Tallinna Ülikool
Informaatika Instituut

ETTEVÕTTE TARKVARAARENDUSE ÜLEVIIMINE KOSKMEETODILT AGIILSELE ELION ETTEVÕTTED AS NÄITEL

Magistritöö

Autor: Taavi Sepp

Juhendaja: Marek Kusmin

Autor: ……..... „ … ― …….. 2010.a.
Juhendaja: …… „ … ― …….. 2010.a.
Instituudi juhataja: ……..... „ … ― …….. 2010.a.

Tallinn 2010

Autorideklaratsioon

Deklareerin, et käesolev magistritöö on minu töö tulemus ja seda ei ole kellegi teise poolt varem kaitsmisele esitatud. Kõik töö koostamisel kasutatud teiste autorite tööd, olulised seisukohad, kirjandusallikatest ja mujalt pärinevad andmed on viidatud.

....
(kuupäev) (autor)

Sisukord

1. Sissejuhatus

1990-ndate aastate lõpus hakati maailmas tähelepanu pöörama mitmetele uutele tarkvaraarenduse metoodikatele. Nendes kombineeriti erinevates kombinatsioonides vanu ideesid uutega, kuid ühisosaks oli tihe suhtlemine (eelistatult näost näkku) arendusmeeskonna ja tellija vahel, arenduse inkrementaalne ülesehitus, kiire ja sagedane kasutaja jaoks uue funktsionaalsuse lisamine, vähene sõltuvus pidevalt muutuvatest ärilistest nõuetest ning väike ning iseorganiseeruv, tiheda koostööga arendusmeeskond. Selliseid metoodikaid hakati kutsuma **agiilseteks tarkvaraarenduse metoodikateks** (inglise keeles *agile software development methodologies*). Eestikeelses terminoloogias on neid nimetatud ka „väledateks" või „nobedateks", kuid vastavaid metoodikaid kasutavate praktikute ringis on enim levinud termin „agiilne", mistõttu edasises töös kasutan terminitest viimast.

2001. aasta veebruaris avaldasid 17 tarkvaraarendajat agiilse tarkvaraarenduse manifesti („Manifesto for Agile Software Development"), mis sõnastab agiilse arenduse põhitõed (tõlge Leis 2001 mõningate minupoolsete kohendustega):

„Me tahame luua uusi paremaid tarkvara arendusmetoodikaid. Me hindame

- *enam inimesi ja nendevahelisi suhteid kui protsesse ja arendusvahendeid*
- *enam töötavat tarkvara kui täielikku dokumentatsiooni*
- *enam kliendi osalust arenduses kui lepingute koostamist*
- *enam muudatustele reageerimist kui plaani järgimist.* "

Agiilse tarkvaraarenduse omadustest on kliendi jaoks kõige olulisemaks see, et arendaja annab esimese osa töötavast tarkvarast üle väga ruttu ja sellega saab soovi korral juba turule tulla või toodangusüsteemides kasutusele võtta. Tarkvara ehitatakse üles nii, et saab pidevalt ja tihti lisada uut funktsionaalsust väikeste töötavate osadena. Selleks lahutatakse suured ülesanded väiksemateks, mis saavad eksisteerida iseseisvalt ning planeeritakse koos kliendiga järgmise iteratsiooni sisu. Kliendi jaoks on oluline, et tema otsustab, milline funktsionaalsus realiseeritakse järgmise iteratsiooniga. Ühe iteratsiooni pikkus on tavaliselt üks kuni neli nädalat. Iga iteratsioon on justkui väike iseseisev tarkvaraarendusprojekt, mis

algab plaanimisega ja lõpeb testimise ning kliendipoolse aktsepteerimisega. Kuna klient näeb iga uut valminud funktsionaalsust kohe, siis see minimeerib riski, et projekti lõpus leitakse olulisi tema vajadustele mittevastavusi. Samuti on inkrementaalse arenduse korral lihtsam reageerida ja kohaneda järsku kerkinud muudatusvajadustega, sest nendega saab järgmist iteratsiooni planeerides arvestada. Muudatused võivad tuleneda väljastpoolt (regulatsioonid, muutunud turusituatsioon, uued tehnoloogiad) või seestpoolt (äritellija on kogemusest õppinud). Agiilsed metoodikad leiavad, et muudatused on normaalne nähtus ja nendega tuleb arvestada. Watts Humphrey on öelnud „Kui sa ei saa planeerida täpselt, siis planeeri tihti". Agiilsed metoodikad talitavad selle põhimõtte järgi ja on iteratiivse planeerimise võtnud iteratiivse arenduse osaks.

Erinevatel osapooltel (kasutajad, analüütik, programmeerija) on loodavast rakendusest oma, teistest erinev, arusaam. Iteratiivne arendus kiirendab ühise ettekujutuse kujunemist, sest iga iteratsiooniga saab pilt üha selgemaks ja astutakse samm lähemale lõplikule osapoolte arvates parimale lahendusele. Agiilses tarkvaraarenduses on meeskonnad väikesed (5-9 inimest), suuremate projektide puhul tekivad mitmed väiksemad meeskonnad. Meeskonnad on iseorganiseeruvad ja liikmed enamasti võimelised täitma erinevaid rolle. Liikmed võtavad aktiivselt ise endale ülesandeid ja vastutavad meeskonna ees nende täitmise eest (kontrollijaks ei ole juht).

Vastutus otsuste eest, mida arendatakse (millist funktsionaalsust), on kliendil. Seega on ka projekti vastutus (eelarve, kestus) kliendil, kuna otsuseid (näiteks kas lisada algselt plaanimata funktsionaalsust ja ületada eelarvet või loobuda selle arvelt millestki varemplaanitust) teeb klient.

Kokkuvõttes, agiilse tarkvaraarenduse eestvedajad näevad selle mõtteviisi peamise eelisena võimalust tuua töötav tarkvara välja kiiresti ning erinevate praktikate abil kiirendada ka kogu projekti valmimist, hoides sellega kokku aega ja raha võrreldes monumentaalsete metoodikatega.

Elion Ettevõtted AS-i (edaspidi Elion) jaoks üks olulisemaid probleeme IT ja tootearenduses on madal arendusvõimekus. Võimaliku lahendusena nähakse teostusmeeskonna suurendamist ja/või arendusmetoodika muutmist. Strateegiline otsus e-kanalite üleminekuks agiilsele arendusele on juhtkonna poolt tehtud ning töö eesmärk on leida, milline agiilne arendusmetoodika (või erinevate metoodikate

süntees) sobiks Elioni jaoks kõige paremini ning välja töötada lahendused valitud metoodika juurutamiseks.

Magistritöö teema on ettevõtte jaoks väga oluline, kuna ettevõttel puudub varasem sarnane metoodika muutmise kogemus. Juhtkonna soov on, et magistritöö selles kaasa aitaks.

Ericsson AB-l on sarnane arendusmetoodika muutmise kogemus Rootsist, kus peamiselt muudatusnõuete hulga minimeerimiseks ja kontrolli alla saamiseks valiti tarkvaraarendusprotsessile agiilne lähenemine ning see aitas lahendada probleemi (Dzamashvili Fogelström, Gorschek, Svahnberg ja Olsson 2010). Agiilse meetodi esmarakendamine tõstab projekti riskitaset märkimisväärselt (Koch 2005, 53). Meie transitsioonijuhtumi puhul oleme katsetanud agiilset lähenemist paari väiksema projekti puhul ja oleme ette võtmas suuremaid ja kriitilisemaid projekte. Siin tuleb mängu sama allika soovitus puhkudeks, kui senise meetodi kasutamine on ennast tõestanud ebaefektiivsena. Sellisel juhul võib riskide pehmendamiseks olla asjakohane ka uue meetodi esmarakendamine. Eelneva kogemuse põhjal on selge, et seniseid meetodeid kasutades oleks uue põlvkonna klienditeeninduse e-keskkondade loomine mitmeaastane ja multimiljonilise maksumusega projekt.

Ettevõtte erinevates arendusmeeskondades kasutatavad metoodikad on erinevad, seetõttu piiritlen skoobi ettevõtte e-kanalite arendusega. Töö käigus analüüsin erinevaid agiilseid metoodikaid kasutades Alan Kochi (2005) poolt pakutud hindamismeetodit ja sünteesin neist ettevõtte ja antud projekti eripära arvestades rakendatava arendusmetoodika. Kochi poolt pakutud hindamismeetodi valiku põhjuseks oli meetodi võime viia erinevate metoodikate praktikad sarnasele võrreldavale alusele, kusjuures erinevate osade kaalud on ühetaolised. Töötan kvalitatiivset analüüsi kasutades välja ka põhimõtted, kuidas valitud metoodika kavatsetakse juurutada ning millele selle käigus tähelepanu pöörata, millised on kaasnevad ohud ja kuidas neid vältida. Kvalitatiivse analüüsi valisin juurutuspõhimõtete uurimismeetodiks, kuna eksisteerivad mitmed sarnased arendusmetoodika muutmise kogemused teistes ettevõtetes ja on mõistlik analüüsida sealseid kogemusi, arvestades Elioni eripäradega.

Mõistmaks paremini autori tausta magistritöö kirjutamisel, kirjeldan seda lähemalt. Olen töötanud kümme aastat Elionis, neist viimased viis IT arenduse valdkonnas peamiselt analüütikutest koosneva meeskonna juhina. Minu peamine tegevus igapäevase meeskonna juhtimise kõrval on töö korraldamine arenduspartneritega ja siseklientidega ning osalemine pidevas arendusprotsessi kohandamises. Elion on

telekommunikatsiooni valdkonnas tegutsev suurettevõtte, kus tarkvaraarendus on ettevõtte arenduse (*Research and Development*) osaks, klientidele tarkvaraarendusteenust ei pakuta. Arendusmeeskond tegeleb muuhulgas kasutajate töölaudade, finantssüsteemide, kliendihaldussüsteemide, e-keskkondade ja tootearendusega.

2. Tänane olukord

Käesolevas peatükis kirjeldan ettevõtte e-kanalite komponente ja tehnilist ülesehitust, et oleks parem ülevaade töös käsitletud transitsiooni mahust ja tehnilisest vaatest. Samuti kirjeldan tänases e-kanalite tarkvaraarenduse protsessis osalevate inimeste rolle, arenduste metoodikat ja koostöömudelit arenduspartneriga, selgitamaks arendusmetoodika muutmise algseisu. Peatüki lõpetuseks hindan ettevõtte organisatsioonikultuuri sobivust agiilseks arenduseks, et mõista metoodika muutusega kaasnevat võimalikku vajadust mõjutada ka organisatsioonikultuuri.

2.1 E-kanalite kirjeldus

Elioni e-kanalite peamisteks komponentideks on avalik veeb, siseveeb ning klientidele suunatud e-teenindus. Avalikus veebis on ettevõtte üldinfo, teenuste tutvustused ja abimaterjalid/juhendid klientidele. Siseveebis on teenindusuudised ja teenuste tutvustused ning abimaterjalid/ja juhendid teenindaja võtmes kirjeldatuna. E-teeninduses näeb klient oma arvete ja laekumiste infot, olemasolevate teenuste parameetreid, võimalus on tellida teenuseid ja lisateenuseid või neist loobuda, lepinguid sõlmida või sõlmitud lepingutega tutvuda ja teha saab veel mitmeid muid toiminguid.

2.2 E-kanalite tehniline kirjeldus

Tehniliselt on avaliku veebi taga Java platvormil sisuhaldusrakendus ning Oracle'i andmebaas. Siseveebi sisu hallatakse Javal baseeruval infoportaali rakendusel (kasutades Oracle'i andmebaasi). E-teenindus on samuti Java rakendus, mis üle PL/SQL API-de suhtleb teiste süsteemidega, kust saab funktsionaalsuseks vajaliku info. Andmebaasi osa baseerub jällegi Oracle'il. Kõik nimetatud Java rakendused on arendatud spetsiaalselt meie vajadustest lähtuvalt koostöös arenduspartneriga, kes pakub meile analüüsi- ja programmeerimisteenust.

2.3 Rollid, muudatuste metoodika kirjeldus ja koostöömudel arenduspartneriga

Rollid

Äritellija on isik, kes koondab erinevate osapoolte huvid ja kirjeldab neid arvestades muudatuste lähteülesande, mida sisuliselt (äriliselt) teha on vaja ja mis on eesmärk, mida muudatusega soovitakse saavutada.

Elioni analüütik on isik, kes töötab läbi ärilise lähteülesande, toob välja ülesande täiendusvajadused, pakub alternatiivseid lahendusvariante soovitud tulemuse saavutamiseks. Tema disainib süsteemidevahelise andmevahetuse mudeli ning edastab analüüsi tulemused arenduspartneri analüütikule ja võtab vastu teostuse.

Arenduspartneri analüütik on isik, kes rakendusesiselt analüüsib teostusvajadust, pakub vajadusel kinnitamiseks välja lahenduse HTML-prototüübi näol, püstitab programmeerijale lähteülesande ning kirjeldab/täiendab spetsifikatsioone.

Muudatuste metoodika ja koostöömudel

Väiksemate muudatuste korral, kui sisu on üheselt selge, tuvastab Elioni analüütik muudatusvajaduse ja seotud süsteemi ning tellib vastava süsteemi muudatuse teostuse. Siin ei saa rääkida ühest või teisest metoodikast, kuna tegu on niivõrd triviaalsete asjadega.

Suuremate muudatuste korral tutvustab Elioni analüütik muudatust partneri analüütikule, võimalikud lahendused arutatakse koos läbi (kaasates vajadusel äritellija või teisi huvitatud osapooli) ja partneri analüütik pakub selle põhjal HTML prototüübi näol välja lõpliku lahendusvariandi, mida Elioni poolse heakskiidu järgselt teostama asutakse. Kogemused näitavad, et tellija jaoks annab parima ülevaate, mis ta tellinud on, prototüüp. Tänane mudel, mis tugineb prototüüpimisele, on prototüübi koostamise etapis lähedane agiilsele, sest arendaja jaoks tulevad välja lahtised kohad, mille lahenduskäike arutame üsnagi jooksvalt, samuti saab tellija prototüübiga tutvudes välja tuua täiendusvajadused. Tegemist ei ole otseselt kosemudeliga, sest lähteülesande koostamine ei toimuilma arendajaid kaasamata ning seda ei lukustata enne arendajale üleandmist. Küll pole tegu ka agiilse arendusega, sest prototüübi kinnitamise järgselt enam funktsionaalsuste teostamise ja teostusviiside üle arutelu ei toimu, välja arvatud

erandjuhud. Abstraktse analüüsi asemel prototüübi pakkumine on agiilsete meetodite poolt eelistatud lähenemine (Koch 2005, 135), XP soovitus on aga prototüübi asemel kohe minimaalsete kulutustega töötav rakendus välja pakkuda.

Testimine on mitmeosaline. Partneri arendusmeeskonnas on programmeerijatele lisaks ka testija, kes testib koodi kvaliteeti. Lisaks läbib iga versioon ka Elioni kvaliteediosakonna poolse testimise, mille käigus testitakse nii kvaliteeti kui ka nõuetele vastavust. Elioni testija teeb ettepanekuid funktsionaalsuse muutmiseks, kui oskab pakkuda olemasolevast paremaid lahendusi. Elioni testija viib läbi ka aktsepteerimistesti tellijaga, kes kinnitab, kas valminu on vastavuses vajadustega (just vajadustega, sest tellija jaoks võib olla arusaam vajadusest vahepeal muutunud). Seega on meie praeguses testimise töökorralduses agiilseid elemente.

2.4 Organisatsioonikultuuri sobivus agiilseks arenduseks

Agiilsele arendusele ülemineku üks kriitilise võtmeaspekte on organisatsioonikultuur. Asendades hierarhilise struktuuri koostööle sunnatud struktuuriga, võib organisatsioonikultuuril selle muudatusega kohanemine võtta märkimisväärse aja (pigem aastaid) (Koch 2005, 38).

Organisatsioone saab jagada kultuuri põhjal mehhanistlikeks ja orgaanilisteks (Burns, Stalker 1994). Burns ja Stalker toovad välja hulga omadusi, mis iseloomustavad üht või teist. Lisasin antud omadusi kokkuvõtvasse tabelisse (tabel 1) kolmanda veeru, kus hindasin oma kogemuse põhjal Elioni arenduse organisatsioonikultuuri. Suhtlemine konkreetsete projektide sees ja juhtimisahelad on meil lähedased orgaanilisele organisatsioonile, protsessid ja rollijaotus pigem mehhanistlikule.

Mehhanistlik	**Orgaaniline**	**Elioni arendus**
Spetsialiseeritus	Töötajad võtavad eri rolle	Pigem mehhanistlik
Võimuhierarhia	Meeskondade võrk	Orgaaniline
Vertikaalne kommunikatsioon	Lateraalne kommunikatsioon	Pigem orgaaniline

Ülemused ja alluvad	Konsulteerimine, mitte käsutamine	Orgaaniline
Juhised ja otsused	Info ja nõuanded	Orgaaniline
Lojaalsus ja käsutäitmine	Eesmärgile pühendumine	Orgaaniline
Töötajad töötavad omaette	Töötajad töötavad koos	Keskmine
Tsentraliseerimine	Detsentraliseeritus	Pigem mehhanistlik
Reeglid ja standardsed operatsioonid	Nõtked tööprotsessid	Pigem mehhanistlik
Kirjalik suhtlus	Verbaalne suhtlus	Keskmine

Tabel 1. Orgaanilise ja mehhanistliku organisatsiooni võrdlus

Suhtlemine konkreetsete projektide sees ja juhtimisahelad on meil lähedased orgaanilisele organisatsioonile, protsessid ja rollijaotus pigem mehhanistlikule. Agiilsed meetodid sobivad pigem orgaanilisele organisatsioonile (Vinekar, Huntley 2010) ja kuna omaduste hindamisel kogunes rohkem hinnanguid „orgaaniline" või „pigem orgaaniline", siis saab väita, et organisatsioonikultuur sobib agiilsete metoodikate juurutamiseks.

3. Agiilsete arendusmetoodikate Elionile sobivuse analüüs

Peatüki esimeses osas toon välja ettevõtte omapärad, millega analüüsis arvestada tuleb. Teises osas valin analüüsimiseks välja metoodikad ning analüüsin erinevate metoodikate sobivust ettevõtte kontekstiga, rühmitades metoodikate praktikaid agiilse arenduse manifesti printsiipide järgi. Kolmandas osas võtan kokku tulemused ja valin ettevõttele sobiva metoodika.

3.1 Ettevõtte omapärad

Metoodika valikul tuleb arvesse võtta ettevõtte konteksti. Kirjeldan käesolevas peatükis, milliseid ettevõtte konteksti eripärasid kavatsen metoodika valiku analüüsimisel arvesse võtta.

Elioni E-kanalite arendusmeeskonda on planeeritud kaks ettevõttes töötavat programmeerijat, kellel puudub vastava süsteemiga varasem kogemus. Ülejäänud arendajad kavatsetakse sisse osta teenusena väliselt partnerilt. Partneri valikul on projekti juhtrühmal vabad käed valida soovijatest sobivaim. Kuna suur osa funktsionaalsuse jaoks vajalikust informatsioonist päritakse teenuste kaudu liidestatud süsteemidest, siis oleks mõistlik kasutada eraldi analüütikuid, kellel on kogemused vastavate süsteemidega ja kes oskavad interpreteerida sisendite ja väljundite sisu. Arendusmeeskonnas saame arvestada iga alamprojekti jaoks analüütikuga, kahe kuni kolme inimesega. Testimise korralduses oleks eelistatud meetod, mille puhul moodultestimist viib läbi programmeerija ning projektis on ka spetsialiseerunud testija, kes teostab täisahela testimist kasutaja perspektiivist.

