsioon oleks vastavalt muudatustele ajakohastatud. Siinkohal järeldan, et infosüsteemide arendusprojektide õnnestumise suurendamiseks avalikus sektoris on vajalik juurutada tarkvaranõuete muutuste juhtimise protsess - muutmistaotluste vastuvõtmiseks, analüüsimiseks ja huvigruppide kaasamiseks, mis peab olema kõigile selge. Eraldi rõhutan ka muudatuste jälgitavust.

Toetan intervjuudes märgitud projektipõhiselt tootepõhisele ideoloogiale üleminekut. Kui projektil on kindel algus ja lõpp siis tänapäevase tarkvaraarendamise puhul selline põhimõte üksinda ei kehti, kuid ometi projektipõhiselt IT-arendusi juhitakse. Sekundeerin Karu (2019), projektijuhtimine ei ole universaalne lähenemine igas võimalikus olukorras, paratamatult käib rahastus käesoleval hetkel projektide kaudu, kuid samal ajal vajab tootepõhisus hoopis elutsükli põhist vaadet, kus toote elutsükkel algab ideest ning lõpeb olukorras, kus toodet enam ei kasutata. Arendusprotsessiga seotud

ettepanekud võiksid olla seotud projektipõhise raami paindlikuks muutmine ja etappideks jaotamine nagu tõi ka Sommerville (2011), kirjeldades üksikasjalikult igas etapis vajalikud ülesanded, ressursid ning erinevad stsenaariumid. Ka Helmese arendusjuht Ennust on öelnud, et /.../ *Suuri projekte tuleks ellu viia väiksemate alamprojektide või äriliselt loogiliste osade, näiteks ühe teenuse kaupa. Tihti on suurte projektide läbiv risk inimlik optimism – loodetakse, et mahuka projekti detailset käekäiku on võimalik kaks-kolm aastat ette näha. Terviklike väiksemate alamosade arenduste eraldi tellimise ehk paindliku (ingl agile) tarkvaraarendusega saab palju kiiremini aru, kui projekt hakkab valesse suunda kiskuma ja korrektuuride tegemine on kordades odavam /.../* (Pau, 2017). Sekundeerin, et selliselt väheneb ka risk ümbritseva keskkonna muutumise ja pika projekteerimisperioodi jooksul tehtud otsuste vananemise osas.

Samuti tuleks veenduda, et projekti kavas võetakse arvesse olemasolevaid ressursside piiranguid ning, et katsetamiseks ja kvaliteedikontrolliks oleks piisavalt aega. Lõpptootele loob kindlasti väärtust täiesti kõrvaliste isikute kaasamine arendusprotsessi. Arendusprotsessi optimeerimise huvides oluline informatsiooni visualiseerimine selle arusaadavuse ja omandamise kaalutlusel (nt piltide, diagrammide, graafikute, skeemide, kaartide ja tabelitena). Seda muidugi tingimusel, et projektimeeskond ja protsessiga seotud inimestel on võimekus infot mõista, seda monitoorida, tuvastada mustreid ja parendada tööprotsesse. Mõttekohana toon siinkohal välja ka veel eksperdi arvamuse sellest, et ehk oleks mõistlik rakendada avaliku sektori tarkvaraarenduste puhul sarnaselt ehitusvaldkonnale ehitusjärelevalvega analoogset loogikat, et oleks keegi erapooletu, kes teostab tehnilist järelevalvet ning seisab tellija huvide eest. Sarnane loogika võib olla SF rahastusega projektide puhul, kuid nagu juba varasemalt töös selgunud, siis SF projekti puhul on kontrolli eesmärk eelkõige raha eesmärgipärane kasutamine, mitte süsteemi efektiivsuse või töökindluse mõõtmine.

Intervjuudes ekspertidega sai kajastust ka muudatuste mõju projektile aga ka lõpprootele. Kõige esmasem mõju, millest lähtus peamiselt ka Riigikontroll (2019) oli seotud tarneaja nihkumisega, ehk tulenevalt nõuete muutumisest kipuvad projektid venima ning ajakava nihkuma. Eriti riskantne võib see olla SF projektide puhul, kus ajaraam on selgelt määratletud aga ka näiteks LA puhul, kus ajaraami kehtestab EL. Teiselt võib olla nõuete muutumisel mõju ka projekti käsitlusalale - projekti skoop võib laieneda või kitseneda. Näiteks avaliku sektori fikseeritud hinnaga projektide puhul võib see tähendada arendajale tasuta töö tegemist aga nagu töö esimeses osas selgus on mõistlik arendajal juba pakkumise faasis sisse arvestada teatav puhver muudatuste juhtimiseks. Kolmas oluline ja otsene

mõju projektile on seisakud mis võib olla tingitud näiteks seaduse muutmise vajadusest, mis omakorda tingib projekti seiskumise - arendusmeeskond ei saa tööga jätkata ja ressurss seisab tühjalt ning ajakava läheb lõhki. McGee ja Grees (2012), et muutuvad tarkvaranõuded ei ohusta ainult lõpplahenduse edukat tarnimist, vaid sellest sõltub ka võimalik lõpplahenduse kasutatavus ja väärtuslikkus. Siinkohal järeldan, et süsteemne tarkvaranõuete juhtimine, sealhulgas ka nõuete muutuste juhtimine kogu arendusprotsessi vältel annab olulise panuse efektiivse, toimiva ja kasutajasõbraliku lõpplahenduse loomisele.