Ettevõtte arendusüksuses on protsessid standardiseeritud ja paigas. Inimesed on harjunud töötama projektimeeskondades, kus juhised ei tule mitte struktuurijärgselt juhilt, vaid projektimeeskonna seest. Projekti meeskondade komplekteerimine toimub vajalike kompetentside alusel ja ressursijuhtide poolt määramise kaudu. Ressursijuhid lähtuvad prioriteetide määramisel programmijuhtide (äripoolne otsustav roll) poolt projektidele määratud prioriteetidest – kõrgema prioriteediga projektid saavad planeeritud ressursi. Projekti liige on projektijuht, kes koostab ja hoiab värskena projektiplaani ning jälgib projekti püsimist plaanis. Projektijuht koordineerib projektiliikmete tegevust.

Ettevõttes teostatakse tarkvaraarendust ainult enda (siserakendused) ja oma klientide tarbeks (rakendused välisele kasutajale).

Arendustsükli (versioonide toodangusse viimise vahe) pikkuse eelistuseks on üks kuu, sest sama kestusega on ka tuumrakenduste, millega projektis loodav rakendus liidestub, arendustsükkel. Hoides neid tsükleid sünkroonis, on vähem tööd erinevaid süsteeme puudutavate muudatuste versioonivahetusse kaasamisega.

Pidev integratsioon (i.k. *continous integration*) on ettevõttes juurutamisel. Senine praktika on olnud, et arendaja viib uut funktsionaalsust arendusbaasi mitte koheselt

peale iga uue omaduse valmimist, vaid üldjuhul suurema kogumina, mistõttu tekib tellijal tutvumise võimalus hiljem. Tänu igakuisele väljalasete tsüklile läheb siis aga väga kiireks parandusi või muudatusi sisse viia.

3.1.1 *Probleemid, mida uus metoodika lahendama peab*

Elioni jaoks on muudatuse läbiviimise juures kõige olulisem saavutada parem arendusvõimekus. Probleem number üks, mida lahendama läheme, on ootelolevate arendustellimuste hulga vähendamine läbi kiirema arenduse ja parema koostöö tellijaga, et leida, millised arendused on tõesti oluline ära teha ja millised võib jätta kõrvale. Oluline on ka uute lahenduste turuletuleku kiiruse parandamine. Kolmas oluline probleem on kohanemine ebamääraste või muutuvate nõuetega.

Lahendamist vajab ka projektide läbiviimisel suhtlusviivitusest tekkiv ebaefektiivsus. Kui teine osapool pole töö käigus tekkinud küsimustele vastamiseks kohe kättesaadav, siis tekib ümberlülitustega kaasnev ajakulu. See puudutab nii tellijaga suhtlust kui ka projekti liikmete omavahelist suhtlust. Täiendav probleem, mis sellest lisaks ajakulule tuleneb, on versiooni ajakava ohtuseadmine, sest vähesel määral tegemise või vastamise aega nõudvad teemad jaotuvad pikka ajavahemikku suurte lünkadega, versioonitsükkel on aga väga tihe. Efektiivsuse saavutamiseks üks võimalik arengukoht oleks vähendada inimestel erinevate ülesannete vahel ümberlülitumisele kuluvat aega – võimalusel tuleks leida lahendus, kuidas inimesed saaksid keskenduda vähemale hulgale ülesannetele ja seda ühe ülesande kontekstis pikemaks pidevaks ajavahemikuks.

3.2 Agiilsete metoodikate valik ja ettevõtte kontekstiga sobivuse analüüs

Käesolevas töös on agiilsetest metoodikatest analüüsimiseks valitud välja alljärgnevad. Valikut tehes olid otsustavateks teguriteks nende levik ja aktuaalsus

- Scrum
- Extreme Programming (edaspidi XP)
- Lean Software Development (edaspidi LSD)
- Adaptive Software Development (edaspidi ASD)

- Feature Driven Development (edaspidi FDD)
- Dynamic Systems Development (edaspidi DSD)
- Crystal Clear
- Agile Unified Process (edaspidi AUP)

Analüüs on üles ehitatud Alan Kochi pakutud põhimõttel (Koch 2005), kus valitud metoodikate praktikaid võrreldakse ettevõttele sobivuse kontekstis empiiriliselt, jagades praktikad Agiilse arenduse manifestis kirjeldatud sobivate printsiipide kaupa ning lisades ühe manifestis kirjeldamata printsiibi. Töö kirjalikus osas on võrdluse aluseks ettevõttele sobivus ja olulisus, töö lisaks olevas töövihikus on võrdlusel natuke teistsugune fookus, ettevõtte kontekstis juurutamise lihtsus. Töövihiku tulemused võtab kokku tabel 2 kirjalikus osas. Analüüsi tulemusena leian Elioni jaoks sobiva metoodika või sünteesin sobiva metoodika.

Vaadates agiilse tarkvaraarenduse manifesti nelja postulaadi (väärtuse) võtmes, siis kõigil analüüsiks valitud metoodikatel ei ole iga postulaadi jaoks seonduvat praktikat, samuti on paljudel metoodikatel kokkulangevusi praktikate sisus. Allpool püüan välja tuua iga postulaadi võtmes, millised metoodikate ideed või praktikad on seotud vastava postulaadiga ning võrdlen neid olulisuse ja sobivuse põhjal ettevõtte kontekstiga.

3.2.1 Väärtustame enam inimesi ja nendevahelisi suhteid kui protsesse ja arendusvahendeid

Pealkirjas toodud postulaadiga on seotud neli printsiipi manifestist ning üks sõnastamata printsiip, mis on allpool paksus kirjas ning täpploendis välja toodud.

- **Projektide tuumaks on motiveeritud inimesed, keda tuleb usaldada**

Seda printsiipi toetavad praktikad on:

ASD-s „Projekti osalised on iseseisvad agendid" ja „Adaptiivne (eestvedamis-koostöö) juhtimismudel",

DSDM-s „Meeskonnale tuleb anda volitused tegutsemiseks",

LSD-s „Meeskonnale tuleb anda volitused tegutsemiseks – motivatsioon",

AUP-s „Meeskonna liikmed teavad, mida nad teevad“,

XP-s „Kogu meeskond“ ja „Koodi ühisomandus“,

FDD „Klassil (koodil) on konkreetne vastutaja“,

CC-s „Isiklik turvalisus“.

Need praktikad saab jagada kaheks rühmaks. Esimene rühm on projekti meeskonna liikmete õiguste ja vastutusega seotud ning praktikate põhimõte on, et liikmed on võrdsed, on volitatud tegutsema ja neid tuleb usaldada. Projekti vaates on siin esimese rühma metoodikate praktikate vahel hinnang võrdne, sest ettevõtte arendusprojektide põhimõtetega sobib väga hästi, et projekti osalised on võrdväärsed partnerid, on piisavate volitustega ja vastutavad projekti õnnestumise eest ühiselt. Kirjeldatud metoodikate praktikate vahel pole nii suuri erinevusi, et mõnda esile tuua või mõnele madalamat hinnangut anda. CC „Isikliku turvalisuse“ praktikat sellesse rühma lugeda ei saa, sest see ei kata meeskonnaliikme kohustusi ja muid õigusi peale turvalisuse erimeelsusi väljendada. Teises rühmas on koodi omanduse praktikad ning XP ja FDD on siin vastanduvatel seisukohtadel. Ettevõtte kontekstis on sobivam XP lähenemine, kuna väline arenduspartner ei jää sama koodiga edaspidi tööle, vaid edaspidiste muudatuste tegemine jääb ettevõtte enda programmeerijatele. Nii on „Koodi ühisomandus“ sobivam praktika, kuna ettevõtte programmeerijatel on parem ülevaade koodist tervikuna.

- **Iseorganiseeruvad meeskonnad**

Seda printsiipi käsitlevad praktikad on:

ASD-s „Oleta: projekti sissejuhatus“ ning „Adaptiivne tsükli planeerimine“,

DSDM-s „Meeskonnale tuleb anda volitused tegutsemiseks“,

XP-s „Planeerimismäng“,

FDD-s „Omaduspõhised meeskonnad“,

LSD-s „Meeskonnale tuleb anda volitused tegutsemiseks – iseotsustamine“,

Scrumis „Scrumi meeskond“,

CC-s „Fookus“.

Nende praktikate puhul ei ole meeskonnasisese iseorganiseerumise osas olulisi erinevusi. ASD, DSDM, LSD, CC toetavad seda printsiipi üldisemalt, Scrum ja XP lähevad täpsemaks. Siiski ei saa välja tuua ettevõtte konteksti arvestades olulisi erinevusi. FDD eristub veidi selle poolest, et meeskonnad moodustuvad koodi klassi omanikest, meie huvides aga ei ole koodi osadele omanikke anda, pigem sobib XP „Koodi ühisomanduse" praktika (vt ka eelmist printsiipi).

- **Parim suhtlusvorm on näost näkku vestlus (koospaiknemine)**

Seda printsiipi puudutavad praktikad on:

XP-s „Paarisprogrammeerimine", „Kogu meeskond", „Planeerimismäng" ja „hoonete strateegia",

Scrumis „Igapäevased Scrumi koosolekud",

CC-s „Osmootiline kommunikatsioon".

Selle printsiibi puhul on hinnangud erinevad. Scrumi järgi on igapäevane koosolek koht, kus vahetatakse omavahel infot, kuid seda hästi lühidalt (Schwaber, Beedle 2002, 40). Pole nõuet, et kõik peavad füüsiliselt kohal olema. Muus osas suhtlust ette ei kirjutata.

XP puhul on planeerimiskoosolek ja paarisprogrammeerimine XP lahutamatud osad, ilma nendeta ei saa XP-d XP-ks nimetada (Beck 1999). Samuti soovitab XP tungivalt, et kogu meeskond töötaks ühes ruumis ja vajadusel oleks koheselt saadav kontakt kliendi esindajaga. Tänapäeva tehnilised lahendused (kiirsuhtlus ehk i.k. *instant messaging* ja IP videokonverentsid (näiteks Skype'i vahendusel)) võimaldavad võrreldes XP algusaegadega vahetada infot reaalajas sellisel tasemel, et pidev koosviibimine pole enam nii oluline. Nii on ka kunagine praktika „Kohapealne klient" muundunud praktikaks „Kogu meeskond".

Scrumi koosolekupraktika on Elionis juurutamiseks sobiv, XP praktikad „Kogu meeskond" ja „Hoonete strateegia" on Elioni jaoks väärtuslikud, aga mõlemad rasked juurutada. Täna puudub tellijatel praktika, et nad ühele projektile nii suures mahus panustavad. Igal tellijal on mitmed erinevad projektid ja tavapraktika on, et ühele neist ei panustata üle 20 % ajast (empiiriline kogemus). XP soovitus, et kogu meeskond paikneks ühes ruumis, on keerukas juurutada seetõttu, et osa meeskonnast on arenduspartneri töötajad, osa Elioni töötajad. Senine praktika on olnud, et vaatamata tihedatele kohapealsetele ja IP videokonverentsil põhinevatele koosolekutele, töötavad välised arendajad oma ruumides, inimesed on sellega

harjunud. Küll ei ole ülesanne võimatu, Elionis leiab vajadusel ka sobiva meeskonnatöö ruumi.

CC toob välja, et ühes ruumis oleva meeskonna vahel toimub info liikumine justkui osmootiliselt, ilma et peetaks spetsiaalseid infovahetuse koosolekuid. XP ja CC praktika vahel valides on Elioni kontekstis sobivam XP lähenemine, et teostusmeeskond ilma tellija esindajata on ühes ruumis koos. Tellija pidevat juuresviibimist juurutada on meeskonna ühte ruumi saamisest oluliselt raskem.

- **Jätkusuutlik arendus, mis on võimeline hoidma ühtlast tempot**

Seda printsiipi toetab XP praktika „Jätkusuutlik tempo", mis sobib ettevõtte väärtuste ja arendusmeeskonna juhtimisega hästi. Juba mitu aastat on töid planeeritud realistliku arendaja poolt antud hinnangu alusel ning jäetud sisse ka puhvrid ootamatuste jaoks.

- **Sõnastamata printsiip: Sobivad protsessid ja vahendid**

Agiilse arenduse sellist printsiipi sõnastatud pole. Koch soovitab (2005) analüüsi käigus vaadelda metoodikaid ka sobivate protsesside ja vahendite võtmes. Selle printsiibiga seonduvad praktikad on:

FDD-s „Konfiguratsiooni juhtimine",

LSD-s „Võimendatud õppimine: sünkroniseerimine"; „Tarni nii ruttu kui võimalik: tõmbesüsteemid (i.k. *pull systems*), järjekordade teooria, viivitamise hind" ning „Näe tervikut: mõõtmised",

CC-s „Tehniline keskkond automaatsete testide, konfiguratsioonijuhtimise ja sageda integratsiooniga",

XP-s „Pidev integreerimine".

LSD-s on tugev fookus protsessidel ja vahenditel, tervelt viis LSD vahendit tegelevad nendega. „Sünkroniseerimine" sisaldab omakorda terve hulga alamvahendeid (koodi omandus, integreerimisprotsess, suitsutestid, liidese mudelid (termin i.k. *stubs and harnesses*), automatiseeritud testivahendid, automatiseeritud versiooni integreerimise vahendid, arenduse maatriksülesehitus), mis on Elioni kontekstis sobivad, eriti viimane. See näeb ette, et mitmete meeskondade puhul (nagu meie kontekstis) arendatakse (ja testitakse) kõigepealt liideste osa, et

sünkroniseerimisprobleemid saaksid lahendatud varakult. Kuna Elionis on tihti tegemist süsteemidevaheliste liidestega, kus arenduspartnerid on erinevad, siis tuleb ette olukordi, kus hilises faasis välja tulnud liidestamise probleemid toovad endaga kaasa suuri probleeme. Koodi omandus on käsitletud eespool, pidevat (automatiseeritud) integratsiooni käsitlen XP vastava praktika kontekstis. Suitsutestimine on ettevõttes töötav praktika, liideste mudeldamist (*mock-up* teenused) ning automaatteste oleme juurutamas antud analüüsist sõltumatult.

LSD „Tõmbesüsteemide" praktika sisu on, et tellija vajadus määrab ära versiooni sisu ja valmiduse – versioon on valmis, kui tellija jaoks vajalik osa on valmis ja töötab. Siin ei ole suurt erinevust teiste praktikate agiilsetest põhimõtetest (väikesed inkremendid, regulaarne näost näkku kommunikatsioon, „infokiirgurid" projekti seisu ja tööde järjekorra kajastajana). „Järjekorrateooriat" soovitab LSD kasutada arendusprotsessi kitsaskohtade leidmiseks. Alternatiiviks ja minu eelistuseks on tagasivaated (retrospektiivid) – protsessi osalised oskavad välja tuua lisaks objektiivselt mõõdetavatele probleemidele ka emotsionaalset poolt. „Viivituse hind" annab soovituse, kuidas projekti käigus tegemist vajavad otsused saab viia rahas mõõdetavaks ja seeläbi ühel tasandil võrreldavaks ja otsuse tegemist kergendavaks. „Mõõtmiste" praktika pakub Elioni kontekstis väärtust läbi selle, et kirjeldab mõõtmise ohtusid („saad, mida mõõdad") ja soovitab mõõta võimalikult kõrgel tasemel, mitte väga detailseid parameetreid. Näiteks isiku edenemise taseme asemel mõõta meeskonna edenemist.

FDD toob eraldi praktikana välja konfiguratsioonijuhtimise, seda suhteliselt üldisel tasemel. Mõned soovitused, nagu kontrollida ka dokumentatsiooni lisaks koodile, on siiski kasulikud.

CC praktika osad annavad kokku põhimõtteliselt sama sisu, mis XP „Pidev integreerimine" (pidev ahel test-arendus-integratsioon). Hindan neid praktikaid Elioni kontekstis vajalikeks ja võrdselt kõrgelt.

XP „Pideva integreerimise" praktika väärtus seisneb selles, et annab tegijatele võimalikult kiire tagasiside nende töö kvaliteedile. Kui süsteem ei integreeru, siis tuleb veel tööd teha. Levinud praktika on hoida kogu integreerimine iteratsiooni lõppu ühe suure sammuna. Kui integratsioonijärgne test näitab vigu, siis on raske üles leida, milline muudatus täpselt probleeme põhjustab. Kui vigane koht üles leitakse, siis peab keegi minema tagasi nädalaid tagasi kirjutatud koodi juurde, meenutama, kuidas see on disainitud ja välja mõtlema, mida vea parandamiseks muutma peab. (Koch 2005, 140). Teisalt jõuab nii ka uus funktsionaalsus kiiresti

tellijale tutvumiseks ja saab tagasiside, kas tehtu vastab ärilisele vajadusele (ASD kliendi fookusgruppide praktika). Alan Koch soovitab juhul, kui versiooni integreerimine võtab alla 15 minuti, juurutada täielikult pidev integratsioon. Kui integratsioonile kulub tund kuni kaks, on soovitus integreerida igapäevaselt ja veel suurema ajakulu juures vähemalt kord nädalas (Koch 2005, 144). Tänu tehnika arengule on tänapäeval võimalik integreerida ka suuri süsteeme pidevalt, selleks kasutatakse eraldiseisvaid servereid. Kokkuvõtteks ei anna FDD konfiguratsioonijuhtimise praktika minu hinnangul Elionis täna kasutusel olevale piisavalt juurde, et selle juurutamist tõsisemalt kaaluda. LSD suure praktikate ja vahendite hulga seast on aga üht-teist valida, mis teistes praktikates puudu. Osa vahendeid kattub siiski teiste metoodikate praktikates kirjeldatutega. XP või CC praktika on igati juurutamist väärt.

3.2.2 Väärtustame enam töötavat tarkvara kui täielikku dokumentatsiooni

Selle postulaadiga on seotud alljärgnevad kaks paksus kirjas ja täpploendis välja toodud printsiipi.

- **Töötavat tarkvara antakse välja sagedasti**

Seda printsiipi toetavad praktikad on:

ASD-s „Adaptiivne elutsükkel“,

DSDM-s „Sage väljalase“ ja „Iteratiivne ning inkrementaalne arendus“,

XP-s „Väikesed väljalasked“,

FDD-s „Korrapärane versioonivahetuste plaan“ ja „Omaduste haaval arendamine“,

LSD-s „võimendatud õppimine: iteratsioonid“,

Scrumis „Sprint“;

Crystalis „Sage väljalase“,

AUP-s „Väljalaske paigaldus“.

Scrumi puhul on tiimil peale iteratsiooni (Scrumi termin *sprint*, kestusega 1 kuu) planeerimist täielik iseseisvus plaanitu elluviimisel. Tiimil on volitused teha mida iganes, et saavutada sprindi eesmärk kokkulepitud ajaga. Kui tiim jõuab veendumusele, et eesmärgi kokkulepitud ajaga saavutada pole võimalik, on tiimil

õigus sprint katkestada. Sellele järgneb automaatselt uus sprindi planeerimine (arvestades saadud kogemusi) (Koch 2005, 142).

AUP esitab nõude planeerida versioone, kuid ei määra sagedust. XP ja FDD määravad tegema nii väikeseid versioone kui vähegi võimalik - nii näeb tellija võimalikult vara tulemust. XP puhul on soovitav iteratsiooni pikkus 1-4 nädalat, mis sobib ettevõttes soovitud igakuise versioonivahetuste tsükliga. Ülejäänud praktikate soovitus on üldisem vastavalt agiilse arenduse põhimõttele – planeeri iteratsiooni pikkus selliseks, et saaksid juurde luua uusi omadusi ning lase tarkvara versioone välja nii tihedalt, kui võimalik ja mõistlik. Elioni kontekstis oleks mõistlik kuupikkune versioonivahetuste vahe, kuna sama rütmiga toimuvad e-keskkondadega liidestatud tuumrakenduste versioonivahetused.