Kokkuvõttes mõjutab tarkvaranõuete muutumine arendusprotsessi vältel oluliselt projekti õnnestumist - sealhulgas ka kulusid, mis mõjutavad oluliselt valmiva süsteemi omadusi.

5.1. Ettepanekud

Tulenevalt käesoleva magistritöö tulemuste osas tehtud järeldustest ja arutelust teen järgmised ettepanekud avaliku sektori tarkvaraarenduse protsessi optimeerimiseks ja parendamiseks, et tulla toime muutuvate nõuetega arendusprotsessi vältel.

Esmalt pakun välja **muudatuste juhtimise protsessi kirjelduse** (vt Lisa 1), mis tagab muudatuste süsteemse haldamise põhimõtted, eesmärgiga sarnase või seotud informatsiooni selgeks ja süsteemseks hoiustamiseks kindlas kokkulepitud asukohas koos viidetega konkreetsele tõendusdokumendile reaalajas - nt ajakava, tööplaan, analüüsidokumendid jne.

Joonis 4. Tarkvaranõuete muudatuste juhtimise protsess (Autori loodud).

Mudel koosneb neljast sammust: muudatusettepaneku registreerimine, selle analüüs, otsustamine ja muudatuse juurutamine, mille olen täpsemalt lahti kirjutanud käesoleva tll lisas (vt Lisa 1). Mudeli olen tuletanud kvaliteedijuhtimises tuntud PDSA (ingl *Plan-Do-Study-Act*) mudelist (Langley, Nolan, Nolan, Norman, Provost, 1996: 10)

Teise ettepanekuna juhin tähelepanu, et tarkvaraarenduse alus- ja töödokumendid tuleb hoida ajakohased ning kättesaadavad reaalajas - näiteks soovitan selleks kasutada mõnda koostööplatvormi nagu *Confluence*, *Teams* vms, mis võimaldab mitmel erineval isikul reaalajas sama dokumendiga

töötada. Oluline on kindlasti see, et koostööplatvormi valikul oleks see heakskiidetud avaliku sektori turvalisuse kaalutlustel ning samuti ka see, et oleks defineeritud, missugused nõuded peavad olema dokumenteeritud ja missugusel tasandil. Kõik vajalikud muudatused peavad olema integreeritud ja uuendatud, et ka tulevikus oleks võimalik tagasivaatavalt vajaliku ja asjakohast infot kiiresti leida.
Kolmas ettepanek on suunatud peamiselt avaliku sektori esindajatele ehk tellijale, kelle ülesanne on hankida infosüsteemi arendust sealhulgas koostada lähteülesanne ning juurutada uus tarkvara eesmärgipäraselt. Selleks tuleb investeerida inimestesse: panustada nende protsessi- ja IT-teadlikkuse suurendamisse ja suunata neid teadlikult arendama innovatsiooni ja nägema kaugemale iseenda harjumuspärasest perspektiivist. Samuti leian, et oluline on inimese eneseteadlikkus oma rollist ja vastutusest. Kõik meeskonnaliikmed peavad olema teadlikud oma ülesannetest, mis on oluliseks kriteeriumiks hea juhtimise puhul. Muutuvas maailmas on edukad need, kes on teadlikud, suudavad olla paindlikud, kohanevad ja tulemuslikud. See eeldab pidevat õppimist, et arendada uusi teadmisi ja arusaamu. Samuti on targa tellija oluline omadus teadlikkus vajadusest - tellijal on juba enne hankesse minemist kirjeldatud ja optimeeritud põhiprotsessid, mille pinnalt on arendajal võimalik mõista tellija vajadust. Selleks, et oleks eeldus toimivaks protsessi juhtimiseks, on vaja panustada organisatsioonide strateegilise juhtimise arengusse - nimelt avalikus sektoris on tegevuste väljundid võrreldes erasektoriga üpris abstraktsed, mistõttu on inimestel keeruline ilma juhtimissüsteemi ja tulemusmõõdikuteta oma töö väärtust mõista.

Viimane, kuid mitte vähem oluline ettepanek, on suurendada prototüüpimise tähtsust. Prototüüpimine on üks oluline, tõhus võimalus ja meetod, et panna lõpplahendus juba enne tarnimist kliendiga suhtlema, saada tagasisidet ning sisendit võimalike ettetulevate muutuste kohta. Kuid oluline on märkida ka tõsiasi, et kasutaja ei vaja enne prototüübi kasutamist koolitust või juhendamist, kuna see, kuidas kasutaja saab prototüübis hakkama kohe esimesel korral, on oluline indikaator prototüübi kvaliteedi ja kasutajasõbralikkuse kohta.

KOKKUVÕTE

Käesoleva magistritöö eesmärk oli välja selgitada, missugused tarkvaranõuded on kõige altimad muutuma, mis on nõuete muutumise peamised põhjused ning missugused on avaliku sektori praktikad tarkvaranõuete muutumise juhtimiseks. Tsiteerides Lennart Meri: /.../ *Muutuvas maailmas võidab see, kes maailmaga koos käib, käib natukene kiiremini kui maailm. Jõuab maailmast ette, oskab ette näha neid probleeme, neid küsimusi, neid lahendusi, mida elu talle seab /.../.*

Antud magistritöö teema valisin lähtuvalt teema relevantsusest - maailm on järjest enam ja kiiremini muutumises ning tarkvaranõuete muutuste juhtimine võib olla just üks võimalus ja edutegur Eesti avaliku sektori tarkvaraarendusprojektide õnnestumisel. Eesmärgi saavutamiseks püstitasin uurimisküsimused.

- Missugused nõuded on kõige altimad muutuma?
- Mis on tarkvaranõuete muutumise peamised põhjused?
- Missugused on avaliku sektori praktikad muutuvate tarkvaranõuete juhtimiseks? Missuguseid protsesse ja tehnikaid selleks kasutatakse ning mis on peamised väljakutsed?

Töös keskendusin tarkvaraarendusprotsessi nõuete juhtimisele muutuvas keskkonnas, mis on tingitud peamiselt kliendist - kliendi oskus, võimekus ja ettenägelikkus, mille ilmestamiseks toon näidisjuhtumi LA-s arendatava drooniregistri arendusprotsessist.

Andmete kogumiseks viisin läbi viis poolstruktureeritud intervjuud valdkonna ekspertidega: projektijuht, tellija esindajad ning arendaja esindajad ning tegin kvalitatiivset sisuanalüüsi erinevate dokumentide osas. Tulemustele kõrvutasin iseenda ekspertarvamuse ja -kogemuse süsteemi peakasutaja seisukohast.

Intervjuudest selgus, et avaliku sektori tarkvaraarendusprotsessi käigus teadlikult tarkvaranõuete muutuste juhtimist ei toimu. Ometi kinnitasid eksperdid tarkvaranõuete muutumist, mis on paratamatus. Peamiste põhjustena tulid välja inimfaktoriga seotud muutused, mis on enim põhjustatud piisava teadlikkuse puudumisest - arendusprojekt on oluline arenguhüppe platvorm tellijale või hoopis

inimeste vahetumisest võtmepositsioonidel. Ekspertide hinnangul on esinenud olukordi, kus arendaja poolt tehtud detailanalüüs on tehniliselt õige, kuid tegelikkuses ei suuda inimesed seda mõista ega omaks võtta. Samuti said olulise muutuse põhjustajana kajastust seadusmuudatustest tulenevad muutused, puudulikest protsessi- ja strateegilisest juhtimisest tulenevad muutused, aga ka kolmandatest osapooltest tingitud muutused.

Teadlik muudatuste juhtimise protsessi rakendamine võib olla oluline võimaldaja, aga samas ka väljakutse - mõista muutuste ennetamise olemust, mis on ebamäärane ning keeruline. Ometi selgus intervjuudest, et kasutusel on mitmed muutusi toetavaid praktikaid. Esiteks kombineeritakse erinevaid arendusmeetodeid - kui põhiprojekti juhtimine toimub koskmudeli põhimõttel, siis madalamal tasemel on kasutusel mitmeid agiilse meetodi põhimõtteid, nagu näiteks pidev kommunikatsioon, projektijuhtimise põhimõtete rakendamine ja projektijuhi oluline roll. Intervjuudest selgus, et tugev projektijuht ning peakasutaja on oluline edutegur projekti õnnestumisel, samuti juriidilise taustaga inimeste kaasamine töörühma tasandil.

Tulenevalt toodud järeldustest pakun neli võimalikku ettepanekut (vt ptk 5.1) avaliku sektori tarkvaraarenduse protsessi parendamiseks: tarkvaranõuete muutuste juhtimise protsessikirjeldus; tähelepanu ja fookuse suurendamine dokumentide jälgitavuse tagamisel; pidev panustamine inimeste ja organisatsioonide arengusse; ning prototüüpimine, mis on ekspertide sõnul üks tõhusaim meetod muutuste võimalikult varajaseks tuvastamiseks tänastes tingimustes. Pidev muutus on reaalsus ja edukad arendusmeeskonnad peavad saama hakkama dünaamilises keskkonnas.