- **Töötav tarkvara on edenemise peamine mõõdik**

Seda printsiipi toetavad praktikad on:

ASD-s „Kliendi fookusgruppide ülevaatus" ja „Oleta – adaptiivne tsükli planeerimine",

DSDM-s „Sobivus ärilisteks vajadusteks",

XP-s „Pidev integreerimine" ja „Väikesed väljalasked",

FDD-s „Aruandlus/tulemuste nähtavus",

LSD-s „Võimendatud õppimine: tagasiside",

Scrumis „Sprindi ülevaatus" ja „Sprindi planeerimine",

AUP-s „Projektijuhtimise kord".

Ettevõtte kontekstis on projekti seisu jälgimiseks lisaks printsiibiga „töötav tarkvara on peamine edenemise mõõdik" nõustumisele vajalik ka järgmise iteratsiooni täpsem ning kogu projekti ulatuses ligikaudne planeerimine ja ootus metoodikale on, et see pakub selleks sobivaid praktikaid. Vajalik ei ole mitteagiilne planeerimine, kus väga täpselt on projekti plaan paigas, küll on kogu projekti ulatuses vajalik planeerida kulud ja töötajad, samuti väliste partnerite kasutamine (millises ajavahemikus kui palju inimesi). FDD lähenemine põhineb etappide lõppu tähistavatel verstapostidel (i.k. termin *milestone*) ja nende kaaludel. Kliendipoolne läbivaatus on üks verstapostidest. See on suhteliselt unikaalne lähenemine, mis võib Elioni kontekstis töötada. ASD kliendi fookusgruppide ülevaatuse idee on

intrigeeriv ja selle rakendamist tasub kaaluda. Planeerimise praktika (adaptiivne tsükli planeerimine) on ka ettevõttele sobiva täpsusega. XP pideva integreerimise põhimõte tagab, et programmeeritud muudatused saavad kiiresti integreeritud ja muutuvad seega huvitatud osapooltele vaadeldavaks ja testitavaks. See on Elioni kontekstis väga vajalik praktika ja vajab kindlasti juurutamist. Minimaalsete väljalasete praktika osas sobib Elionile kuupikkune tsükkel (vt eelmise printsiibi kommentaare). DSDMi põhimõte on, et peamine mõõdik projekti kulgemisele on töötava ja äriliselt sobiva funktsionaalsuse hulk. Projekti edukuse tagamiseks pakub DSDM väikeseid iteratsioone ja kasutajapoolset tihedat tagasisidet, lootes sellega projekti pidevalt õigel teel hoida. Planeerimise jaoks tahaksin vaja aga enamat. AUP pakub üsna sobivat projektijuhtimise protsessi, kuna on piisavalt põhjalik, samas on protsessid sobivad agiilse arenduse põhimõtetega. LSD sarnaselt DSDM-ile pakub praktikaks kasutajapoolset tagasisidet ja ei paku praktikat planeerimise toetuseks. Scrumi praktikad sprindi ülevaatus ja sprindi planeerimine annavad Elioni kontekstis piisava raamistiku, kuidas planeerida projekti, kuidas mõõta tulemusi ning kuidas projekti kulgu hinnata. Selle printsiibi praktikate kokkuvõtteks võib öelda, et parimaid praktikaid Elioni jaoks pakuvad AUP ja Scrum.

3.2.3 Väärtustame enam kliendi osalust arenduses kui lepingute koostamist

Agiilse arenduse printsiip, mille saab siduda selle manifesti punktiga, on

- **Lähedane ja igapäevane koostöö tellijate ja arendusmeeskonna vahel**

Agiilsete metoodikate praktikad, mis seonduvad selle printsiibiga, on:

ASD-s „Projekti osapooled kui iseseisvad agendid" ning „Adaptiivne (eestvedamis-koostöö) juhtimismudel,

DSDM-s „Aktiivne kasutaja kaasatus" ning „koostööle ja koostegutsemisele suunatud lähenemine",

XP-s „Kogu meeskond",

LSD-s „Versioneeri terviklikult: teadvustatud terviklikkus ja kontseptuaalne terviklikkus" ning „Näe terviklikult: Lepingud",

Scrumis „Toote tööde nimekiri (i.k. Scrumi termin *Product backlog*)" ja „Scrumi koosolek",

AUP-s „Testimise protsess",

CC-s „Lihtne juurdepääs eksperttasemel kasutajatele".

CC praktika on Elioni kontekstis väga sobiv – tuleb tagada, et vajalikud tellija esindajad oleks kergelt kättesaadavad. Just see on ka ettevõtte varasemate projektide juures välja tulnud probleem, et projekti liikmete ja tellijate vahelises (asünkroonses, näiteks e-mail) suhtluses jääb vajaka vastuste kiirusest. Ka Scrumi praktikad sobivad hästi, sest näevad ette igapäevaseid koosolekuid ja tihedat suhtlust osapoolte vahel. XP-s on aja jooksul muudetud praktika „Kohapealne klient" ümber „Kogu meeskonnaks", kus kliendi esindaja kogu aeg kohapeal viibimise nõudest on loobutud ja asendatud see sarnaselt CC-le nõudega, et kliendi pädev esindaja peab olema kogu aeg kättesaadav. Nagu CC puhulgi kommenteerin, on see Elioni kontekstis väga sobiv praktika. AUP protsessides on tellija esindaja määratud aktsepteerimistesti tegema, samuti viidatakse tema aktiivse (projekti töös) osaluse kriitilisusele projekti õnnestumises. LSD terviklikkuse praktika viitab tellija rollile terviklikkuse mõistmisel, see jääb ettevõtte vajaduste jaoks teistest veidi kaugemaks. Lepingute praktika on Elioni kontekstis mõnevõrra vähem oluline, sest projekti osas arendusmeeskonna ja tellija vahel lepinguid ei sõlmita – tegemist on firmasisese projektiga. Välise arendusfirmaga sõlmitakse raamleping ning konkreetse projektide kohta lepingu juurde kuuluvad lisad, mille sisu toetab agiilse arenduse printsiipe, siin saab rakendada LSD vastavat praktikat. ASD praktikaid on varem analüüsitud eelnevate printsiipide juures (Projektide tuumaks on motiveeritud isikud ...). Tellijaga koostöö kui võrdne võrdsega on juba juurdunud praktika. DSDM praktikad on ettevõttele sobivad (tihe koostöö tellijaga on ülioluline). Koostöö printsiibiga seotud metoodikate praktikate analüüsi kokkuvõtteks võib öelda, et üsna võrdselt sobivad nii XP, Scrumi, DSDM kui ASD pakutavad praktikad. Lisaks on huvitav LSD lepingute praktika.

3.2.4 Väärtustame enam muutustele reageerimist kui plaani järgimist

Selle manifesti punkti alla tooksin alloleva printsiibi:

- **Tere tulemast nõudeid muutma, ka arenduse hilises faasis!**

Selle printsiibiga seonduvad järgmised praktikad:

ASD-s „Adaptiivne elutsükkel“,

DSDM-s „Kõik muudatused on tagasivõetavad“ ja „Nõuded on suurtes piirides paigas“,

XP-s „Süsteemi metafoor“ ja „Disaini täiustamine“,

FDD-s „Domeeni objekti mudel“,

LSD-s „Otsusta nii hilja kui võimalik: variantide läbimõtlemine, viimane otsustamise moment, otsuste tegemine“ ning „Versioneeri terviklikult: refaktoorimine“,

Scrumis „Sprindi planeerimise koosolek“,

AUP-s „Implementeerimise kord“.

DSDM praktika on agiilse arenduse jaoks ootamatu – nõuded on suurtes piirides paigas enne arenduse alustamist. See ei tähenda, et detailides ei võiks muudatusi teha. Siiski – Elioni vajadus oleks võimalus pidevalt (kas läbi õppimise või väliste tegurite mõjul) nõudeid muuta. Seega see praktika pigem ei sobi. Muudatuste tagasivõtmise praktika on aga sobiv. ASD praktika tuginebki läbi õppimise, uute teadmiste, nõuetesse muudatuste sisseviimisele ja sellele reageerimisele, ettevõtte vaatest pole aga õppimine ainus muudatuste allikas. XP praktikatest „Süsteemi metafoor“ on suunatud pigem töökorraldusele ja „Lihtne disain“ tarkvara arendamise spetsiifikale. Siia alla kuulub ka koodi pidev refaktoorimine. Mõlemad praktikad sobivad Elioni kontekstis kasutamiseks väga hästi. Koodi refaktoorimine ja mitmed teised lihtsa disaini põhimõtted on sees ettevõttes kasutatavas programmeerija kompetentsimudelis kui hea praktika kasutamiseks. LSD praktika muutustega toimetulekuks on teha otsuseid viimasel võimalikul hetkel, siis on suurem võimalus, et ei pea muutunud nõuete tõttu palju ümber tegema. LSD teine praktika on sarnaselt XP-le koodi pidev refaktoorimine. Scrum lubab nõudeid muuta kuidas tahes, aga seda järgmises iteratsioonis – töösolev iteratsioon pannakse peale planeerimist lukku ja sinna ei tohi ka pisemaid muudatusi teha. See on Elioni vaates Scrumi kõige nõrgem koht, kuna olulisi ja väikeseid muudatusi tuleb sisse veel versioonivahetuse nädalal ja sellest praktikast ei tahaks loobuda. AUP ei too välja konkreetset praktikat muudatustega toimetulekuks, küll pakub tegutsemiskorra. Ka AUP soovitab koodi refaktoorida.

3.2.5 Ülejäänud printsiibid

Kolm alltoodud täpploendis olevat printsiipi pole otseselt seotud manifesti väärtustega, vaid on tehnilist laadi.

- **Pidev tähelepanu tehnilisel täiuslikkusel ja heal disainil**

Selle printsiibiga seonduvad praktikad on:

ASD-s „Õpi: Kvaliteedi revideerimine: Tarkvara ülevaatus“,

DSDM-s „Testimine läbi kogu elutsükli“,

XP-s „Testimisel põhinev arendus“, „Paarisprogrammeerimine“ ja „Koodistandardid“,

FDD-s „Ülevaatused“,

LSD-s „Võimendatud õppimine: (lahendusvariantide) komplekti põhine arendus“, „Anna meeskonnale volitused: eestvedamine ja kompetentsus“ ning „Versioneeri terviklikult: testimine“,

AUP-s „Implementeerimise kord“,

Scrumis „*Scrum master*“.

Kaks põhilist praktikate gruppi on koodi ülevaatus ja testimine. XP lähenemine koodiülevaatusele on tugevalt seotud praktikatega „Paarisprogrammeerimine“ (paariline vaatab juba jooksvalt üle), „Koodi ühisomandus“ (kes iganes meeskonnast võib sattuda sellele koodiosale) ning „Koodistandardid“ (meeskonnas on kokku lepitud ühine arusaam standarditest). Selline terviklik lähenemine sobib ettevõtte konteksti väga hästi. Traditsiooniliste metoodikate puhul vastutavad programmeerijad koodi kirjutamise eest ja testijad testimise läbiviimise ning vigade leidmise eest. Nii kujuneb välja olukord, et programmeerija justkui ei vastuta koodi kvaliteedi eest. XP „testimisel põhineva arenduse“ praktika muudab seda mõtteviisi, asetades vastutuse koodi kvaliteedi eest selgelt programmeerijapaarile. XP põhimõte on, et testid kirjutatakse valmis enne tegelikku programmeerimist sellesama paari poolt, kes hiljem ka koodi kirjutab. Seega peavad programmeerijad mõtlema enne sellele, kuidas kood katki läheb ja alles hiljem, kuidas kood tööle panna. Hiljem, programmeerimisfaasis, mõtleb klaviatuurita paariline muuhulgas ka sellele, kas kirjutatud testid katavad seda funktsionaalsust, mida parajasti

kirjutatakse. Kui mitte, siis peatutakse ja kirjutatakse valmis ka seda katvad testid. Samal ajal XP ei välista programmeerijapaarist sõltumatut testimist (Koch 2005, 198). Elioni kontekstis on seda praktikat hea kasutada ning samuti programmeerijate kõrval ka professionaalset testijat, kes testib kogu tervikahelat.

DSDM praktika läheneb testimisele hoopis laiemalt kui arenduse testimine – alates plaanile vastavuse kontrollist. See praktika pakub tuge kõrgekvaliteedilise tulemuse saavutamiseks, kuid nõuab oma põhjalikkuse tõttu protsesside kohendamisel ja juurutamisel rohkem tööd. ASD vaade koodi ülevaatusele tehnilise kvaliteedi võtmes on üsnagi range – iga programmeerija töö vaatab üle teine programmeerija. Hindan seda siiski mõistlikuks, sest mitmed allikad (Beck 1999, Cockburn 2000) toovad välja saadava kasu vigade varases staadiumis leidmise ja elimineerimise näol, samuti toimub teadmiste ülekandumine. FDD koodiülevaatus praktika sarnaneb ASD-le ja hinnang Elioni kontekstis on sama. LSD-s on mitmeid praktikaid, mis seda printsiipi toetavad. Lahendusvariantide kompleksne analüüs ja selle baseeruv arendus on teiste agiilsete metoodikatega võrreldes unikaalne ning keskendub erinevate lahendusvariantide väljaselgitamisele ja nende analüüsimisele, jättes otsuse tegemise hilisemaks. Kuna otsuste tegemine on lükatud võimalikult hilisesse faasi, siis Elioni organisatsioonikultuuri ja väärtuste („Alusta kohe!") vaates see praktika pigem ei sobi (see muidugi ei tähenda mõtlematu kiirustamise väärtustamist). Eestvedamise praktika on jällegi organisatsioonikultuuri ja väärtustega („Läheb korda!") täielikus kooskõlas. Ka kompetentsuse praktika (tuvasta vajalikud kompetentsi ning taga, et meeskonnas oleks need olemas) on mõistlik ja vajalik. Testimise praktika sarnane DSDM-iga, ülejäänutest laiema ulatusega. AUP implementeerimise kord sisaldab vähe tehnilise täiuslikkuse saavutamise tööriistu. Scrumi *Scrum master*'i roll on jälgida, et meeskond jälgiks Scrumi praktikaid ja reegleid ning selle kaudu oleks tagatud tehniline kvaliteet ja hea disain. Scrum ei paku mingeid tehnilisi praktikaid selle jaoks, küll on Scrumi puhul paigas meetodid, kuidas kompetentside ja motiveerituse kaudu tehniline kvaliteet saavutada. Võttes kokku tehnilise täiuslikkuse ja hea disaini printsiibiga seotud metoodikate analüüsi, leian, et XP pakutavad praktikad on Elioni jaoks väärtuslikud, samuti ASD ja FDD pakutavad koodiülevaatused. Scrum on veidi nõrgem just tehniliste praktikate puudumise tõttu.

- **Lihtsus**

Kuigi lihtsus on agiilsete meetodite peamine teema (Koch 2005, 177), on selle printsiibiga seotud praktikaid vaid neli:

XP-s „Lihtne disain“,

LSD-s „Eemalda kasutu: kasutu nägemine“ ja „Eemalda kasutu: väärtusevoo kaardistus“,

AUP-s „Lihtsuse filosoofia“.

XP põhimõte on, et koodi ei lisata midagi, mida pole töösoleva loo jaoks vaja. Seda mõtteviisi on Elioni arendusmeeskondades kindlasti üsna raske juurutada, sest senine põhimõte on olnud just lahenduste üldistamine – näiteks mingi teenuse projekteerimisel püütakse ette näha ka tulevased võimalikud vajadused ja teenused teha piisavalt universaalsed, et neid tulevasi vajadusi juba ette katta. Koos pideva refaktoorimise praktikaga on seda lihtsam omaks võta, sest koodi töötatakse nagunii pidevalt ümber. LSD pakub hulga tööriistu kasutu eemaldamiseks. „Osaliselt tehtud töö“ on sarnane XP „Lihtsa disainiga“ ja annan ettevõtte kontekstis sama hinnangu. „Lisaprotsessid“ paneb hindama, kas kõik protsessid annavad väärtust. Kuigi ettevõte on üsnagi protsessipõhise tegutsemisega, sobib siinses kontekstis siiski põhimõte, et kui protsess ei toeta arendust ega ole ka oluline kvaliteedi tagamiseks, siis tasub kaaluda selle kasutamise lõpetamist. „Lisaomaduste“ tööriista tuleks ettevõtte kontekstis mõista nii, et arendajal ei tasuks **realiseerida** omadusi, mida tellija pole soovinud. Küll oleks mõistlik selliste omaduste kohta arendajal **teha ettepanekuid**, et tellija saaks otsustada, kas ja millise prioriteediga vastavat lisaomadust realiseerida. „Ülesannete vahel ümberlülitumise“ tööriist, nagu ka „Ootamine“, annavad tööprotsessile juurde efektiivsust ja on mõlemad ettevõtte kontekstis kasulikud. Mõlemad probleemid, millele need tööriistad lahenduse pakuvad, olid varasemalt tuvastatud kui lahendamist vajavad. „Liikumine“ on vahend üle vaatamaks konkreetse inimese tegevused pilguga, kas sealt saab midagi elimineerida/automatiseerida (sarnaselt „Lisaprotsesside“ üldiste protsesside ülevaatusega) ning „Vead“ viitab vigadele kui suurimale ajaraiskajale arendusprotsessis. Nagu ka XP, ütleb LSD, et vead tuleb avastada võimalikult varajases faasis, siis on kulu nende parandamisel kõige väiksem. XP praktikad toetavad selle saavutamist minu hinnangul põhjalikumalt, kui LSD üldisemad praktikad.

AUP lihtsuse filosoofia ongi pigem mitte praktika, vaid üldisem soovitus hoida dokumentatsioon lihtne, mitte tuhandetes lehekülgedes. XP praktika on teha dokumentatsiooni siis, kui on teada kelle jaoks vaja ja mida dokumentatsioon peab sisaldama. Kuna dokumentatsiooni kasutajatel on sellele erinevad ootused, siis tuleb dokumentatsiooni sisu hoida ootustega vastavuses ja mitte lisada üleliigset.

Kokkuvõttes on XP praktikad Elionile sobilikud, kuid ka LSD mitmed praktikad toetavad ettevõtte arenduses olevate probleemide lahendamist.

- **Regulaarsed tagasivaated ja korrektsioonid**

Selle printsiibiga seotud praktikad on:

ASD „Õpi: kvaliteedi ülevaatus: *postmortem*“,

CC „Tagasivaatav täiustumine“,

Scrumi „sprindi tagasivaade“.

Need praktikad on sisuliselt üsnagi sarnased, mistõttu ei too ma üht teise ees esile. Kerge eelis on Scrumil, mille osas on Elionis toimunud koolitusi. Tagasivaate praktika kui selline on ettevõttes väärt juurutamist, kuna aitab kogemustest õppida ja efektiivsemaks saada.

3.3 Analüüsi kokkuvõte ja järeldused

Võtan allpool agiilse arenduse manifesti printsiipide kaupa kokku erinevate analüüsitud metoodikate eelised ja puudused ning peatüki lõpus kannan tulemused ülevaatlikku tabelisse. Metoodikate koguhinnangu põhjal valin ettevõttes juurutamiseks välja rakendatava metoodika või sünteesin selle eriveate metoodikate praktikatest.

- **Projektide tuumaks on motiveeritud inimesed, keda tuleb usaldada**

XP saab eelise ja FDD miinuse koodi omanduse praktika osas. Projekti meeskonna õiguste osas pole ükski eelistatud.

- **Iseorganiseeruvad meeskonnad**

Jällegi ei saa ühelegi praktikale eeliseid anda, FDD miinus tuleneb klassi omanduse praktikast, mida Elionis rakendada ei taheta.

- **Parim suhtlusvorm on näost näkku vestlus (koospaiknemine)**

Scrumi koosolekute praktika on sobivaim, CC oma väga raske juurutada.

- **Jätkusuutlik arendus, mis on võimeline hoidma ühtlast tempot**

XP praktika sobib hästi.