SUMMARY

Subject: The Principles Of The Information System Development In Case of Changing Requirements In The Public Sector Of Estonia

The purpose of this master thesis was to find out which software requirements are most prone to change; what are the main reasons for the change of the requirements and what are the practices in the Estonian public sector for managing the changes of software requirements. Quote by Lennart Meri, President of the Republic of Estonia from 1992 to 2001 /.../ *In a changing world, will win whoever walks with the world a little faster than the world. Comes ahead of the world, can anticipate the problems, the issues, the solutions that life poses to him* /.../.

The topic of this master's thesis is based on the relevance that the world is changing more and more, and managing the changes of software requirements can be just one opportunity and success factor for the development projects in the Estonian public sector. To achieve this goal, I have set up the following research questions:

- Which requirements are most prone to change?
- What are the main reasons for the change of software requirements?
- What are the practices in the Estonian public sector for managing the change of software requirements? Which processes and techniques are used and what are the main challenges?

Mostly, I have focused on managing the requirements during the software development process in a changing environment, which mainly depend on the clients' skills, ability and foresight. For the sake of illustration, I will give a sample case of the development process of drone register being developed in the ECAA (Estonian Civil Aviation Administration).

For data collection, I conducted five semi-structured interviews with experts on the following positions: project manager, customer representatives and developer representatives. In addition, I performed a qualitative content analysis of the relevant documents. The results are compared with my

own expert opinion and experience from the point of view of the main system user. The interviews revealed that the changes of software requirements during the software development process in the public sector are not consciously managed. However, experts confirmed the changing software requirements, which are inevitable. The main reasons were related to human factors, which are mostly caused by the lack of sufficient awareness - software development projects are an important leap platform of development for the client or changing people in the key positions. According to the experts, there have been situations where the detailed analysis performed by the developer is technically correct, but in reality people are unable to understand or accept it. Following changes were also reflected as significant causes: changes due to changes in law; changes due to poor process and strategic management; as well as changes due to third parties.

Conscious implementation of the management of change process can be an important enabler, but also a challenge to understand well the nature of the prevention of change, which is vague and complex. However, the interviews revealed that there are several following practices that support the taming of changes. For example combining different development methods: while the main project management is based on the Waterfall model, several agile methods are used at a lower level - such as continuous communication. Next, considering important project management principles, where the project manager plays an important role - interviews reveal that a strong project manager and a main user are both vital factors for a successful project. Last, but not least, the involvement of legal advisors at a working group level or arranging a separate legal working group, if necessary.

Based on the conclusion, I have proposed four possible suggestions to improve the software development process in the Estonian public sector: 1) describing the software requirement in the change management process; 2) increasing the focus on the traceability of relevant documents; 3) contributing continuously to the development of the individuals and the organisation and 4) prototyping, which, according to experts, is one of the most effective ways to detect changes as early as possible. As constant changes of software requirements is a reality, development teams must be able to cope in a dynamic environment to be successful.

KASUTATUD MATERJALID

Aavik, A. (2018). *Eesti riigi digitaalne taak.* Kasutatud 29.04.2020, https://arileht.delfi.ee/news/uudised/eesti-riigi-digitaalne-taak?id=84304375

Agilealliance. (2020). *Product Backlog*. Kasutatud perioodil 20.05.2020 kuni 24.05.2020, https://www.agilealliance.org/glossary/backlog

Alsanad, A.A., Chikh, A., Mirza, A. (2019). *A Domain Ontology for Software Requirements Change Management in Global Software Development Environment*. IEEE Access, 7, 49352 - 49361. DOI: 10.1109/ACCESS.2019.2909839

Al-Saiyd, N., Zriqat, E. (2015). *Analyzing the impact of Requirement Changing on Software Design. European Journal of Scientific Research.* Volume 136 No 1 November 2015. Kasutatud 22.02.2020, https://www.europeanjournalofscientificresearch.com/issues/EJSR_136_1.html?fbclid=IwAR14Vgda6OhM1UKtjTL0e0yCv1mAiMngLNf_FiiqCBsG27nX2unYf55SU68

Anwer, S., Wen, L., Wang, Z. (2019). *A Systematic Approach for Identifying Requirement Change Management Challenges: Preliminary Results*. EASE, April 2019, 230 - 235. DOI: 10.1145/3319008.3319031

Atkinson, R. (1999). *Project management: cost, time and quality, two best guesses and a phenomenon, it's time to accept other success criteria*. Pergamon, 6, 337 - 342. Kasutatud 10.03.2020, https://notendur.hi.is/vio1/Project_management_Cost_time_and_quality.pdf

ATKRR. (2013). *Avalike teenuste korraldamise roheline raamat.* Majandus- ja Kommunikatsiooniministeerium. Kasutatud perioodil 22.01.2020 kuni 19.03.2020, https://www.mkm.ee/sites/default/files/avalike_teenuste_korraldamise_roheline_raamat.pdf