- **Sõnastamata printsiip: Sobivad protsessid ja vahendid**

FDD pakub terve hulga vahendeid, mille seast sobivaid komplekteerida, osa on küll kaetud ka teiste praktikate poolt. Kindlasti pakuvad huvi „viivituste hind" ja" mõõtmiste" praktika. XP ja CC pakuvad sarnaseid praktikaid, mida samuti kõrgelt hindan.

- **Töötavat tarkvara antakse välja sagedasti**

Kuigi erinevate metoodikate praktikad on suhteliselt sarnased, saavad Scrum ja XP eelise sobiva iteratsiooni pikkuse põhjal.

Töötav tarkvara on edenemise peamine mõõdik

Seda printsiipi toetavatel praktikatel on palju väärtusi, valisin välja AUP projektijuhtimise korra ja Scrumi sprintide praktikad, mis samuti annavad hea raamistiku projekti korraldamiseks.

Lähedane ja igapäevane koostöö tellijate ja arendusmeeskonna vahel

Üsna võrdselt sobivad nii XP, Scrumi, DSDM kui ASD pakutavad praktikad. Lisaks on huvitav LSD lepingute praktika.

Tere tulemast nõudeid muutma, ka arenduse hilises faasis!

XP praktikad sobivad kõige paremini, DSDM miinuseks nõuete laias piiris paigasoleku soov ja Scrumi miinuseks käimasoleva iteratsiooni (sprindi) skoobi täielik lukusolek.

Pidev tähelepanu tehnilisel täiuslikkusel ja heal disainil

XP praktikad sobivad ettevõttele väga hästi, väärtuslikud on ka ASD ja FDD koodi ülevaatuse praktikad.

Lihtsus

Jällegi on XP praktikad ettevõttele sobilikud, kuid ka LSD mitmed praktikad toetavad just Elioni probleemide lahendamist.

Regulaarsed tagasivaated ja korrektsioonid

SCRUM, CC, ASD pakuvad siin ühtemoodi sobivaid praktikaid, kerge eelis on Scrumil, mille osas on ettevõttes toimunud koolitusi.

Järgnevasse tabelisse olen kokku võtnud eelnevalt analüüsitud erinevate metoodikate eelised ja puudused, märkides olulise eelise plussmärgiga ja olulise puuduse miinusmärgiga. Tabeli lõpus liidan plussid ja miinused ning saan tulemuse, mis näitab vastava metoodika ettevõttele sobivuse hinnangut. Eeliseid ja puudusi toon välja agiilse arenduse manifestis kirjeldatud põhimõtete kaupa, põhimõtete väärtus on enam-vähem ühesugune, mis tähendab, et erinevate ridade plussid või miinused on sarnase kaaluga. Erinevate metoodikate tulemused on võrreldavad samal skaalal.

	ASD	DSDM	XP	FDD	LSD	Scrum	AUP	CC
Motiveeritud inimesed			+	-				
Iseorganiseeruvad meeskonnad				-				
Näost näkku suhtlus						+		-
Jätkusuutlik arendus			+					
Protsessid ja vahendid			+	+				+
Sagedased väljalasked						+		
Töötav tarkvara			+			+	+	
Koostöö tellijaga	+	+	+			+		
Nõuete muutmine		-	+			-		
Tehniline täiuslikkus	+		+	+				
Lihtsus			+		+			
Tagasivaated	+					+		+
Summa	**+3**	**0**	**+8**	**0**	**+1**	**+4**	**+1**	**+1**

Tabel 2. Analüüsi kokkuvõte

Võttes analüüsi tulemused kokku tabelis 2, näeme XP-l suuri eelisi, neljapunktise vahega järgneb Scrum ja omakorda punktiga vähem ASD.

Lisaks tehtud analüüsi kokku võtvale tabelile 2 kasutasin ka Alan Kochi/ASK Process Inc välja töötatud agiilsete meetodite hindamise töövihikut, mida veidi modifitseerisin ja täiendasin metoodikate osas. Töövihiku tulemus (vt tabel 3 ja lisa 1) erineb teatud määral käesoleva peatüki analüüsi tulemusest, kuna töövihik hindab skaalal „lihtne-keeruline juurutada", analüüs on tehtud aga „väheoluline-oluline ettevõttele" skaalal.

	ASD	DSDM	XP	FDD	LSD	Scrum	AUP	CC
Organisatsiooni-kultuur	4,4	4,0	4,2	3,1	3,7	4,0	3,7	4,0
Tellijad	3,8	3,6	3,9	3,0	3,6	3,6	3,7	3,3
Projektid	3,7	3,6	4,0	2,9	3,7	3,8	3,4	3,7
Vahendid ja protsessid	4,0	3,8	4,1	2,6	3,4	4,0	3,7	4,0
Meeskond	3,9	3,5	3,8	2,5	3,3	3,6	3,3	3,6
Sobivus	**4,0**	**3,7**	**4,0**	**2,8**	**3,6**	**3,8**	**3,6**	**3,7**

Tabel 3. Agiilse arenduse töövihiku meetodite hinnangu kokkuvõte

Analüüs näitab, et Elionile sobivuse poolest on teistest praktikatest tunduvalt sobivam XP, hästi sobivad ka Scrum ja ASD. Juurutamise lihtsuse analüüs näitab, et juurutada on lihtsam XP-d ja ASD-d, väikese vahega järgneb Scrum. Kuna XP praktikad on rohkem tehnilist laadi, siis on mõistlik valida kõrvale praktikaid, mis keskenduvad mittetehnilistele aspektidele ja selleks sobib hästi Scrum oma „Igapäevase koosoleku", „Sprindi ülevaatuse" ning „Sprindi tagasivaate" praktikatega, valin need juurde. Täiendavalt pakun kombineerida XP praktikad „Kogu meeskond", „Planeerimismäng", „Süsteemi metafoor" vastavate Scrumi praktikatega „Scrumi meeskond" (just tootejuhi roll sellest praktikast), „Sprindi koosolek", „Toote tööde nimekiri". Lisaks valin sünteesitud metoodikasse LSD-st „Lepingute", „Viivituse hinna" ning „Mõõtmiste" praktikad kui unikaalsed ja Scrumi-XP kooslust hästi täiendavad.

Läbiviidud analüüsile tuginedes teen ettepaneku juurutada sünteesitud metoodika, mis sisaldab XP metoodikat täismahus ning lisaks XP praktikaid täiendavaid praktikaid Scrumist ning LSD-st.

4. Juurutamine

Käesolevas peatükis analüüsin erinevate juhtumianalüüside alusel kriitilisi edutegureid, probleeme ja ohte ning kirjeldan eelnevat aluseks võttes, kuidas planeerin valitud metoodikate ja praktikate elluviimise. Ohud kirjeldan riskianalüüsis (lisa 2) ning võtan kokku tabelis 4. Juurutamisel mängivad olulist rolli meeskondadevahelise koostöö koordineerimine, meeskonnasiseste rollide jaotumine ja iseorganiseerumine, töökorraldus ning kommunikatsioon ja tulemuste mõõtmine, käsitlen neid osasid käesolevas peatükis allpool.

E-kanalite uuendamise arendusprojektid on algusjärgus ja sellevõrra on uue metoodika juurutamine kergem. Christopher O'Connell viitab juhtumianalüüsile, kus agiilsele metoodikale mindi üle projekti sellises staadiumis, kus suur osa koodist oli valmis ja tegeldi järjekordse versiooni väljalaskmiseks leitud vigade parandamisega („tulekahjude kustutamisega"). Sellises situatsioonis ei pidanud meeskond uue metoodika juurutamist abivahendiks edaspidi sarnaste probleemide vältimiseks, vaid veel üheks takistuseks töö valmimisel.

Juurutuse teeb kergemaks ka see, et juhtkonnas on olemas selge poolehoid agiilsele lähenemisele ning ka IT arenduse juht on selle tugev toetaja, „sponsor". „Sponsori" vajadusele, kes on kõike väljapanevalt valmis kaitsma agiilse arenduse põhimõtteid, viitavad oma Primavera Inc juhtumianalüüsis ka Bob Schatz ja Ibrahim Abdelshafi (Schatz, Abdelshafi 2005). Samuti tuleb agiilsete metoodikate juurutamisel kasuks meie valitud arenduspartneri enam kui viie aastane pikaajaline kogemus XP kasutamisel.

4.1 Projekti kriitilised edutegurid

Subhas Chandra Misra, Vinod Kumar ja Uma Kumar viisid läbi uurimuse 241 vastanu seas ja leidsid 9 kriitilist edutegurit: kliendi rahulolu, kliendiga koostöö, kliendi pühendumus projektile, otsuste tegemiseks kuluv aeg, organisatsioonikultuur, kontroll, isikuomadused, sotsiaalne kultuur ning koolitus ja õppimine (Misra jt 2009). Kliendi (tellija) rahulolu taseme püüame kõrge hoida läbi pideva koostöö projekti käigus. Samuti on oluline asjaolu, et Scrumi ja XP kooslus annab teostatavate tööde nimekirja koostamise ja haldamise tellija kätte, misläbi on tellija mõjuulatus suur ja rahulolu usutavasti kõrgem. Meeskond osutab talle abi

otsuste tegemise kergendamiseks lugude hindamisega (plaanimismäng). Kliendi pühendumist saavutada aitab ühest küljest projektide eelnev prioritiseerimine, et teostatavad projektid oleksid ka tellija jaoks olulised ja ta tunneks nendega seotust. Teisest küljest on oluline tellija ajaplaneerimine, et sarnaselt arendusüksuses inimressursi planeerimisele ja broneerimisele oleks ka tellijal projekti jaoks aeg broneeritud. Otsuste tegemiseks saab agiilse metoodika puhul kuluma vähem aega, sest meeskonnal on volitused otsuseid ise teha. Organisatsioonikultuuri analüüsisin peatükis 2.4 ja leidsin selle Elionis agiilsete metoodikate jaoks sobiva olevat. Kontrolli projekti kulgemise üle käsitlen peatükis 4.6. Agiilsetele metoodikatele sobivate ja vajalike isikuomadustega saab arvestada värbamise käigus, koolitusvajadustega tuleb arvestada juurutamise planeerimisel (vt riskianalüüsi lisas 2). Õppimine toimub lisaks koolitustele ka tööprotsessi käigus, abiks on regulaarsed (igakuised) tagasivaatekoosolekud (retrospektiivid).

Tsun Chow ja Dac-Buu Cao viisid läbi uuringu 109 agiilse projekti seas ja leidsid 6 projekti kriitilist edutegurit: Meeskonda ümbritsev keskkond, meeskonna võimekus, kliendi kaasatus, juhtimisprotsessid, agiilsed tarkvaraarendustehnikad ja tarnestrateegia (Chow, Cao 2007). Meie projekti kontekstis pean eriti olulisteks tarnestrateegia takistamatut toimimist (funktsionaalse tarkvara esimeste osade kiire valmimine ja pidev juurdevalmimine), tehnilistest teguritest dokumentatsiooni hulga balansi saavutamist (tegelikeks vajadusteks piisavalt, kuid mitte enam) ning integratsioonijärgsete testide toimimist, meeskonna võimekuse valdkonnast meeskonnaliikmete kõrget kompetentsi ja motiveeritust, juhtimisprotsesside valdkonnast head planeerimise (sh hindamise) ja progressi jälgimise vahendit (i.k. *planning and tracking*) ja igapäevaseid infokoosolekuid, toimivat iseorganiseeruvat meeskonda ning tellija kaasatust ja pühendumist.

Mitmed allikad on viidanud juhtumianalüüside kokkuvõtetes, et õnnestumiseks oleks pidanud kaasama konsultandi varasemas faasis, mitte ületamatute raskuste tekkides. Samuti on oluline koolitus ja tugi (*coaching*) ülemineku algusfaasis. Võtame seda arvesse.

Scott H. Ambleri uuring näitas, et 62% organisatsioonidest rakendas edukalt agiilset lähenemist projektides, mis olid integreeritud pärandsüsteemidega. 54% organisatsioonidest pidid üle saama koskmudeli kultuuri mõjudest, 52% pidi tõsiselt tegelema tellija pühendumise saavutamisega, 31% organisatsioonidest oli IT meeskond (agiilseks lähenemiseks) liiga spetsialiseerunud. (Ambler 2009).

Pärandsüsteemidega integreerimise edukuse protsent pakub meie ettevõtte kontekstis huvi, kuna e-kanalid saavad suure osa infost teistest süsteemidest (kasutades SOA/EDA põhimõtteid). Tellija pühendumise saavutamisest oli juttu eespool. Arendajate vahel Elionis liigset spetsialiseerumist pole ning arendaja, testija ja analüütiku rolle planeerime kasutada ka edaspidi.

Agiilsetes meetodites puudub mittefunktsionaalsete nõuete käsitlemise tugi (Paetsch jt 2003, Sillitti jt 2005). Küll on võimalik neid nõudeid ise lisada tööde nimekirja, arvestame sellega.

Rasmus Mencke toob Salesforce'i juhtumianalüüsis välja 5 võtmetegurit:

- tootejuhi roll on õnnestumise põhiliseks teguriks, neid tuleb koolitada varakult ja põhjalikult;
- väliseid konsultante tuleks kaasata varakult;
- tuleb õppida, kuidas senise põhjaliku dokumentatsiooni asemel kirjutada lihtsaid lugusid (*story*);
- kadude vältimiseks peavad lood olema kirjeldatud heal tasemel, st sisaldama ka aktsepteerimiskriteeriume ning olema kasutatavuse seisukohast testitud;
- oluline on juhtide kaasamine neile ülesannete andmise näol.

Võtan neid tegureid arvesse riskianalüüsis (lisa 2).

Bob Schatz ja Ibrahim Abdelshafi (2005) leiavad, et ülemineku läbiviimine vajab sellega põhjalikult tegelemist, puudub n.ö. „hõbekuul" ning toovad Primavera Inc juhtumianalüüsi põhjal välja seitse tegurit, millele tähelepanu pöörata:

- välise konsultandi kasutamine, kes oskaks objektiivselt ja ausalt tagasisidet anda;
- meeskonnatunde tekitamine ja meeskonnatööle suunatus, fookus ülesannetelt meeskonnale;
- agiilsel arendusel ka agiilne keel, vanade terminite uues kontekstis kasutamine toob kaasa ka vana mentaliteedi jätkumise;
- juhtkonna toetus ja arusaam, et agiilse lähenemise käigus liigub initsiatiiv pigem alt üles ehk meeskonnalt juhtkonnale, mitte vastupidi;

- ületundide tegemise vajaduse puudumine ehk jätkusuutlik arendustempo;
- meeskonnaliikmete läbirääkimis- ja suhtlusoskused, sest võrreldes monumentaalmetoodikatega on suhtlusel hoopis olulisem roll;
- probleemide varajane äratundmine ja nendega tegelemine.

Konsultandi kaasamist on käsitletud juba eespool, meeskonnale suunatust võtan arvesse peatükis 4.1 ning vanade terminite kasutamisega seotud ohtu peatükis 4.5. Juhtkonna toetust ja meeskonnaliikmete sotsiaalseid oskusi on arvesse võetud riskianalüüsis (lisa 2). Jätkusuutlik arendustempo on valitud XP metoodika osa, seega kuulub kindlasti juurutamisele. Meie e-kanalite projektide puhul on võimalik saada klientidelt pidevat tagasisidet lisaks pöördumistele ka kasutustatistika põhjal. Nii on võimalik lihtsalt tuvastada, kas uued võimalused saavad loodetud kasutajatehulga ja oskame probleeme kohe näha ning nendega tegelema asuda.

Eelpooltoodut arvesse võttes pakun välja Elioni kontekstis kriitilised edutegurid:

- tootejuhi rolli käivitumine, sh oskuse tekkimine lugusid õigesti kirjutada ja vastutuse võtmine toote omaduste nimekirja haldamise eest;
- tellijaga koostöö ja tema piisav pühendumus;
- meeskondlikku tegutsemist ja koostööd soodustava keskkonna loomine;
- meeskonnaliikmete tehniliste ning läbirääkimis- ja suhtlusoskuste olemasolu;
- iseorganiseeruva meeskonna käivitumine;
- juhtimisprotsesside muutmine agiilsele arendusele vastavaks;
- tarnestrateegia „vara ja tihti" käivitumine;
- pärandsüsteemidega liidestamise edukus;
- probleemide varajane äratundmine ja nendega tegelemine.

4.2 Võimalikud ohud ja riskianalüüs

Töötasin käesoleva magistritöö tegemisel läbi mitmeid juhtumianalüüse ja kogusin teiste kogemuste põhjal enamlevinud ohud kokku töö lisas 2 olevasse riskianalüüsi. Analüüsis on kirjeldatud risk, tagajärg realiseerumisel, vältimise võimalus ning tegevusplaan riski realiseerumisel. Riski tähtsust hindan riski tekkimise tõenäosuse (skaalal 1-3, kus 1 on väike tõenäosus ja 3 suur tõenäosus) ning riski mõju (skaalal 1-3, kus 1 on väike mõju ja 3 suur mõju) korrutisena. Mida suurem on tulemuse arvväärtus, seda olulisem risk. Allikates viidatud ohud, mille puhul Elioni kontekstis on kas mõju või tõenäosus nullilähedane, on jäetud tabelist välja. Kokkuvõte olulisematest riskidest on peatüki lõpus tabelis 4. Allpool on toodud koos viidetega valik ohtudest, mida riskianalüüsis on arvesse võetud.

Ettevõtte Progressive Insurance (Jochems, Rodgers 2007) kogemused agiilsele metoodikale üleminekul olid järgmised:

- kui meeskond ei ole muudatuse poolt, siis tuleks uusi praktikaid tuua sisse vähehaaval;
- tuleb panustada piisavalt kogu meeskonnale agiilse arenduse eeliste selgitamisse ning nende eelistuste agiilse lähenemise kasuks kallutamisse,
- tuleb pakkuda võimalust agiilse testimise vahendite ja meetodite koolituseks;
- projekti meeskonnale tuleb luua keskkond, mis toetaks arenemist ja nendepoolset soovi täiustada arendusprotsessi,
- konsultandid tuleb valida lähtuvalt meeskonna vajadustest;
- info tuleb hoida liikumas ja püüda probleemid kiirelt tuvastada.

Bob Schatz ja Ibrahim Abdelshafi (2005) toovad välja Primavera Inc agiilsele arendusele ülemineku juhtumianalüüsi käigus leitud takistused:

- arendajad innustusid liigselt ja avaldasid iteratsiooni ülevaatuse käigus ka lugusid, mis polnud kas täielikult testitud või sisaldasid olulisi vigu. Selliste lugude kuhjumine viis selleni, et terve sprint kulus ainuüksi kuhjunud vigade parandamiseks;
- arendajad kasutasid kiire tulemuse saavutamiseks meetodeid, mis polnud õiged jätkusuutlikkuse tagamiseks;

- huvipooled pidasid kasutuseloleva mõõdiku ehk kõigi tööde ja valminud tööde suhet kajastava graafiku (i.k. *burndown chart*) pakutavat infot väheseks ja vajasid rohkem infot mingi konkreetse omaduse valmimise tähtaja või versiooni valmimise tähtaja osas;

- arendajad võtsid iteratsiooni tööde ülevaatusel tootejuhtide poolt antud kommentaare kui järgmises iteratsioonis realiseerimist vajavaid lugusid ja teostasid need. Seega vajas iteratsiooni sisu lõpliku kinnitamise protsess kontrolli lisamist.

Nendest takistustest jagusaamiseks võeti ette mitmeid samme, suurima mõjuga oli seniste osaliselt valitud praktikate asemel kõigi Scrumi praktikate juurutamine. Määrati täpsemalt valmisoleku kriteerium ning pakuti meeskonnale abi tööde lugudeks lahutamise juures. Tehnilise poole pealt kuulutati prioriteetseks koodi lihtne hooldatavus ja laiendatavus. Selleks hakkasid tootejuhid lisaks kasutajatele väärtust pakkuvatele omadustele tööde nimekirja lisama ka tehnilisi omadusi ja prioritiseerima neid samadel alustel kasutajale pakutavate omadustega. Uuesti tõsteti esile heade koodidisaini praktikate kasutamise olulisust. Lõpuks võeti kaustusele lisaks Scrumile ka XP praktikaid (testidel põhinev arendus, paarisprogrammeerimine) ja saavutati selle abil 75% vigade hulga vähendamine seitsme iteratsiooni jooksul.

Kieran Conboy oma meeskonnaga (2010) uuris inimestega seonduvat agiilsele arendusele üleminekul ning tõi välja üheksa ohtu, millega arvestada:

- arendajate hirm, et puudujäägid oskustes paistavad rohkem välja. Agiilsed praktikad toovad üsna selgelt välja meeskonnaliikmete oskused ja arenguvajadused. Kuna võrdlus käib meeskonnaliikmete vahel, siis võib ka väga hea arendaja tunda enda allajäämist veelgi paremale ja seetõttu halvasti. Autorid pakuvad lahenduseks meeskonnas usaldusliku õhkkonna loomist, samuti arenguvõimaluste pakkumist;

- arendaja peab olema meister igal alal, mitte ekspert kitsas valdkonnas. Selliseid inimesi on raske leida ja ka koolitamine on aeganõudev ja kulukas. Inimesed ise tunnevad, et kuna nende areng toimub erinevates valdkondades, jääb puudu põhjalikumate teadmiste ja oskuste tekkimisest spetsiifilisemas valdkonnas ja see tuleb edaspidi tööjõuturul neile kahjuks. Autorite lahendus sellele on, et peab olema nii laiemas diapasoonis valdkondades teatud tase kui ka spetsialiseerumine mingile

kitsamale valdkonnale – üks ei välista teist, nende vahel peab valitsema tasakaal;

- suureneb sõltuvus sotsiaalsetest oskustest. Sellega on seotud mitmed aspektid, näiteks tehnilistelt oskustelt kõrgtasemel inimene ei suuda meeskonna ees esineda või inimene ei suuda eristada, mida rääkida klientidele ja mida mitte. Lahenduseks pakuvad autorid sotsiaalsete oskuste koolitust, ka videokoolitusena. Lisaks, autorite arvates on hea mõte teatud juhtudel (näiteks projekti suure hulga kogemusteta arendajate kaasamisel) ja teatud määral asendada suhtlus dokumentatsiooniga;

- arendajatel puuduvad teadmised ärivaldkonna osas. Agiilse arenduse põhimõte on, et arendaja ei tööta läbi suurel hulgal dokumentatsiooni, et saada selgust vajadustes, vaid küsib selgitusi jooksvalt äri esindajalt. Seeläbi jääb aga äritellijale mulje, et arendusmeeskond on täiesti ebakompetentne ärivaldkonnas ja kaotab usalduse. Lahenduseks pakuvad autorid, et arendajate palkamisel tuleks ärivaldkonna tundmisele samuti tähelepanu pöörata, lisaks tehnilistele ja muudele oskustele. Samuti on kasuks arendajate koolitamine ärivaldkondades;

- vajalik on mõista agiilse arenduse väärtusi ja printsiipe, mitte tunda vaid praktikaid. Autorite uuring näitas, et uuritud juhtudest kümne puhul ei vastanud agiilseid praktikaid juurutades tulemus siiski agiilse arenduse põhiväärtustele (näiteks meeskonna koosolekud toimusid, kuid õhkkond oli teineteise suhtes väga kriitiline, mitte toetav). Autorite pakutud lahendused on ühekordse koolituse asemel pidevad koolitused ning koolitustele täienduseks arendava juhtimise (i.k. termin *coaching*) kasutamine, kus juht hoiab kogu aeg fookust väärtustel ja printsiipidel ning suunab meeskonda nende omaksvõtu poole;

- arendajatel puudub motivatsioon agiilsete metoodikate omaksvõtuks. Seda tuleb ette pigem ettevõtetes, kus otsus agiilse lähenemise kasuks ei ole tehtud meeskonna enda poolt, vaid tulnud juhtidelt. Arendajad võivad pidada muudatusi kurnavaks, keeruliseks ning liigselt aeganõudvateks. Autorite soovitus on levitada agiilsele arendusele ülemineku edulugusid, mis julgustavad meeskonnaliikmeid ümber kohanduma;

- otsustusprotsesside muutumine, kus otsuseid ei tee enam juht, vaid kogu meeskond. On oht, et meeskonnas ei teki iseorganiseerumist, vaid kaos, samuti võib juht sattuda segadusse oma rolli osas. Autorid soovitavad siin uuritud ettevõtete praktikaid, nagu igale meeskonnaliikmele nädalas 15 minutine neljasilmakohtumine juhiga, kus ta saab esitada oma mured või arvamused, mida kogu meeskonna ees pole soovinud teha. Otsustusprotsess oli enamikes autorite uuritud ettevõtetes demokraatlik hääletus, kus igaühel oli võimalik oma arvamus öelda ja selle poolt hääletada. Teatud juhtudel (kolme ettevõtte puhul) kasutati ka praktikat, kus projektijuht oli meeskonnas vahendaja rollis ning lülitus peale meeskonna poolt probleemi arutamist teise (juhi) rolli, milles tegi lõpliku otsuse. See annab võimaluse olla võrdväärne meeskonnaliige, säilitades siiski vastutuse otsuste eest;

- vajalik on agiilse arendusega sobiv inimeste hindamine. Meeskonnatöö juurutamine on raske, kui mõõdikud on isikupõhised. Tehniliste oskuste kõrval peaksid mõõdikud arvestama ka teiste agiilse arenduse jaoks oluliste omadustega nagu sotsiaalsed oskused, loov mõtlemine ja iseorganiseerumine. Ära ei tohi unustada ka äritellija esindaja hindamise puhul tööd, mis ta arendusmeeskonnaga koos tegi – autorite uuringu põhjal leiti, et pigem on suundumus hinnata selle rolli puhul ainult tema põhitöö tulemusi. Seega, hindamisel tuleks pöörata tähelepanu mõõdikutele, mis toetavad agiilset arendust, kasuks tuleb tulemustasude programm, mis on rohkem meeskonnapõhine kui isiklikule tulemusele keskendunud. Meeskonnapõhise hindamise tulemuslikumaks muutmiseks on soovitus juhi poolse hindamise asemel liikmete üksteise hindamine ja ise meeskonna poolt hinnatav olemine (näiteks 360 kraadi tagasiside uuring);

- värbamispõhimõtted ei arvesta agiilseks arenduseks sobivate oskuste ja omadustega ning koolide IT erialade lõpetajatel puudub kokkupuude agiilse arendusega. Heaks praktikaks pidasid autorid kolme uuritud ettevõtte värbamisel kasutatavaid praktikaid, kus kandidaadile anti ülesandeid, mille lahendamine sarnanes igapäevase agiilse arenduse meeskonnaliikme tööga. Protsessi jälgides ja tulemusi hinnates saab nii üsna hea pildi, kuidas värvatav võiks edaspidi meeskonnas hakkama saada.

Veel ohte:

- pideva integreerimise keskkonna loomine on keeruline, kui tegemist on erinevate platvormide ja süsteemidevaheliste sõltuvustega (Svensson, Höst 2005). Selle ohu vastu aitab süsteemidevahelisel suhtlusel kasutada teenuspõhist (SOA) või sündmuspõhist (EDA) arhitektuuri, kuna süsteemidevahelised liidestused on väga selgepiirilised ja standardsed ning kasutus kontrollitav;

- agiilsel arendusel ei ole võrreldes monumentaalmetoodikatega arhitektuuri disain samavõrra fookuses, mis võib viia halbade süsteemidisaini otsusteni (McBreen 2003; Stephens, Rosenberg 2003). Selle probleemiga aitab toime tulla üldine rakenduse arhitektuuri kavand, kuid säilib oht, et paljude samaaegselt töötavate arendusmeeskondade puhul jääb seoste juhtimine arendusmeeskondadel omavahel ja üldise kavandiga nõrgaks, mis viib siiski kirjeldatud ohu realiseerumiseni (Petersen, Wohlin 2009). Meie ettevõtte näitel on samaaegsete meeskondade hulk piisavalt väike, et koordineerimisprobleeme ei teki;

- agiilsed arendusprojektid ei skaleeru hästi (Cohen jt 2004) Petersen ja Wohlin leidsid oma töös, et kuigi väikesi arendusmeeskondi on kergem juhtida ja mõõta, on omakorda suurt vaeva nõudev väikeste meeskondade omavahelise kommunikatsiooni ja koordineerimise korraldamine (Peters, Wohlin 2009). Jällegi, nagu eelmise ohu puhul, on meie ettevõtte kontekstis samaaegsete arendusprojektide meeskondade arv väike ja selliseid probleeme ei teki;

- paarisprogrammeerimist peetakse ebaefektiivseks ja ressurssi raiskavaks (Ilieva jt 2004; MacKenzie, Monk 2004; Tessem 2003). Ma näen sellel probleemil kaht tahku – üks on meeskonna enda arvamus ja teine huvipoolte (tellijad, juhtkond) arvamus. Meeskonda planeeritud kahest Elioni enda programmeerijast üks on paarisprogrammeerimise kasus veendunud, teine mitte eriti. IT arenduse juhtkond on paarisprogrammeerimise teed valmis minema, projekti käigu võib aga välja tulla tellijaid, kes pole selle praktika kasus veendunud ja võivad hakata nõudma praktikast loobumist. Nii meeskonnaliikme kui tellijate puhul näen lahendusena tutvustada teiste edulugusid (ka minu töös on mitmes viidatud juhtumianalüüsis

selliseid näiteid). Kui projekti edenemisel tekivad juba meie enda edulood, siis saab kasutada ka neid;

- meeskonnaliikmed peavad olema kõrge kvalifikatsiooniga, et agiilsed projektid õnnestuksid (Merisalo-Rantanen 2005). Selle ohuga aitab toime tulla asjaolu, et suurem osa meeskonnaliikmeid on partnerfirmast, millel on pikaajaline kogemus XP kasutamisel ning töötajad heade oskustega. Elioni arendajad hakkavad töötama koos partneritega ja nii tekib teadmiste ja oskuste ülekandumine;

- meeskonnad on kõrge sidususega, mis tähendab meeskonna sees head infovahetust kuid sellevõrra kannatab meeskondadevaheline suhtlus (Karlström, Runesson 2005; Pikkarainen jt 2008). Selle ohuga toimetulekut olen selgitanud juba eelmiste punktide juures, meil ei ole korraga suurt hulka erinevaid meeskondi töötama planeeritud. Lisaks on hea praktika meie arenduspartneri kasutatav igapäevane infokoosolek, kus üle erinevate projektide kõik arendajad annavad kõigile infot, mitte ainult oma meeskonna sees. Seda saame ettevõttes omakorda täiendada üle e-kanalitega tegelevate meeskondade (nii äri- kui arendusmeeskondade) ühise infokoosolekuga, mis on osaliselt juba juurutatud (praegu osalevad juhid ja võtmeisikud, mitte kõikide meeskondade liikmed);

- arendajatele suurema mõjuvõimu andmine esmalt hirmutab juhte ja sellega toimetulek nõuab juhtide piisavat koolitamist (Karlström, Runesson 2005). Meie ettevõtte koostöömudel on ka siiani olnud spetsialiste usaldav, neile initsiatiivi pakkuv ning volitusi ja kohustusi võrdselt andev. Juhtide koolitus Scrumi praktikate kasutamisel on juba ka toimunud;

- rakenduse implementeerimine algab varases projekti faasis, seega tulevad implementeerimisega seotud probleemid välja varakult, mis on aga (projekti) juhtrühma arvates „liiga vara" võrreldes monumentaalmetoodikate kasutamisel nähtuga (Karlström, Runesson 2005). Selle ohuga toimetulekuks tuleb selgitada agiilse arenduse põhimõtteid ja toimimise protsesse ning kirjeldada eeliseid;

- tellija esindaja on pidevalt kaasatud ja peab oma panuse andma kogu projekti kestel, mis suurendab tema koormust võrreldes

monumentaalmetoodikates kasutatud praktikatega ja tekitab stressi (Martin jt 2004). Ühest küljest peab tellija küll projekti kestel rohkem panustama, kuid teisalt ei ole ta teadmatuses projektis tehtavate tööde osas ja ei pea kartma, kas arendaja tõlgendab tema kirjeldatud soove õigesti ja see peaks stressi vähendama. Tellija esindajal peab olema juhiga kokku lepitud ja planeeritud ressurss projektis osalemiseks, sarnaselt nagu juhime arendajate ressurssi projektides osalemisel tööplaani alusel neis tunde erinevatele projektidele ja tegevustele (nt puhkused või koolitused) ette planeerides;

- paarisprogrammeerimine pole rakendatav, kui osapoolte tasemevahe on ülisuur (Melnik, Maurer 2002). Tasemevahe peab olema tõesti ülimat suur, et selline oht tekiks (ühele osapoolele täiesti tundmatu programmeerimiskeel vms). See oleks projekti õnnestumisele tõsine oht, kuna meie töötajatest arendajate kogemused Java programmeerimiskeelega on tunduvalt madalamad koostööpartneri töötajatest, samuti puudub igasugune kogemus agiilsete praktikate kasutusest. Tegime mõnepäevase proovisessiooni, et selle riski realiseerumist hinnata ning sessiooni tagasiside oli mõlemapoolselt, et koostöö toimis. Lisaks saame riski realiseerumise ohtu vähendada läbi nõrgemate arendajate koolitamise.

Viimasena käsitlen ohtu, mis on seotud tagasiside ahela viivitusega. Projekti kaks kõige olulisemat tarbijat on Elioni klient (veeb ja e-teenindus) ning Elioni töötaja (siseuudised). Agiilsed meetodid eeldavad kliendi (täieliku pädevusega) esindaja pühendunud kaasatust. Siit kerkib üles probleem, millele on viidanud ka Dzamashvili Fogelström, Gorschek, Svahnberg ja Olsson oma 2010 aasta artiklis. Nimelt agiilsed metoodikad keskenduvad kiire väärtuse loomisele konkreetsele kliendile ja ootavad kliendi esindajalt kohest kinnitust, kas pakutavad lahendused on seda. Meil ei saa aga lõplikku hinnangut koheselt keegi anda. Parim, mida enne arendust saab teha, on abivahendite nagu GAP analüüsi, kliendile pakutava väärtuse analüüsi (CVA ehk *Customer Value Analysis*) ja fookusgruppide kasutamine (Gorschek 2006).

Sellises situatsioonis on soovitatav ette võtta täiendavaid meetmeid projekti eelsetes tegevustes seoses nõuete valiku ja planeerimisega (Dzamashvili Fogelström, Gorschek, Svahnberg ja Olsson 2010). Üks neist on versiooni planeerimine nii, et pool mahust on omadused, mis kindlasti selle versiooniga

realiseeritud peavad saama. Näitena võib tuua omadused, mis on eelduseks teiste projektide jaoks, omadused, mida on kasutatud reklaamikampaaniates vms.

Riski kirjeldus	**Riski tagajärg**	**Vältimise võimalus**	**Tegevused realiseerumisel**
Agiilse metoodika juurutuse käigus esilekerkivate probleemide mittemärkamine ja nende eskaleerumine.	Agiilne arendusmetoodika ei juurdu, projektid ebaõnnestuvad.	Juurutusprotsessi juhil olla kursis võimalike tekkida võivate probleemidega ja nende sümptomitega, olla kursis meeskonna tööga. Meeskond ise on teadvustatud sellisest ohust ja jälgib samuti.	Probleemide esile kerkides lahendame need koos meeskonnaga.
Lugude kvaliteet pole piisav, puuduvad aktsepteerimiskriteeriumid.	Arendajal pole selge, mida teha vaja on. Ajakulu täiendavatele täpsustustele või ümbertegemisele.	Tootejuhi koolitamine, koos meeskonnaga kogemustest õppimine.	Tootejuhi koolitamine, koos meeskonnaga kogemustest õppimine. Väliste konsultantide kaasamine.
Tootejuhi vähene koolitus.	Tootejuhi roll projektis ei saa täidetud piisavalt kvaliteetselt (lugude kirjeldamine ja prioritiseerimine), projekt ei edene mõistliku kiirusega.	Tootejuhtide varajane ja põhjalik koolitamine.	Tootejuhtide koolitamine, kogenud meeskonnaliikmete tugi, välise konsultandi kaasamine.
Arendaja hirm oma puuduste osas (agiilsed metoodikad toovad oskusi ja nende puudusi selgemalt esile).	Arendaja moraal langeb, efektiivus väheneb ning ta võib ka meeskonnast lahkuda.	Meeskonnasisese usaldusliku meeleolu loomine, läbi mille väheneb hirm ning võimaluste pakkumine oma oskuste parandamiseks.	Arendajaga arutamine, mida ta vajab, et end kindlamalt tunneks (koolitus).

		Tagasiside nelja silma all. Isiklik mentor uutele meeskonnaliikmetele. Paarisprogrammeerimisel paaride jagunemine nii, et uued liikmed on paaris kogenud liikmetega.	
Suurenenud sõltuvus arendaja sotsiaalsetest oskustest.	Nõrgemate sotsiaalsete oskustega inimesed ei suuda panustada oma tavalisel tasemel, efektiivsus langeb ning võivad ka lahkuda.	Koolitusprogrammides tuua sisse meeskonna enda poolt pakutud teemad, mis aitavad arendada sotsiaalseid oskusi. Kasutada piisavat (kuid mitte liigset) dokumentatsiooni, mis toetaks just arendusmeeskonna enda vajadusi.	Värbame sobivaid inimesi.
Otsustusprotsesside muutumine juhi poolt otsustamiselt meeskonna poolt otsuste vastuvõtmiseks ei hakka tööle.	Otsustamatus, halvad otsused, projekti kontrolli alt väljumine.	Selge meeskondliku otsustuskorra väljatöötamine ja meeskonnaga kokkuleppimine, et vältida kaost. Demokraatlik hääletamine. Juht ei ole meeskonnatöös juhi rollis, vaid võrdväärne osaline. Neljasilma vestluse aeg igal liikmel oma juhiga igal nädalal.	Sama, mis vältimisel, tuleb rohkem panustada.
Meeskonna normaalne omadus on olla kõrge sidususega.	Meeskondadevahelin e infoliikumine on pärsitud.	Kõiki osapooli kaasavad ärilised ja tehnilised infokoosolekud, kus räägitakse teemadest üle kõigi projektide.	Kui selgub, et koosolekud pole piisavad, siis arutame koos meeskondadega, mis infovahetust

			pärsib ja kuidas sellest vabaneda.
Projekti kulgemise mõõtmine toimub valesti.	Ei saa aru projekti edenemise seisust, meeskonnaliikmetele läbimõtlematult/valesti määratud mõõdikud suunavad kõrvale kokkulepitud eesmärkide saavutamisest.	Läbimõeldud, meeskonnaga läbi arutatud, tegelikke eesmärke toetavad ja soovitavalt meeskondlikud mõõdikud.	Mõõdikute korrigeerimine.
Ei käivitu väljalasete planeeritud tempos valmimine.	Esimest väljalaset tuleb oodata plaanitust kauem, järgmised väljalasked hilinevad võrreldes plaaniga.	Plaan tuleb koostada reaalsetele võimaluste vastav, mitte nimetada unistust plaaniks. Regulaarsed meeskonna arutelud (sh igapäevased koosolekud ning tagasivaatekoosolekud), kus probleemide kerkimisel neist räägitakse.	Meeskonnaga koos arutada põhjuseid, objektiivsete põhjuste korral korrigeerida plaani, meeskona töös kitsakohtade leidmisel lahendada neid koos meeskonnaga ühiselt.
Pideva integreerimisjärgse testimise käivitamine ei õnnestu.	Vead hakkavad kuhjuma.	Probleemi tekkimise võimaluse fookuses hoidmine. Meeskond jälgib oma liikmete tööd (iseorganiseeruv meeskond).	Vajadusel panustada rohkem, kaasata abijõude. Selle riski realiseerumist ei tohi mingil tingimusel lubada.
Mittefunktsionaalsed nõuded ei saa piisavalt tähelepanu.	Mittefunktsionaalsete nõuete mittearvestamine toob lisatööd ja tekitab segadust.	Jälgida mittefunktsionaalsete nõuete õigeaegset lisamist tööde nimekirja.	Arutada koosmeeskonnaga läbi, kuidas sellist probleemi edaspidi vältida saab. Hetkel

			üleval olevad probleemid lahendada.
Pikad viivitused tagasiside ahelas.	Arendaja ei saa kiiret tagasisidet realiseeritud loo kohta, kas see on aktsepteeritud.	Iteratsiooni lugude valimise eel vajalike tegevuste tegemine (analüüsid, sh kasutatavuse analüüs, fookusgruppide kasutamine), iteratsiooni planeerimine nii, et pooled lood on märgitud kui kindlasti realiseerimist vajavad ja teine pool on töid, mis kiiremate lugude (eelnevate iteratsioonide lugude muudatuste/täiendust e) ilmumisel võivad edasi lükkuda.	Leppida koos meeskonnaga (sh tellija esindaja) kokku protsess, kuidas tagada pika tagasiside ahela juures esmane aktsepteerimine ja kuidas edasisi muudatusi teostatud lugudes käsitleda (uute lugudena?) ja see juurutada.