Awad, M., A. (2005). *A Comparison between Agile and Traditional Software Development Methodologies.* The University of Western Australia. Kasutatud 22.01.2020, http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.464.6090&rep=rep1&type=pdf

Balaban, M., Strum, A. (2018). *Software engineering lap: an essential component of a software engineering curriculum.* Software Engineering: Software Engineering Education and Training (ICSE-SEET). IEEE/ACM. DOI: 10.1145/3183377.3183395

Bano, M., Imtiaz, S., Ikram, N., Niazi, M., Usman, M. (2012). Causes of requirement change - A systematic literature review. *IET.* DOI: 10.1049/ic.2012.0003

Baur, N. (2019). *Linearity vs. Circularity? On Some Common Misconceptions on the Differences in the Research Process in Qualitative and Quantitative Research.* Frontiersin. DOI: 10.3389/feduc.2019.00053

Beck, K., Cockburn, A., Jeffries, R., ja Highsmith, J. (2001). *Agile Manifesto*. Kasutatud 15.03.2020, http://www.agilemanifesto.org

Bigelow, S.J. (2019). *How to tame ever-changing requirements in software development*. Kasutatud perioodil 03.03.2020 kuni 08.03.2020, https://searchsoftwarequality.techtarget.com/tip/How-to-tame-ever-changing-requirements-in-software-development

Bourque, P., Failey, R.E. (2014). *SWEBOK V3.0. Guide to the Software Engineering Body of Knowledge*. IEEE Computer Society. Kasutatud perioodil 22.01.2020 kuni 19.03.2020, https://cs.fit.edu/~kgallagher/Schtick/Serious/SWEBOKv3.pdf

E-estonia. (2019). *Estonia to launch U-space services in partnership with AirMap*. Kasutatud 19.03.2020, https://e-estonia.com/estonia-to-launch-u-space-services-in-partnership-with-airmap/

E-estonia. (2019). Government Cloud. Kasutatud 21.05.2020, https://e-estonia.com/solutions/e-governance/

Eesti Kvaliteediühing. (2020). Kasutatud perioodil 23.04.2020 kuni 24.05.2020, https://www.eaq.ee/

Eesti Rahvusringhääling. (2018). *Lennuametil on droonilennutajatele hea uudis*. Kasutatud 23.03.2020, https://www.err.ee/831105/lennuametil-on-droonilennutajatele-hea-uudis

Eesti Rahvusringhääling. (2019). *Riik valmistub laialdase drooniliikluse alguseks*. Kasutatud 21.03.2020, https://www.err.ee/974418/riik-valmistub-laialdase-drooniliikluse-alguseks

EL regulatsioon 2019/945. (2019). *Regulation on UAS operations in „open“ and „specific“ categories*. Euroopa Komisjon. Kasutatud perioodil 22.01.2020 kuni 19.03.2020, https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX:32019R0947

Gotel, O., Cleland-Huang, J., Hayes, J.H., Zisman, A., Egyed A., Grünbacher, P., Dekhtyar, A., Antoniol, G., Maletic, J., Mäder, P. (2012). *Traceability Fundamentals*. Software and Systems Traceability. Springer. DOI: 10.1007/978-1-4471-2239-5_1

Grebennikov, M. (2018). *Managing Changing Projects Requirements*. Kasutatud 01.02.2020, https://www.digiteum.com/changing-project-requirements

Györkös, J. (1994). *Measurement in software requirements specification process*. University of Maribor, 40, 893 - 896. DOI: 10.1016/0165-6074(94)90063-9

Himma, M. (2018). *E-riigi arhitekt: ükski infosüsteem pole kunagi valmis ega lõpuni turvaline*. Kasutatud perioodil 24.04.2020 kuni 25.04.2020, https://novaator.err.ee/872918/e-riigi-arhitekt-ukski-infosusteem-pole-kunagi-valmis-ega-lopuni-turvaline

Hussain, A., Kamal, F., Mkpojiogu, E. (2016). *The Role of Requirements in the Success or Failure of Software Projects*. DOI: 10.1063/1.4960886

IEEExplore. (2020). *Standards*. Kasutatud perioodil 18.04.2020 kuni 24.05.2020, https://ieeexplore.ieee.org/browse/standards/collection/ieee

ISKE. (2017). *Infosüsteemide kolmetasandilise etalonturbe süsteem Riigi Infosüsteemi Amet*. Kasutatud perioodil 03.03.2020 kuni 19.03.2020, https://www.ria.ee/sites/default/files/content-editors/ISKE/iske_rakendusjuhend.pdf

Jayatilleke, S., Lai, R. (2017). *A systematic review of requirements change management.* Information and Software Technology, 2017, 1-23. DOI: 10.1016/j.infsof.2017.09.004

Kalmus, V., Masso, A., Linno, M. (2015). Kvalitatiivne sisuanalüüs. Kasutatud 24.05.2020, http://samm.ut.ee/kvalitatiivne-sisuanalyys