Tabel 4. Riskianalüüsi tähtsamate riskide ülevaade

Riskianalüüsi tulemuste põhjal leidsin, et arendusmetoodika vahetusega seotud riskidest ka kõige suurema tähtsusega riskid ei ulatu oma väärtuselt äärmiselt kõrge tähtsusega riskide piirkonda. Tähtsuse definitsioon on mõju ja tõenäosuse korrutis. Leidsin 12 riski, mille üks komponentidest on kõrge tasemega ja teine keskmise tasemega ning esitan tabelis 4 riskide ülevaate, vältimise võimaluse ja tegevused realiseerumisel.

4.3 Koordineerimine

Oluline on pöörata tähelepanu meeskondadevahelise töö korraldamisele. Projekti tipphetkedel räägime kolmest-neljast erinevast meeskonnast, kellel igaühel on oma suhteliselt iseseisev alamprojekt. Tööde koordineerimisel ja väljalasete planeerimisel tuleb arvestada ühiste ressurssidega. Uuringud on leidnud, et tarkvaraarenduse meeskondades on jõudlus seotud meeskondade töö koordineerimisega (Hoegl, Gemuenden 2001; Kraut, Streeter 1995).

Monumentaalmetoodikates kasutatava plaanipõhise lähenemise juures on töö koordineeritud hierarhiliselt, selgelt rolle eristava käsu ja kontrolli põhimõtte järgi (Nerur, Balijepally 2007; Nerur jt 2005). Agiilse lähenemise korral koordineerib tööd iseorganiseeruv meeskond, kus meeskond ise otsustab koordineerimisviisi üle (Boehm, Tumer 2003). Meeskonnad, kus seni on kasutatud plaanipõhist lähenemist, koosnevad tavaliselt oma alameesmärgile keskendunud ja oma tegevusi suhteliselt vabalt ise planeerivatest spetsialistidest ning üleminek iseorganiseeruva meeskonna praktikale on üks suuremaid väljakutseid agiilsele arendusele ülemineku juures (Nerur, Balijepally 2007). Ei organisatsioonikultuur ega inimeste mõtteviis ei ole lihtsalt muudetavad, mis teeb ülemineku paljudele ettevõtetele veelgi hirmutavamaks (Boehm, Tumer 2003). Inimesi määrata gruppi ja seda sildistada „iseorganiseeruvaks" pole piisav, ei saa oodata, et selle järgselt inimesed automaatselt oskaks ise koordineeruda ja töötada kui agiilne meeskond (Moe jt 2009). Elionis on organisatsioonikultuur meeskonnasisest vastutusevõtmist toetav, näiteks enamasti meeskond ise koostab reaalse projektiplaani, see ei tule juhtidelt. Teisalt on inimestel tänastes meeskondades kindlad rollid sõltuvalt nende professionaalsetest oskustest ja spetsialiseerumisest teatud valdkonnale. Kui sarnaseid spetsialistide rolle on projektis mitu, siis on näiteid ka iseorganiseerumisest, kuid enamasti on täna inimestel oma kindel osa projektist, mida nad vastavalt plaanile täidavad. Projekti juhib lõppkokkuvõttes siiski täna projektijuht, mitte meeskond koos. Oluline on silmas pidada, et suunatus oleks meeskonnale, mitte ülesandele (Schatz, Abdeldshafi 2005). Meeskond on see, kes koos ülesandeid lahendab ja seeläbi sünergia saavutab. Ilkka Lehto ja Kristian Rautiainen (2009) leidsid keskmise suurusega tarkvaraettevõttes Scrumi juurutades kaks koordineerimisega seotud probleemi – tooteomanike ja tiimide vahel ning projekti ajakavas olemise tagasisideahelates. Tooteomanike ja meeskondade vahel tekkis olukordi, kus mitu eri meeskonda vajasid sama tooteomanikuga tihedat koostööd ja meeskondade töö efektiivsus sai kannatada. Esines ka teistpidi probleemi, mitu tooteomanikku olid oma nõuetega sama meeskonna töösoleva iteratsiooni planeeritud tööde nimekirjas (Scrumi spetsiifiline termin, i.k. *sprint backlog*) ning meeskonna töö oli häiritud liigtihedate koosolekutega. Meie projektis ei ole ette näha, et mitmel tooteomanikul oleks samale meeskonnale töid, küll võib ette tulla sama tooteomaniku koostöö mitme meeskonnaga samaaegselt. Siin on lahenduseks analüütiku rolli ja volituste laiendamine nii, et arendusmeeskond saab teda pidada tellija vahendajaks (i.k termin *customer proxy*). Analüütiku otsustada on, millal ta peab vajalikuks tooteomaniku poole pöörduda info või kinnituse saamiseks. Veel oli probleemiks, et algselt planeeritud lugude hindamine ja

ümberhindamine ei hakanud soovitud viisil tööle, mistõttu ei olnud võimalik hinnata projekti edasiliikumist. Erinevate projekti planeeritud tööde nimekirjade (Scrumi spetsiifiline termin, i.k. *product backlog*) vahel olid küll viited, kuid mitte dünaamilised.

Elioni e-kanalite arendusprojektides on planeeritud infokiirgurina kasutada veebipõhist rakendust Pivotal Tracker (www.pivotaltracker.com), millele on ligipääs kõigil meeskonnaliikmetel ja vaatlejana ka erinevatel huvipooltel. Erinevate tootejuhtide tööde nimekirjades olevate seotud tööde haldamine oli probleemiks ka Salesforce'i juhtumianalüüsis, seal oli lahenduseks sarnase infokiirguri nimega Scrumforce arendus enda vajaduste lahenduseks (Mencke 2008). Meil on planeeritud, et tootejuhid hoiavad ja haldavad infokiirguris oma tööde nimekirju, töödel on märgitud aktsepteerimiskriteeriumid. Meeskond hindab nimekirjas olevate tööde mahud ning meekonnaliikmed võtavad tööde nimekirjast omale ülesandeid. Antud vahendit kasutab meie arenduspartner ja leidsime teistes projektides pilootkasutuses selle väga sobiva olevat. Ainsaks puuduseks on, et lugusid saab hinnata vaid keerukuse skaalal 0-3, mis ei ole meie jaoks piisavalt detailne. Eelistaksime kasutada täpsemat skaalat, näiteks 0, 1, 2, 3, 5, 8, 13, 20, 40, et iteratsioonide mahtu täpsemalt planeerida. Selleks tuleks aga kasutada vahendi märkuste välja, mis ei ole eriti mugav. Läheme esialgu liikuma vahendi pakutava skaalaga ja võtame märkuste välja hindamiseks kasutusele alles siis, kui tekivad probleemid või leiame teise sarnase infokiirguri, kus skaalat ise võimalik määrata.

Meeskonna iseorganiseerumiseks muutmine ei ole lihtne, nagu eespool mainitud. Iseorganiseeruva meeskonna omadused on (Morgan 2006; Nerur, Balijepally 2007; Nonaka, Takeuchi 1995; Katzenbach, Smith 1993):

- meeskonnale orienteeritus;
- funktsionaalne liiasus, samu funktsioone on võimelised täitma mitmed meeskonnaliikmed;
- võime asendada teist meeskonnaliiget;
- tihe suhtlemine meeskonna sees;
- võime hinnata meeskonnakaaslaste resultatiivsust ja tagasiside andmine teistele meeskonnaliikmetele;
- meeskondlik koordineerimine;
- jagatud eestvedamine;
- pidev arenemine, õppimine;
- suhteliselt madal personaalne autonoomsus, mis on aluseks kõrgele meeskondlikule võimekusele (Langfred 2000, Moe jt 2009).

Nils Brede Moe, Torgeir Dingsųyr ja Tore Dybå toovad oma juhtumianalüüsi (2009) põhjal välja olulised teemad, millele tähelepanu pöörata:

- meeskonnasisene usaldus – usalduse puudumisel raiskavad meeskonnaliikmed aega ja energiat enda kaitsmisele, kontrollimisele ning teiste inspekteerimisele, mis vastandub kõrge usaldustasemega meeskondade väärtustpakkuvale koostööle (Salas jt 2005). Kui keegi liikmetest kaotab teiste usalduse, näiteks „kaaperdades ülesande" ehk tegeldes ülesandega, mida pole endale meeskonna ees võtnud, siis leian, et sellise olukorraga ei tohi leppida, usaldus tuleb taastada või vahetada välja meeskonnaliige. Usalduse puudumine pidurdab ka info jagamist, kui meeskonnaliige kardab ennast teiste poolt ebakompetentseks peetavat (Bandow 2001);
- jagatud eestvedamine – leian, et otsuste tegemine ei tohi muutuda piiratud grupi meeskonnaliikmete õiguseks ja kohustuseks, vaid selle peab osalema terve meeskond. Vastasel juhul ei tunne meeskond ennast eesmärkidega seotud olevat;
- ühine mentaalne mudel – koostöö nõuab meeskonnalt ühiste jagatud mentaalsete mudelite arendamist (Salas jt 2005).

Iseorganiseeruva meeskonna mudelile ülemineku kergendamiseks leian, et tuleb pidevalt jälgida ja hoida fookuses eelpooltoodud omaduste taset meeskonnas ja vajadusel võtta ette samme korrigeerimiseks. Mõned võimalikud sammud on toodud eespool juhtumianalüüsi (Moe jt 2009) kommentaarides.

4.4 Meeskonna rollid

Elioni e-kanalite projektide eripäraks on asjaolu, et suur osa funktsionaalsusest tuleb liidestada teiste süsteemidega, kus on peidus ka äriloogika. Ettevõtte e-kanalite programmeerijatel puudub kogemus, kuidas interpreteerida päringutele saadavaid vastuseid ja kas need on (ärilises mõistes) adekvaatsed või mitte. Seetõttu on meeskonna liige ka analüütik, kelle ülesandeks pole programmeerida, vaid kirjeldada (tellida väliselt süsteemilt vastavalt vajadustele) liidestusteenuseid

ja interpreteerida teenuse sisendeid ja väljundeid. Näiteks, kui kliendi teenusele pakutakse mitut hinnapaketti, siis analüütik oskab vastata küsimusele, milliste parameetrite väärtuste komplekti puhul on lubatud milline hinnapakett. Elionis on olemasolevate ärireeglite kompetents enamasti analüütikutel (domeenipõhiselt) ja selle edasiandmine teistele rollidele võtaks aastaid. Eksisteerib suur hulk erinevaid seoseid, mis määravad lõpliku ärireeglite komplekti. Analüütik täidab vajadusel ka äritellija vahendaja (i.k. *Customer proxy*) rolli – kui töö käigus on vaja vastu võtta ärilisi otsuseid, siis esindab analüütik äritellijat (vajadusel konsulteerides tollega). Mõistlik meeskonna koosseis on ühe alamprojekti jaoks kaks kuni neli programmeerijat ja üks analüütik ning projektile kokku üks-kaks koguahela testijat, kes kirjutavad ka süsteeme läbivaid automaatteste. Seniste projektide kogemuse baasil leian, et üks analüütik saab osaleda täiemahuliselt ühes meeskonnas ja selle kõrvalt teha muid, madalama prioriteediga ülesandeid. Põhjuseks on peatüki alguses toodud asjaolu, et palju informatsioonist on teistes süsteemides ja peidetud kujul, mistõttu arendajatel on mõistlik teha tavalisest tihedamat koostööd analüütikuga.

Ükski agiilsetest metoodikatest ei räägi sõltumatust kontrollist ja valideerimisest (i.k. *independent verification and validation*), kuid nende kasutuselevõtu käigus on teistes ettevõtetes teemaks kerkinud testija rolli sisu. Sisemine testija on programmeerija liitlane, ta aitab täita programmeerija ülesannet tarnida kvaliteetset koodi. Sõltumatu testija seevastu on kliendi liitlane, kes aitab kontrollida, et valmis tehtu on vastav kliendi vajadustele. Agiilsed metoodikad suunavad vastutuse seega programmeerijale ja kliendile ja näevad kvaliteedi tagamise (i.k. *Quality Assurance*) spetsialiseerunud testijale ette uut rolli (Koch 2005, 163).
Kuna juurutame XP metoodika, siis arendus saab olema testidel põhinev ja programmeerijad ise kirjutavad moodultestimiseks testid. Senise kvaliteedi tagamise üksuse testija või testijate rolliks saab olema testida süsteeme läbivate protsesside tervikahelaid ning kirjutada automaatteste, seda koos tellija esindajaga. Tellija teab ärilisi vajadusi ning IT professionaal oskab välja tuua täiendavaid testimist vajavaid aspekte.

Aktsepteerimistestide läbiviimine traditsiooniliste projektide puhul toimub projekti lõpus. See etapp toob tavaliselt hulgaliselt täiendusi ja uusi väljalaskeid. Agiilsete meetodite puhul toimuvad aktsepteerimistestid iga iteratsiooni käigus ja seega mitmeid kordi projekti jooksul. Nii on aktsepteerija projekti jooksul paremini arengutega kursis ja suuri ümbertegemisi on oluliselt vähem. Leian, et oluline on ka

aktsepteerimistesti toimumisaeg versioonivahetuse tsüklis. Varasem töökorraldus oli, et peale väljalaske üleandmist arendaja poolt (tulemus oli programmeeritud ja testitud) tegi Elioni analüütik suitsutesti, et veenduda üleantud muudatuse toimimises ja tellitule vastavuses ning suunas positiivse tulemuse järel põhjalikule testimisele. Testija leitud vead saadeti arendajale, parandati ja testiti uut parandatud versiooni. Aktsepteerimistest toimus alles täiesti toimiva muudatuse alusel ja selleks ajaks oli versioonivahetuse tähtaeg ähvardavalt lähedal – aktsepteerimistesti käigus leitud probleemide lahendamiseks ei jäänud enam aega. Agiilse metoodika kasutuselevõtul saame tellijale tutvustada lahendust varases faasis (juba arenduskeskkonnas kasvõi koos teada vigadega), et võimalikud ärilise vaatega mittesobivused jõuaks veel likvideerida. Samuti on oluline lugudes märgitud aktsepteerimiskriteeriumite hea kvaliteet. Mida parema kvaliteediga on need kriteeriumid lugu kirjeldades määratud, seda vähem tuleb probleeme välja aktsepteerimistestide käigus.

Oluline, kuid mitte igapäevaselt sekkuv roll on lahenduse arhitekt, kes vastutab tehnilise lahenduse optimaalse, tervikliku ja jätkusuutliku rakenduse arhitektuuri eest. Versioonivahetusi korraldab arendusüksuses versioonihaldur, versioonide plaan on vähemalt kvartali jagu ette teada ja avalik. Erakorralisteks versioonivahetusteks etteteatamise nõue on maksimaalselt 48 tundi, minimaalselt kaks tundi. Seega ei saa versioonivahetuste korraldamise protsess takistuseks iteratiivsele tarkvaraarendusele.

Äritellija roll ja kasutatavus

Äritellijaga koos töötab disainimeeskond, kelle ülesanne on prototüüpimine, kasutatavuse hindamine, visuaalne disain ja selle tehniline realiseerimine.

Agiilsed metoodikad räägivad töötava tarkvara tarnimisest, kuid enamasti ei ütle, kellele see hindamiseks üle antakse. Metoodika vaatekohalt on see õige, sest erinevates projektides on tegemist erineva osapoolega. Küll võib selle teema olulisus seetõttu jääda tähelepanuta. Elioni e-kanalite arenduses on osade (kasutus)lugude puhul täpsemalt määratletavaid kasutajaid ka ettevõttest seest (rakenduse administraator, sisutoimetaja jne), kuid enamasti on tegu lõppkasutaja näol inimesega, kes külastab meie e-keskkondi. Nõuete püstitamisel (lugude kirjutamisel) tuleb enamasti lähtuda meeskonna arusaamast, mida kasutaja ootab ja kasutajate tagasisidet saab fookusgruppide kaudu. Senine praktika on olnud

teha aeg-ajalt ennetavalt kasutajate seas uuringuid, millised oleksid kasutajate ootused ühe või teise võimaluse osas.

Elioni kontekstis on tellija alati ettevõttesisene, mis paljuski lihtsustab koostööd tellija ja arendaja vahel. Võimalik on XP nõutav tellija esindaja pidev saadavalolek ja aktiivne arenduses osalemine. Erinevad allikad viitavad, et tihti eeldab tellija, et tema osaleb projektis finantseerijana ning lõpptulemuse aktsepteerijana ja ei taha panustada lisaks jooksvalt oma inimressursiga (see on ju lisakulu). Elionis tahavad tellijad ise projektis rohkem kaasatud olla, et näha jooksvalt tulemusi ja enamasti ka tunnistavad, et lihtsam on mõista tellimust ja seda täiendada juba prototüübi või vähese funktsionaalsusega rakenduse abil võrreldes kogu lähteülesande abstraktselt dokumendiks vormistamisega.

Äritellija (ehk Scrumi terminoloogias *Product owner*) ülesanne on igas projektis hallata tööde nimekirja (Scrumi terminoloogias *Backlog*), määrata töödele prioriteedid ja otsustada iteratsiooni koosseis. Samuti on äritellija ülesanne tagada arendajatele vajalik informatsioon äriliste nõuete osas nii tööde esialgsel kirjeldamisel kui arenduse käigus jooksvalt ning korraldada selle kooskõlastatus huvipooltega. Äritellija püüab sisendiks pakkuda prototüüpe ja sellisel juhul on need disainimeeskonna poolt hinnatud kasutatavuse seisukohast. Vastav praktika on kasutusel Salesforce.com-is (Mencke 2008) ja on ennast seal tõestanud. Prototüüpide puudumisel teeb arendusmeeskond võimalikult varases faasis valmis ka kasutajaliidese, mille saab äritellijale kasutatavuse hindamiseks anda. Michael Budwig annab juhtumianalüüsile tuginedes (Budwig 2009) soovituse valmistada järgmise iteratsiooni (Scrumi termin *sprint*) kasutajaliidese elemendid ette ühe-kahe sellele eelneva iteratsiooni käigus, kuna nii on arendusmeeskonnale tagatud kvaliteetse sisendi olemasolu. Meie projektide kontekstis on kasutajaliidese elementide disain ja kasutatavuse analüüs planeeritud enne äritellija poolt arendusülesande esitamist (ehk toimubki enne vastava arenduse iteratsiooni, kuid arendusmeeskonnal on võimalus pakkuda paremaid ideid ning positiivse otsuse korral läheb ettepanek disainimeeskonnale tagasi kasutatavuse analüüsiks ja disainimiseks. Selleks on vajalik disainimeeskonna ja arendusmeeskonna väga tihe koostöö, millele viitab ka Hegle Sarapuu oma kogemuse põhjal eestimaise arendusprojektiga (Sarapuu 2010).

Erinevate äritellijate tellimusi koordineeriv roll on e-kanalite arendusjuht, kes koordineerib üle kõigi e-kanalite arendusprojektide tootejuhtide tegevust. Sarnase

mudeli on pakkunud Ken Power (2010) oma huvipoolte identifitseerimist ja grupeerimist käsitlevas töös.

4.5 Töökorraldus, kommunikatsioon

Lisaks varem kirjeldatud töökorraldusega seotud praktikate juurutamisele alates paarisprogrammeerimisest toon allpool ka kokkuvõtte, milliseid regulaarseid koosolekuid on plaanis käivitada.

Igapäevaselt:

- igapäevane ülevaatekoosolek. Sisuks info igalt liikmelt, mis tehtud, mis järgmisena tegemisele läheb ja kas on takistusi.

Iteratsioonhaaval:

- iteratsiooni kokkuvõtte koosolek. Vaadatakse koos tellijaga üle iteratsiooni käigus valminud tööd, tuvastatakse lõpetamata lood ja otsustakse nende saatus;
- tagasivaatekoosolek (i.k. termin *retrospective*). See on meeskonnale hea võimalus õppida kogemustest ja saada seeläbi paremaks. Vaadatakse tagasi eelmisele perioodile, mis läks hästi ja mis läks halvasti. Tuvastatakse kitsaskohad senises protsessis ja võimalused protsessi paremaks kohendada;
- järgmise iteratsiooni planeerimine. Tellija pakub välja nimistu järgmise iteratsiooni töödest ja vajadusel selgitab täpsemalt sisu. Meeskond annab hinnangu tööde mahule. Tulemuseks tööde list, mis on plaanitud valmima järgmisse iteratsiooni.

Monumentaalmetoodikate kasutamisel on arendajate kokkupuude juhiga väiksem ning ei ole erandlik, kui juht ja programmeerija ei satu tihedamini rääkima, kui kord nädalas. Agiilsete metoodikate puhul on tavapraktika pigem igapäevane vestlus. Tasub tähelepanu pöörata sellele, et meeskonnaliige ei hakka seda pidama juhi sooviks tugevdada kontrolli töötaja tehtava üle. Selline arvamus võib tekitada vastupanu agiilsele arendusele ülemineku muudatusele, kuna töötajad tunnetavad vestluste käigus jutuks tulevate tööülesannete osas

veel täitamata ülesannete või venivate tähtaegade kohta etteheiteid. Mike Cohn ja Doris Ford (2003) soovitavad selle vältimiseks ja usalduse võitmiseks juhtidel demonstreerida oma soovi kaasa aidata takistuste eemaldamisel ning mitte teha etteheiteid, kui ülesannete täitmine eeldatust rohkem aega võtab. Eespool peatükis 4.2 oli kirjeldatud ka soovitus igale meeskonnaliikmele võimaldada juhiga iganädalaselt vähemalt 15 minutine neljasilmavestlus, et arutada teemasid, mida teiste ees ei soovita arutada (Conboy 2010). Arendusmeeskonnas on juba täna igal liikmel nädalas planeeritud tunnised koosolekud juhiga, mida ressursi puudumisel oleme vahel ka pooletunnisteks lühendanud vastastikusel kokkuleppel.

Bob Schatz ja Ibrahim Abdelshafi (2005) toovad välja nüansi, mida on kirjeldatud ohtude juures. Nimelt agiilset metoodikat juurutades tuleb jälgida, et kasutusele võetaks ka „agiilne keel". Vanade terminite ületoomine uude metoodikasse viib enamasti selleni, et inimesed kipuvad neid vanamoodi käsitlema ja vastavalt sellele ka käituma. Parem anda terminitele uued nimed ja jälgida ühtlasi, et selliseid probleeme ei tekiks.

4.6 Mõõtmine

Tähelepanu tuleb pöörata, kuidas tagada ülevaade projekti kulgemisest nii, et aruandluse tootmine võimalikult vähe häiriks arendusmeeskonda, samal ajal aga oleks tagatud ülevaade projekti seisust ning ohud oleks avastatud võimalikult varakult. Probleemide kiire tuvastamine on oluline, et tagada nende võimalikult väike mõju projektile. Sellele on viidanud ka erinevad juhtumianalüüsid, näiteks Progressive Insurance'i (Jochems, Rodgers 2007) või Primavera Inc (Schatz, Abdelshafi 2005) kogemus. Üks võimalus on mõõdikud lasta defineerida äritellijal (Mencke 2008), sest tema teab kõige paremini, mis mõõdikud kirjeldavad projekti edukust.

Plaanimise ja hindamise vahend, reeglid ja ühikud on midagi, mis kokku leppida meeskonnaga. Viitan siin Marek Kusmini kogemusele, et kui hindamisel hakkas meeskond arenduseks kuluvate tundide asemel hinnanguid andma abstraktsetes ühikutes (loo punktid ehk i.k *story points*) ja kui varem olid hinnangud pigem olenevalt inimesest ala- või ülehinnatud kaaluga, siis abstraktsemaks muutmine hakkas andma palju täpsemaid tulemusi. Mõõtmise metoodika ja ühikud peavad

olema meeskonnale üheselt mõistetavad ja nende vajalikkus selge. Samuti tuleb kokku leppida, mida tähendab „valmis" (kood programmeeritud vs testitud vs integreeritud vs toodangusse antud). Tundide asemel punktide kasutuselevõtt lugude hindamisel ei tähenda seda, et progressi mõõtmine ebatäpsemaks jääks. Paari-kolme iteratsiooniga kujuneb välja arusaam, kui palju punkte meeskond iteratsiooni jooksul realiseerida jõuab ning seda arusaama aluseks võttes saab edaspidiseid iteratsioone juba üsna täpselt planeerida. Meeskonna jõudluse muutudes saab vastavalt korrigeerida ka plaani. Sarnasele kogemusele ettevõttes AgileCo viitavad ka Jayakanth Srinivasan ja Kristina Lundqvist oma juhtumianalüüsis (2010).

Kuigi meeskond on iseorganiseeruv, siis saan juhina jälgida, et meeskond selle initsiatiivi ka vastu/omaks võtaks.

Tüüpiline agiilse projekti edenemise raport sisaldab nimekirja projekti võtmekuupäevadest, lühikest kokkuvõtet projekti hetkeseisu kohta, peamiste mõõdikute (nt uute vigade juurdevool, läbitud testide protsent) väärtused, peamiste riskide loetelu ning langev graafik (i.k. termin *burndown chart*), mis võrdleb teha jäänud töö hulka algselt planeerituga. Mike Cohni ja Doris Fordi kogemuste põhjal on see enamike juhtide jaoks piisav (Cohn, Ford 2003). Marek Kusmin kirjeldab oma kogemust, kus langev graafik lugude lisandumisel või äralangemisel tegi järske langusi, hüppeid või hoopiski püsis pikalt samal tasemel, kuigi meeskond tegi tublisti tööd. Selline graafik ei anna adekvaatset infot projekti kulgemisest ja hetkeseisust ning seepärast soovitab Marek Kusmin kasutada pigem tõusvat graafikut (i.k. *burnup chart*), kus lugude lisandumine või äralangemine ei mõjuta selliselt graafiku joont, vaid nihutab lõpp-punkti asukohta.

Leitud vigade puhul on oluline, et oleks hinnatud ka umbkaudne parandamisele kuluv maht (loo punktides). See aitab hinnata, kas versiooniga ollakse graafikus. Christopher O' Connor viitab (2010) samuti sarnasele probleemile ja selle mõjule projekti kulgemisele oma juhtumianalüüsis.

Kokkuvõttes, mõõtmise juures kavatseme järgida eeltoodud aspekte arvestades LSD praktikat „Mõõtmine" („saad, mida mõõdad", „mõõda meeskonda, mitte persoone").

4.7 Muud juurutustegevused

Versioonihalduse protsessis pole plaanis suuri muudatusi ette võtta, kuna agiilne arendus sobib hästi olemasolevasse protsessi. Suurim muudatus on arenduskeskkonnas pideva integreerimise (i.k. *continous integration*) sisseviimine, et tagada erinevate arendajate tehtud muudatuste kiire sisseviimine koodi. Ka selle juurutamisega on juba alustatud.

Metoodika valiku analüüsi käigus valisin XP ja Scrumi kõrvale ka LSD-st „Lepingute", „Viivituse hinna" ja „Mõõtmiste" praktikad. Mõõtmisest oli juttu eelmises osas, käesolevas selgitan ülejäänud kahe juurutamist.

„Viivituse hinna" praktika pakub välja, et kõiki projektiga seonduvaid otsuseid saab viia ühele võrreldavale alusele, mis on finantsmõõde. Nii saab projekti meeskond otsuseid vastu võtta võrreldes iga lahendusvariandi rahalist väärtust (tulu ja kulu). Meil on ettevõttes kasutusel projektide nüüdisväärtuse arvutus, seda abiks võttes ja kohendades saame hakata hindama projektide asemel ka projektisiseselt otsuste hinda.

Arenduspartneritega lepinguid sõlmides peame silmas vajadust agiilse arenduse puhul teistsuguste lepingute järgi, kui monumentaalmetoodikate puhul. Kasutame siin LSD „Lepingute" praktikat.

Kokkuvõte

Elion Ettevõtted AS-i (edaspidi Elion) jaoks üks olulisemaid probleeme IT ja tootearenduses on madal arendusvõimekus. Võimaliku lahendusena nähakse teostusmeeskonna suurendamist ja/või arendusmetoodika muutmist. Strateegiline otsus e-kanalite üleminekuks agiilsele arendusele on juhtkonna poolt tehtud ning töö eesmärk on leida, milline agiilne arendusmetoodika (või erinevate metoodikate süntees) sobiks Elioni jaoks kõige paremini ning välja töötada lahendused valitud metoodika juurutamiseks.

Magistritöö teema on ettevõtte jaoks väga oluline, kuna ettevõttel puudub varasem sarnane metoodika muutmise kogemus. Juhtkonna soov on, et käesolev magistritöö aitaks antud muudatust läbi viia.

Püstitasin magistritööle eesmärgi leida Elionile sobiv agiilne arendusmetoodika olemasolevate hulgast või sünteesida see erinevate metoodikate praktikatest ning töötada välja põhimõtted, kuidas valitud metoodikat juurutada ning millele selle käigus tähelepanu pöörata, millised on kaasnevad ohud ja kuidas neid vältida.

Töö käigus teostasin enamlevinud agiilsete metoodikate võrdleva analüüsi Elionile sobivuse osas, kasutades selleks Alan Kochi (2005) pakutavat meetodit, kus erinevate metoodikate praktikad jaotatakse agiilse arenduse manifesti printsiipide kaupa, mida vastav praktika enim toetab. Nii on metoodikate erinevad praktikad ühel alusel võrreldavad. Juurutuspõhimõtete väljatöötamisel kasutan kvalitatiivset analüüsi, kuna eksisteerivad mitmed sarnased arendusmetoodika muutmise kogemused teistes ettevõtetes ja on mõistlik analüüsida sealseid kogemusi, arvestades Elioni eripäradega.

Agiilsete metoodikate võrdleva analüüsi tegin Elioni konteksti arvestades ning kasutades nii „sobib halvasti – sobib hästi" skaalat (peatükk 3.2 kirjalikus osas) kui „keeruline juurutada – lihtne juurutada" skaalat (täielik analüüs Lisa 1 töövihikus, kokkuvõte tabelis 2). Analüüsi tulemused on välja toodud peatükis 3.3.

Analüüs näitab, et Elionile sobivuse poolest on teistest metoodikatest tunduvalt sobivam ekstreemprogrammeerimine (i.k. *Extreme programming, XP*), hästi sobivad ka Scrum ja adaptiivne tarkvaradisain (i.k. *Adaptive Software Development, ASD*). Juurutamise lihtsuse analüüs näitab, et juurutada on lihtsam XP-d ja ASD-d, väikese vahega järgneb Scrum. Kuna XP praktikad on rohkem

tehnilist laadi, siis on mõistlik valida kõrvale praktikaid, mis keskenduvad mittetehnilistele aspektidele ja selleks sobib hästi Scrum oma „Igapäevase koosoleku", „Sprindi ülevaatuse" ning „Sprindi tagasivaate" praktikatega. Täiendavalt pakun kombineerida XP praktikad „Kogu meeskond", „Planeerimismäng", „Süsteemi metafoor" vastavate Scrumi praktikatega „Scrumi meeskond" (just tootejuhi roll sellest praktikast), „Sprindi koosolek", „Toote tööde nimekiri". Lisaks valin sünteesitud metoodikasse LSD-st (i.k. *Lean Software Development*) „Lepingute", „Viivituse hinna" ning „Mõõtmiste" praktikad kui unikaalsed ja Scrumi-XP kooslust hästi täiendavad.

Läbiviidud analüüsile tuginedes teen ettepaneku juurutada sünteesitud metoodika, mis sisaldab XP metoodikat täismahus ning lisaks XP praktikaid täiendavaid praktikaid Scrumist ning LSD-st.

Analüüsisin töö käigus mitmetes allikates toodud kriitilisi edutegureid ning pakun välja Elioni kontekstis kriitilised edutegurid:

- tootejuhi rolli käivitumine, sh oskuse tekkimine lugusid õigesti kirjutada ja vastutuse võtmine toote omaduste nimekirja haldamise eest;
- tellijaga koostöö ja tema piisav pühendumus;
- meeskondlikku tegutsemist ja koostööd soodustava keskkonna loomine;
- meeskonnaliikmete tehniliste ning läbirääkimis- ja suhtlusoskuste olemasolu;
- iseorganiseeruva meeskonna käivitumine;
- juhtimisprotsesside muutmine agiilsele arendusele vastavaks;
- tarnestrateegia „vara ja tihti" käivitumine;
- pärandsüsteemidega liidestamise edukus;
- probleemide varajane äratundmine ja nendega tegelemine.

Töötasin läbi mitmed juhtumianalüüsid ja kogusin erinevates allikates toodud võimalikud ohud ja riskid kokku riskianalüüsi (lisa 2). Riskianalüüsis hindasin riski tähtsust ja kirjeldasin tegevusi riski vältimiseks ning selle realiseerumisel riskiga toimetulekuks. Olulisemad riskid, tagajärjed, nende vältimise võimalused ja tegevused realiseerumisel on välja toodud tabelis 4.

Olulisemad riskid:

- agiilse metoodika juurutamise käigus esilekerkivate probleemide mittemärkamine ja nende eskaleerumine;
- tootejuhi vähene koolitus;
- lugude kvaliteet pole piisav, puuduvad aktsepteerimiskriteeriumid;
- mittefunktsionaalsed nõuded ei saa piisavalt tähelepanu;
- pikad viivitused tagasiside ahelates takistavad lugude valmimisjärgset lõpetatuks kuulutamist;
- arendaja hirm oma oskuste puuduste osas;
- suurenenud sõltuvus arendaja sotsiaalsetest oskustest;
- otsustusprotsesside muutumine, meeskonna poolt otsustamine ei hakka tööle;
- meeskonna sidusus pärsib infoliikumist teiste meeskondadega;
- projekti kulgemise mõõtmine toimub valesti;
- ei käivitu väljalasete planeeritud tempos valmimine;
- pideva integreerimisjärgse testimise käivitumine ei õnnestu;

Tabelis 4 on kirjeldatud ettepanekud nende riskide vältimiseks ning pakutud tegevused riskide realiseerumisel.

Leian, et töö täidab oma eesmärgi, sest arendusmetoodika kandidaat on sünteesitud ja välja pakutud, samuti on välja töötatud juurutamise põhimõtted ning välja toodud riskid ning pakutud tegevused nende vältimiseks ning tegevused riski realiseerumisel.

Summary

An Objective of this Master thesis is to find appropriate agile software development methodology for electronic channels (E-channels) development in Elion Ettevõtted AS (Elion) and establish principles for adoption. The results of this Master thesis are great value for Elion, as decision for adoption has been made, transition must start and there is no such experience in organisation today.

Author describes current situation, which is starting point of transition and evaluates organizational culture of Elion from "ready to transition" point of view.

To analyze different agile methodologies, author has chosen Alan Kochs method described in his book "Agile Software Development". Authors evaluation considers also suitability for Elions needs and adoption complexity in given environment.

Conclusion of evaluation is to adopt synthesis from Extreme Programming methodology and parts from Scrum and Lean Software Design methodologies.

Author also evaluates critical success factors and threats in transition and finds out, which of them are important to Elion. Principles for coordination between different teams and working in teams including roles, communication and measurements have been worked out.

Kasutatud kirjandus

Ambler, S. (2009). Agility at Scale Survey Results from the November 2009 DDJ State of the IT Union Survey. http://www.ambysoft.com/surveys/stateOfITUnion200911.html, 19.11.2010.

Bandow, D. (2001). Time to create sound teamwork. The Journal for Quality and Participation 24 (2)lk 41–47.

Boehm, B.W., Turner, R. (2003). Balancing Agility and Discipline: a Guide for the Perplexed. Addison-Wesley.

Burns, T., Stalker, G.M. (1994). The Management of Innovation, rev. ed., Oxford Univ. Press.

Budwig, M., Jeong, S., Kelkar, K. (2009). When User Experience Met Agile: A Case Study. CHI '09: Proceedings of the 27th international conference extended abstracts on Human factors in computing systems, lk 3075-3084.

Chow, T., Cao, D. (2007). A Survey Study of Critical Success Factors in Agile Software Projects. The Journal of Systems and Software, n.81, , lk 961–971.

Cockburn A., Williams, L. (2000). The Costs and benefits of pair programming. Technical Report, jaanuar 2000.

Cohen, D., Lindvall, M., Costa, P. (2004). Advances in Computers, Advances in Software Engineering. An Introduction to Agile Methods. Elsevier, Amsterdam.

Cohn, M., Ford, D. (2003). Introducing an Agile Process to an Organization. Computer vol 36, nr 6, (juuni) , lk 74-78.

Conboy, K., Coyle, S., Wang, X., Pikkarainen, M. (2010). People Over Process: Key People Challenges in Agile Development. IEEE Software, 02. sept., IEEE computer Society Digital Library.

Dzamashvili Fogelström, N., Gorschek, T., Svahnberg, M., Olsson, P. (2010). The impact of agile principles on market-driven software product development. Journal of Software Maintenance and Evolution: Research and Practice, 22,lk 53–80.

Gorschek, T., Wohlin, C. (2006). Requirements abstraction model. Requirements Engineering 11(1),lk 79–101.

Hoegl, M., Gemuenden, H.G. (2001). Teamwork quality and the success of innovative projects: a theoretical concept and empirical evidence. Organization Science 12 (4) lk 435–449.

Ilieva, S., Ivanov, P., Stefanova, E. (2004). Analyses of an agile methodology implementation. Proceedings of the 30th EUROMICRO Conference (EUROMICRO 2004), lk 326–333.

Jochems, R., Rodgers, Sh. (2007) The Rollercoaster of Required Agile Transition. lk 229-233, AGILE 2007 (AGILE 2007)

Karlström, D., Runeson, P. (2005). Combining agile methods with stage-gate project management. IEEE Software 22 (3), lk 43–49.

Katzenbach, J.R., Smith, D.K. (1993). The discipline of teams, Harvard Business Review 71 (2) , lk 111–120.

Kraut, R.E., Streeter, L.A. (1995). Coordination in software development, Communications of the ACM 38 (3) lk 69–81.