Kamuni, S. K. (2015). *Study of Factors that Induce Software Project Overrun Time*. Cloud State University. Kasutatud perioodil 26.02.2020 kuni 03.03.2020, https://pdfs.semanticscholar.org/174e/886a305d871392c0476dfe4803dc57722629.pdf

Kivimaa, K. (2014). *Agiilsete tarkvaraarendusmeetodite võrdlus*. Seminaritöö. Tallinna Ülikool. Kasutatud 18.02.2020, http://www.cs.tlu.ee/teemaderegister/get_file.php?id=326

Koodivaramu. (2020). *Digiriigi Arhitektuurinõukogu*. Kasutatud perioodil 22.02.2020 kuni 13.03.2020, https://koodivaramu.eesti.ee/e-gov/cfr

Kütt, A. (2019). *Õppetunnid suurtest IT projektidest*. Kasutatud 03.12.2019, https://www.youtube.com/watch?v=e2kC0fqpV5k&feature=youtu.be&fbclid=IwAR1HJyrfxCzXZDgAzRW0IhB2SmkWNqufr6_ACJVIpqX5YfggUmfdMCTa_co

Langley, G., Nolan, K., Nolan, T., Norman, C., Provost, L. (1996). *The Improvement Guide.*

Lauesen, S. (2002). *Software requirements. Styles and techniques*. Harlow. Addison-Wesley.

McGee, S., Grees, D. (2012). *Towards an understanding of the causes and effects of software requirements change: two case studies*. Springer. DOI: 10.1007/s00766-012-0149-0

Meso, P., Jain, R. (2016). *Agile Software Development: Adaptive Systems Principles and Best Practices.* Information Systems Management. DOI: 10.1201/1078.10580530/46108.23.3.20060601/93704.3

Milani, F. (2019). *Digital Business Analysis*. Springer Nature Switzerland AG.

Mäder, P., Egyed, A. (2015). *Do developers benefit from requirements traceability when evolving and maintaining a software system? Empirical Software Engineerin*g 20, 413–441. DOI: 10.1007/s10664-014-9314-z

NSPCC Digital Team. (2018). *The value of prototyping everything*. Medium. Kasutatud 24.05.2020, https://medium.com/@thedigitaldunk/the-value-of-prototyping-everything-4148d01094f6

Pardo, T.A., Scholl, H.J. (2002). *Walking Atop the Cliffs: Avoiding Failure and Reducing Risk in Large Scale E-Government Projects*. IEEE Computer Society. Kasutatud 03.03.2020, http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.103.2373&rep=rep1&type=pdf

Pau, A. (2017). *Helmes: mida teha, et suured IT-projektid lõhki ei läheks*. Kasutatud 24.05.2020, https://tehnika.postimees.ee/4209173/helmes-mida-teha-et-suured-it-projektid-lohki-eilaheks?_ga=2.185390634.2094272993.1517050293-1410815516.1471354840

Pohl, K. (2010). *Requirements Engineering. Fundamentals, Principles, and Techniques*. Springer.

Reinsalu, U. (2018). *Riik on jäänud IT-süsteemide pantvangi.* Äripäev. Kasutatud 22.01.2020, https://www.aripaev.ee/uudised/2018/10/09/riik-on-jaanud-it-susteemide-pantvangi

Riigi Infosüsteemi Amet. (2020). Kasutatud 18.05.2020, https://www.ria.ee/et/riigi-infosusteem/x-tee.html

Riigi IT arhitektuuri raamistik. (2007). Majandus- ja Kommunikatsiooniministeerim. Kasutatud 27.12.2019, https://www.mkm.ee/sites/default/files/riigi_it_arhitektuur.pdf

Riigi IT koosvõime raamistik. (2007). Majandus- ja Kommunikatsiooniministeerim. Kasutatud 27.12.2019, https://www.mkm.ee/sites/default/files/riigi_it_koosvoime_raamistik.pdf

Riigikontroll. (2019). *Riigikontrolli auditi aruanne - valiku sektori tarkvaraarenduse projektide juhtimine.* Kasutatud perioodil 28.11.2019 kuni 24.05.2020, https://www.riigikontroll.ee/Suhtedavalikkusega/Pressiteated/tabid/168/ItemId/1077/amid/557/language/et-EE/Default.aspx

Riigihangete register. (2020). Rahandusministeerium. Kasutatud perioodil 27.11.2019 kuni 26.04.2020, https://riigihanked.riik.ee

Royce, W.W. (1970). *Managing the development of large software systems*. IEEE Wescon. Kasutatud 22.01.2020, http://www-scf.usc.edu/~csci201/lectures/Lecture11/royce1970.pdf

Rämmer, A. (2014). *Valimi moodustamine*. Kasutatud 24.05.2020, http://samm.ut.ee/valimid?fbclid=IwAR2D54knL0nSX5puiRq-giOA6n-fSQ8MrgkIKASa3KFXlDWzU2Ty15dBosE

Sajjad, U., Hanif, M.Q. (2010). *Issues and Challenges of Requirement Elicitation in Large Web Projects*. Blekinge Institute of Technology. Sweden. Kasutatud 03.03.2020, http://www.diva-portal.org/smash/get/diva2:830517/FULLTEXT01.pdf

Sommerville, I. (2016). *Software Engineering. Tenth edition*. Pearson. Kasutatud perioodil 27.12.2019 kuni 24.04.2020, http://dinus.ac.id/repository/docs/ajar/Sommerville-Software-Engineering-10ed.pdf

Struktuurifond. (2020). Euroopa Liidu Struktuuritoetus. Kasutatud 24.05.2020, https://www.struktuurifondid.ee/et/mis-on-struktuuritoetus

Vabariigi Valitsuse 01. juuli 2014. a Perioodi 2014–2020 struktuuritoetuse seadus - STS2014_2020, Riigiteataja. Kasutatud 29.04.2020, https://www.riigiteataja.ee/akt/113032019111?leiaKehtiv

Vabariigi Valitsuse 14. juuni 2017. a Riigihangete seadus. Riigiteataja. Kasutatud perioodil 28.04.2020 kuni 25.05.2020, https://www.riigiteataja.ee/akt/101072017001?leiaKehtiv

LISA 1 - Intervjuu kava

SISSEJUHATUS

1. Oled sa nõus, et kasutan antud intervjuu tulemusi koos sinu nime ning ametikohaga? Kui ei, kas võib viidata kui avaliku sektori spetsialistile?
2. Palun räägi endast ja oma kogemusest avaliku sektori tarkvaraarenduste osas - kui kaua, missuguste arendusprojektidega oled kokku puutunud?

MUUTUVAD NÕUDED

1. Kuivõrd tihti on Sinu kogemuses ette tulnud nõuete muutumist arendusprotsessi vältel?
2. Kas projektide puhul, kus esineb muutuvaid nõudeid, on mingeid ühiseid jooni? Palun selgita.
3. Millest on Sinu hinnangul nõuete muudatused peamiselt põhjustatud?
4. Palun nimeta mõningaid muutuvate nõuete riskifaktoreid, millele peab kindlasti eriliselt tähelepanu pöörama?
5. Missugustes nõuetes peamiselt muudatused ilmnevad Sinu hinnangul? On see projektide lõikes erinev?
6. Missugused tehnikad on kõige mõjusamad muutuvate nõuete ohjamiseks?
7. Mil määral on võimalik arendajal nõuete muudatusi ette näha?
8. Mil määral on võimalik muudatusi ära hoida, suunates klienti?
9. Missugused on avaliku sektori praktikad pidevalt muutuvate tarkvaranõuete ohjamiseks/juhtimiseks?
10. Missuguseid protsesse ja tehnikaid kasutatakse nõuete muutuste juhtimiseks?
11. Missugused on sinu arvates muutuvate nõuete juhtimise protsessi suurimad väljakutsed?

KOGEMUSED JA PRAKTIKAD NÕUETE MUUDATUSTE JUHTIMISE OSAS

Näiteks on MKM arendusprojektidest saadud õppetundide põhjal lisanud arenduskorda nõude kasutada projektides tehtud töödest ja arendusprotsessist parema ülevaate saamiseks MKMi projektijuhtimistarkvara, samuti kasutada tarkvara koodihalduse lihtsustamiseks MKMi koodihoidlat (Riigikontroll, 2019).

1. Missuguste vahendite ja lahenduste abil MKMis arendusprojekte juhitakse?
2. Missugused on senised kogemused - mis on SInu arvates toiminud ja mis osas näed, et täna enam selliselt mitte?
3. Missuguseid meetrikaid/mõõdikuid MKMis kasutatakse arendusprojektide tulemuslikkuse hindamiseks?
4. Kuidas läheneb MKM IT-osakond muutuvate nõuete juhtimisse?
5. Mis on levinud praktikad?

Karu (2019) toob oma artiklis järgmist: Ülesanne välja mõelda, kuidas riigi IT-süsteemide arendust olulisel määral paremuse poole liigutada, lasub MKM-il. Seninähtu ja –kogetu põhjal võib oletada, et efektiivsete lahenduste leidmiseks vajab praegune tõsiste puudujääkidega lähenemine märkimisväärset ümberkorraldust.

6. Kuidas te oma meeskonnas (MKM IT meeskond) näete, et seda protsessi saaks parendada?
7. Mil määral kõlmab antud teema endas ka muudatuste juhtimise funktsiooni arendamist?
8. Missuguseid tegevusi olete selle tarbeks juba ellu viinud ja mis on veel plaanis?

ÜLDISELT INFOSÜSTEEMI ARENDAMISEST

Avalikus sektoris on kasutusel mitmeid erinevaid tarkvara arendusmetoodikaid, mida omavahel kombineerides saadakse sobiv lahendus - lisaks koskmudelile iteratiivne arendusmudel, agiilne tarkvaraarendusmudel, DevOps (ingl development and operations) mudel jt (Riigikontroll, 2019).

1. Missugused tarkvaraarendusmetoodikad on kasutusel MKM arendusprojektide puhul?
2. Mille alusel metoodika valitakse? Millest lähtutakse?
3. Miks just see?
4. Millest see sõltub?
5. Millised arendusmetoodikad on Sinu arvates kõige edukamaks seni osutunud avaliku sektori infosüsteemide arendamisel?
6. Milles see sõltus ja miks?Saad sa mõne konkreetse näite tuua?
7. Miks ja mis osas riigihanke protseduurireeglid ei võimalda agiilset arendusmetoodikat kasutada?

LISA 2 - Nõuete muudatuste juhtimise protsess

M**uudatuste juhtimise protsessi kirjelduse**, mis tagab muudatuste süsteemse haldamise põhimõtted, eesmärgiga sarnase või seotud informatsiooni selgeks ja süsteemseks hoiustamiseks kindlas kokkulepitud asukohas koos viidetega konkreetsele tõendusdokumendile reaalajas - nt ajakava, tööplaan, analüüsidokumendid jne. Mudeli olen tuletanud kvaliteedijuhtimises tuntud PDSA (ingl *Plan-Do-Study-Act*) mudelist (Langley, Nolan, Nolan, Norman, Provost, 1996: 10)

Joonis 5. Tarkvaranõuete muudatuste juhtimise protsess (Autori loodud).

Protsessi esimene samm on muudatusettepaneku registreerimine ehk teadvustamine ja sellekohase info edastamine projektijuhile, sest tulemusliku muudatuse juurutamise huvides on oluline, et analüüimisel võetakse arvesse kogu asjakohast infot konkreetse muudatusettepaneku kohta. Kindlasti loob väärtust standardiseeritud vormi kasutamine, kus võiksid olla vähemalt järgmised lahtrid:

- Muudatusettepaneku tegija, et vajadusel täpsustada ja lisainformatsiooni saada;
- Muudatusettepaneku tegemise kuupäev;
- Viide olemasolevale nõudele (?);
- Algpõhjuse analüüs (?) - mis muutub;
- Ülevaade muudatusest - before ning after olukorra kirjeldus, võimalusel visuaalselt;
- Nõude maksumus/ kulud;
- Asjakohased isikud, kes peavad olema kaasatud analüüsi ning kes peavad saama sellest informeeritud.

M**uudatusettepaneku analüüs**i ülesanne on teha kindlaks missugust väärtust antud muudatus loob ning sellest tingitud mõjuala, ehk kui palju ja missugused süsteemi komponendid ja/või nõuded võivad antud muudatusettepanekust lähtuvalt saada mõjutatud. Analüüsist lähtuvalt toimub **otsustamine** misjärel muudatus kas juurutatakse või mitte. Juhtrühma tasandilt võiks olla määratud rühm eksperte, kellel on piisav pädevus otsustamiseks, kuhu peaks kindlasti kuuluma nii projektijuht kui ka loodava

süsteemi peakasutaja, kelle ülesanne on analüüsida muudatusettepanekuid ning teha lähtuvalt otsus. Muudatuse rakendamise faas hõlmab endas ka nõuete dokumendi (või -registri) uuendamist ning asjakohaste isikute informeerimist.

Lihtlitsents lõputöö reprodutseerimiseks ja üldsusele kättesaadavaks tegemiseks

Mina, Anni Adamson

1. annan Tartu Ülikoolile tasuta loa (lihtlitsentsi) minu loodud teose "Infosüsteemi arendamise põhimõtted muutuvate nõuete korral Eesti avalikus sektoris", mille juhendajad on Reeni Mikkelsaar ja Maris Männiste, reprodutseerimiseks eesmärgiga seda säilitada, sealhulgas lisada digitaalarhiivi DSpace kuni autoriõiguse kehtivuse lõppemiseni.

2. Annan Tartu Ülikoolile loa teha punktis 1 nimetatud teos üldsusele kättesaadavaks Tartu Ülikooli veebikeskkonna, sealhulgas digitaalarhiivi DSpace kaudu Creative Commonsi litsentsiga CC BY NC ND 3.0, mis lubab autorile viidates teost reprodutseerida, levitada ja üldsusele suunata ning keelab luua tuletatud teost ja kasutada teost ärieesmärgil, kuni autoriõiguse kehtivuse lõppemiseni.

3. Olen teadlik, et punktides 1 ja 2 nimetatud õigused jäävad alles ka autorile.

4. Kinnitan, et lihtlitsentsi andmisega ei riku ma teiste isikute intellektuaalomandi ega isikuandmete kaitse õigusaktidest tulenevaid õigusi.

Anni Adamson

27.05.2020