Langfred, C.W. (2000). The paradox of self-management: individual and group autonomy in work groups, Journal of Organizational Behavior 21 (5), lk 563–585.

Lehto, I., Rautiainen, K. (2009). Software Development Governance Challenges of a Middle-Sized Company in Agile Transition. SDG'09, 17. mai, Vancouver, Canada.

Leis, P. (2001). „Agiilmetoodikad", A&A nr .04

MacKenzie, A., Monk, S.R. (2004). From cards to code: how extreme programming reembodies programming as a collective practice. Computer Supported Cooperative Work 13 (1), lk 91–117.

Martin, A., Biddle, R., Noble, J. (2004). The xp customer role in practice: three studies. In: Agile Development Conference, lk 42–54.

McBreen, P. (2003). Questioning Extreme Programming. Pearson Education, Boston,
MA, USA.

Melnik, G., Maurer, F. (2002). Perceptions of agile practices: a student survey. Second XP Universe and First Agile Universe Conference on Extreme Programming and Agile Methods (XP/Agile Universe 2002),lk 241–250.

Merisalo-Rantanen, H., Tuunanen, T., Rossi, M. (2005). Is extreme programming just
old wine in new bottles: a comparison of two cases. Journal of Database Management 16 (4), lk 41–61.

Misra, S.C., Kumar, V., Kumar, U. (2009). Identifying some important success factors in adopting agile software development practices. The Journal of Systems and Software 82, lk 1869–1890.

Moe, N. B., Dingsøyr, T. and Dybå, T. (2010). A teamwork model for understanding an agile team: A case study of a Scrum project. Information and Software Technology, Volume 52 nr 5 (mai), lk 480-491.

Morgan, G. (2006). Images of Organizations, SAGE publications, Thousand Oaks, CA.

Nerur, S., Balijepally, V. (2007). Theoretical reflections on agile development methodologies – the traditional goal of optimization and control is making way for learning and innovation, Communications of the ACM 50 (3) lk 79–83.

Nerur, S., Mahapatra, R., Mangalaraj, G. (2005). Challenges of migrating to agile

methodologies, Communications of the ACM 48 (5), lk 72–78.

Nonaka, I., Takeuchi, H. (1995). The Knowledge-Creating Company: How Japanese Companies Create the Dynamics of Innovation, 12, Oxford University Press, New York.

O'Connor, C. P. (2010). Letters from the Edge of an Agile Transition. SPLASH '10 Proceedings of the ACM international conference companion on Object oriented programming systems languages and applications companion, New York.

Paetsch, F., Eberlein, A., Maurer, F. (2003). Requirements engineering and agile software development. 12th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises, IEEE, Linz, Austria.

Pikkarainen, M., Haikara, J., Salo, O., Abrahamsson, P., Still, J. (2008). The impact of agile practices on communication in software development. Empirical Software Engineering 13 (3), lk 303–337.

Power, K. (2010). Stakeholder Identification in Agile Software Product Development Organizations. Agile Conference 2010. IEEE Computer Society, Orlando.

Salas, E., Sims, D.E., Burke, C.S. (2005). Is there a "big five" in teamwork?, Small Group Research 36 (5), lk 555–599.

Sarapuu, H. (2010). Kasutatavus agiilses arendusprotsessis: case study. World Usability Day , 18. novembril, Tallinn.

Schatz, B., Abdelshafi, I. (2005). Primavera Gets Agile: A Successful Transition to Agile Development. IEEE Software, 22, (3), lk 36-42.

Sillitti, A., Succi, G. (2005). Requirements engineering for agile methods. Engineering and Managing Software Requirements. Wohlin, C., Aurum, A. (eds). Springer: New York, NY, lk 309–326.

Srinivasan, J., Lundqvist, K.(2010). Agile in India: challenges and lessons learned", ISEC'10, February 25-27, Mysuru, Karnataka, India.

Stephens, M., Rosenberg, D. (2003). Extreme Programming Refactored: The Case Against XP. Apress, Berkeley, CA.

Svensson, H., Höst, M. (2005). Introducing an agile process in a software maintenance and evolution organization. Proceedings of the 9th European Conference on Software Maintenance and Reengineering (CSMR 2005), lk 256–264.

Tessem, B. (2003). Experiences in learning xp practices: a qualitative study. Proceedings of the Fourth International Conference on Extreme Programming and Agile Processes in Software Engineering (XP 2004), lk 131–137.

Vinekar, V., Huntley, C. L. (2010). Agility versus Maturity: Is There Really a Trade-Off?. Computer, 43, (5), lk 87-89.

Lisa 1. Agiilsete metoodikate hindamise töövihik

 v.03

Evaluating Agile Methods Workboo

Key
blank: No rating
0: Can't do it
1: Unlikely to work
2: Difficult to do
3: Probably can do
4: Easy to do
5: Already do it

Individuals and Interactions over Processes and Tools		Considerations about					Summary		
Principles	Method: Practice	Culture	Customers	Projects	Tools & Processes	Staff	Fit	Strength	Notes
	Value Summary:	4,2	3,8	3,9	4	3,5	3,9	1	
Motivated individuals – Give them support and trust	**ASD:** Project stakeholders as independent agents	5	4	4	4	4	4,2	1	
	ASD: Adaptive (Leadership-Collaboration) Management Model	5	5	5	5	5	5	1	
	DSDM: 2) Teams must be empowered	5	4	4	4	4	4,2	1	
	XP: Collective ownership	5	4	4	4	4	4,2	1	
	FDD: Class (code) ownership	1	1	1	1	1	1	1	
	LD: Empower the team 14) Motivation	5	4	4	4	4	4,2	1	
	CC: Personal safety	5	4	4	4	4	4,2	1	
	AUP: Your staff knows what they're doing	5	4	4	4	4	4,2	1	
	Principle Summary:	4,5	3,8	3,8	3,8	3,8	3,9	1	
Self-organizing teams	**ASD:** Speculate: Project Initiation & Adaptive cycle planning	4	3	4	4	3	3,6	1	
	DSDM: 2) Teams must be empowered	4	3	4	4	3	3,6	1	
	XP: The planning game	4	3	4	4	4	3,8	1	
	FDD: Feature teams	3	3	3	3	3	3	1	
	LD: Empower the team 13) Self-determination	4	3	4	4	3	3,6	1	
	Scrum: Scrum teams	4	4	4	4	3	3,8	1	
	CC: Focus	4	3	4	4	3	3,6	1	
	Principle Summary:	3,9	3,1	3,9	3,9	3,1	3,6	1	
Face-to-Face Communi-cation	**XP:** Facilities Strategy	4	4	4	4	3	3,8	1	
	XP: Pair programming	4	4	4	4	3	3,8	1	
	XP: Whole team	4	3	4	4	4	3,8	1	
	XP: The planning game	4	4	4	4	4	4	1	
	CC: Osmotic communication	4	2	4	4	3	3,4	1	
	Scrum: Daily scrum meetings	4	4	4	4	4	4	1	
	Principle Summary:	4	3,5	4	4	3,5	3,8	1	
Sustainable pace	**XP:** Sustainable pace	5	5	4	5	4	4,6	1	
	Principle Summary:	5	5	4	5	4	4,6	1	
The unstated principle: Appropriate processes and tools	**FDD:** Configuration management	3	3	3	2	2	2,6	1	
	CC: Technical enviroment	4	4	4	4	4	4	1	
	XP: Continuous integration	4	4	4	4	4	4	1	
	LD: Amplify learning 5) Synchronization	4	4	4	3	4	3,8	1	
	LD: Deliver as fast as possible 10) Pull systems 11) Queuing theory 12) Cost of delay	4	3	4	3	3	3,4	1	
	LD: See the whole 21) Measurements	4	4	4	4	3	3,8	1	
	Principle Summary:	3,8	3,7	3,8	3,3	3,3	3,6	1	

 v.03

Evaluating Agile Methods Workboo

Key
blank: No rating
0: Can't do it
1: Unlikely to work
2: Difficult to do
3: Probably can do
4: Easy to do
5: Already do it

Working Software over Comprehensive Documentation

Principles	Method: Practice	Considerations about Cult	Customers	Projects	Tools & Processes	Staff	Summary Fit	Strength	Notes
	Value Summary:	3,9	3,9	3,5	3,4	3,6	3,7	1	
Early and continuous delivery	**ASD:** Adaptive life	4	4	3	3	4	3,6	1	
	DSDM: 5) Iterative and incremental development	4	4	4	4	4	4	1	
	XP: Small releases	4	4	4	4	4	4	1	
	FDD: Developing by feature	4	3	2	1	2	2,4	1	
	AUP: Deployment	4	4	3	3	3	3,4	1	
	CC: Frequent delivery	4	4	4	4	4	4	1	
	Scrum: Sprint	4	4	4	4	4	4	1	
	Principle Summary:	4	3,9	3,4	3,3	3,6	3,6	1	
Deliver working software frequently	**ASD:** Adaptive life cycle	4	4	3	3	4	3,6	1	
	DSDM: 3) Frequent Delivery	4	4	4	4	4	4	1	
	XP: Small releases	4	4	4	4	4	4	1	
	FDD: Regular build schedule	4	4	4	4	3	3,8	1	
	LD: Amplify learning 4) Iterations	4	4	4	4	4	4	1	
	Scrum: Sprint	4	4	4	4	4	4	1	
	Principle Summary:	4	4	3,8	3,8	3,8	3,9	1	
Working software is the primary measure of progress	**ASD:** Learn: Quality Review: Customer focus-group reviews	4	3	3	4	4	3,6	1	
	DSDM: 4) Fitness for business purpose	3	4	2	2	3	2,8	1	
	XP: Continuous integration	4	4	4	4	4	4	1	
	XP: Small releases	4	4	3	2	3	3,2	1	
	FDD: Reporting/ visibility of results	3	3	3	3	3	3	1	
	LD: Amplify learning 3) Feedback	3	4	2	2	3	2,8	1	
	AUP: Project Management Discipline	4	4	4	4	4	4	1	
	Scrum: Sprint Review	4	4	4	4	4	4	1	
	Principle Summary:	3,6	3,8	3,1	3,1	3,5	3,4	1	

 v.03

Evaluating Agile Methods Workboo

Key
blank: No rating
0: Can't do it
1: Unlikely to work
2: Difficult to do
3: Probably can do
4: Easy to do
5: Already do it

Customer Collaboration over Contract Negotiation

Principles	Method: Practice	Considerations about: Culture	Customers	Projects	Tools & Processes	Staff	Summary: Fit	Strength	Notes
	Value Summary:	4,8	4,1	4	4,1	3,9	4,2	1	
All stake-holders must work together daily	**ASD:** Project stakeholders as independent agents	5	4	4	4	4	4,2	1	
	ASD: Adaptive (Leadership-Collaboration) Management Model	5	5	5	5	5	5	1	
	DSDM: 1) Active user involvement	5	4	4	4	4	4,2	1	
	DSDM: 9) Collaborative and co-operative approach	5	4	4	4	4	4,2	1	
	XP: whole team	5	4	4	4	4	4,2	1	
	LD: Build integrity in 17) Perceived integrity 18) Conceptual integrity	4	4		3	3	3,5	0,8	
	LD: See the whole 22) Contracts	4		4		4	4	0,6	
	AUP: Test discipline		4	3	5	3	3,8	0,8	
	CC: Easy access to expert users	5	4	4	4	4	4,2	1	
	Scrum: Product backlog	5	4	4	4	4	4,2	1	
	Principle Summary:	4,8	4,1	4	4,1	3,9	4,2	1	

 v.03

Evaluating Agile Methods Workboo

Key
blank: No rating
0: Can't do it
1: Unlikely to work
2: Difficult to do
3: Probably can do
4: Easy to do
5: Already do it

Responding to Change over Following a Plan

Principles	Method: Practice	Considerations about Culture	Customers	Projects	Tools & Processes	Staff	Summary Fit	Strength	Notes
	Value Summary:	3,4	3,1	3,5	4	2,9	3,4	1	
Welcome changing require-ments	**ASD:** Adaptive life	4	3	3	4	3	3,4	1	
	DSDM: 6) All changes are reversible	4	4	4	4	3	3,8	1	
	DSDM: 7) Requirements are baselined at a high level	2	1	2	4	2	2,2	1	
	XP: System metaphor	4	4	4	4	4	4	1	
	XP: Design improvement (Refactoring)	4	4	4	4	3	3,8	1	
	FDD: Domain object modeling	3				2	2,5	0,4	
	LD: Decide as late as possible 7) Options thinking 8) Last responsible moment 9) Making decisions	3	3		4	3	3,3	0,8	
	LD: Build integrity in 19) Refactoring	4	4	4	4	3	3,8	1	
	AUP: Implementation	3	4	4	4	3	3,6	1	
	Scrum: Sprint planning meeting	3	1	3	4	3	2,8	1	
	Principle Summary:	3,4	3,1	3,5	4	2,9	3,4	1	

 v.03

Evaluating Agile Methods Workboo

Key
blank: No rating
0: Can't do it
1: Unlikely to work
2: Difficult to do
3: Probably can do
4: Easy to do
5: Already do it

The unstated Value: Keeping the Process Agile

Principles	Method: Practice	Considerations about: Culture	Customers	Projects	Tools & Processes	Staff	Summary: Fit	Strength	Notes
	Value Summary:	3,6	3,4	3,5	3,7	3,2	3,5	1	
Continuous attention to technical excellence and good design	**ASD:** Learn: Quality Review: Software Inspections	4	4	4	4	4	4	1	
	DSDM: 8) Testing throughout the lifecycle	4	4	4	3	3	3,6	1	
	XP: Test first	4	4	4	4	4	4	1	
	XP: Pair programming	4	4	4	4	4	4	1	
	XP: Coding standards	5	4	4	4	4	4,2	1	
	FDD: Inspections	4	4	4	4	4	4	1	
	LD: Amplify Learning 6) Set-Based Development	1	3	2	3	3	2,4	1	
	LD: Empower the team 15) Leadership 16) Expertise	4	4	4	4	4	4	1	
	LD: Build integrity in 20) Testing	4	4	4	3	3	3,6	1	
	AUP: implementation	3	3	3	3	3	3	1	
	Scrum: Scrum master	4	4	4	4	3	3,8	1	
	Principle Summary:	3,7	3,8	3,7	3,6	3,5	3,7	1	
Simplicity: maximize work not done	**XP:** Simple design	3	4	4	4	3	3,6	1	
	LD: Eliminate waste 1) Seeing waste 2) Value Stream Mapping	3	3	4	3	3	3,2	1	
	AUP: Simplicity	3	3	3	3	3	3	1	
	Principle Summary:	3	3,3	3,7	3,3	3	3,3	1	
Regular team retro-spectives	**ASD:** Learn: Quality Review: Postmortems	4	3	3	4	3	3,4	1	
	Scrum: Sprint retrospective	4	3	3	4	3	3,4	1	
	CC: Reflective improvement	4	3	3	4	3	3,4	1	
	Principle Summary:	4	3	3	4	3	3,4	1	

 v.03

Evaluating Agile Methods Workboo

Key
blank: No rating
0: Can't do it
1: Unlikely to work
2: Difficult to do
3: Probably can do
4: Easy to do
5: Already do it

Agile Manifesto Values	12 Principles	Considerations about: Culture	Customers	Projects	Tools & Processes	Staff	Summary: Fit	Strength	Notes
	All Values Summary:	4	3,7	3,7	3,8	3,5	3,7	1	
Individuals and interactions over processes and tools	Motivated individuals (support & trust)	4,5	3,8	3,8	3,8	3,8	3,9	1	
	Self-organizing teams	3,9	3,1	3,9	3,9	3,1	3,6	1	
	Face-to-face communication	4	3,5	4	4	3,5	3,8	1	
	Sustainable pace	5	5	4	5	4	4,6	1	
	The unstated principle: Approapriate processes and tools	3,8	3,7	3,8	3,3	3,3	3,6	1	
	Value Summary:	4,2	3,8	3,9	4	3,5	3,9	1	
Working software over compre-hensive documen-tation	Early and continuous delivery	4	3,9	3,4	3,3	3,6	3,6	1	
	Deliver working software frequently	4	4	3,8	3,8	3,8	3,9	1	
	Working software = progress	3,6	3,8	3,1	3,1	3,5	3,4	1	
	Value Summary:	3,9	3,9	3,5	3,4	3,6	3,7	1	
Customer collaboratio n over contract	Daily collaboration of all stakeholders	4,8	4,1	4	4,1	3,9	4,2	1	
	Value Summary:	4,8	4,1	4	4,1	3,9	4,2	1	
Responding to change over following a plan	Welcome changing requirements	3,4	3,1	3,5	4	2,9	3,4	1	
	Value Summary:	3,4	3,1	3,5	4	2,9	3,4	1	
The unstated value: Keeping the process agile	Technical excellence	3,7	3,8	3,7	3,6	3,5	3,7	1	
	Simplicity: maximize work not done	3	3,3	3,7	3,3	3	3,3	1	
	Regular team retrospectives	4	3	3	4	3	3,4	1	
	Value Summary:	3,6	3,4	3,5	3,7	3,2	3,5	1	

© Copyright 2005 ASK Process, Inc. v.03

Evaluating Agile Methods Workboo

Key
blank: No rating
0: Can't do it
1: Unlikely to work
2: Difficult to do
3: Probably can do
4: Easy to do
5: Already do it

Agile Methods	Considerations about: Culture	Customers	Projects	Tools & Processes	Staff	Summary: Fit	Strength	Notes
All Methods Summary:	**3,4**	**3,2**	**3,2**	**3,3**	**3,0**	**3,2**	**1,0**	
ASD Summary:	**4,4**	**3,8**	**3,7**	**4,0**	**3,9**	**4,0**	**1,0**	
DSDM Summary:	**4,0**	**3,6**	**3,6**	**3,8**	**3,5**	**3,7**	**1,0**	
XP Summary:	**4,2**	**3,9**	**4,0**	**4,1**	**3,8**	**4,0**	**1,0**	
FDD Summary:	**3,1**	**3,0**	**2,9**	**2,6**	**2,5**	**2,8**	**1,0**	
LD Summary:	**3,7**	**3,6**	**3,7**	**3,4**	**3,3**	**3,6**	**1,0**	
Scrum Summary:	**4,0**	**3,6**	**3,8**	**4,0**	**3,6**	**3,8**	**1,0**	
AUP Summary:	**3,7**	**3,7**	**3,4**	**3,7**	**3,3**	**3,6**	**1,0**	
CC Summary:	**4,0**	**3,3**	**3,7**	**4,0**	**3,6**	**3,7**	**1,0**	

 v.03

Evaluating Agile Methods Workboo

Key
blank: No rating
0: Can't do it
1: Unlikely to work
2: Difficult to do
3: Probably can do
4: Easy to do
5: Already do it

Adaptive Software Development (ASD)

Principles	Practices	Considerations about: Culture	Customers	Projects	Tools & Processes	Staff	Summary: Fit	Strength	Notes
	ASD Summary:	4,4	3,8	3,7	4	3,9	4	1	
Motivated individuals – Give them support and trust	ASD: Project stakeholders as independent agents	5	4	4	4	4	4,2	1	
	ASD: Adaptive (Leadership-Collaboration) Management Model	5	5	5	5	5	5	1	
Self-organizing teams	ASD: Speculate: Project Initiation & Adaptive cycle planning	4	3	4	4	3	3,6	1	
Early and continuous delivery	ASD: Adaptive life cycle	4	4	3	3	4	3,6	1	
Deliver working software frequently	ASD: Adaptive life cycle	4	4	3	3	4	3,6	1	
Working software is the primary measure of progress	ASD: Learn: Quality Review: Customer focus-group reviews	4	3	3	4	4	3,6	1	
All stake-holders must work together daily	ASD: Project stakeholders as independent agents	5	4	4	4	4	4,2	1	
	ASD: Adaptive (Leadership-Collaboration) Management Model	5	5	5	5	5	5	1	
Welcome changing require-ments	ASD: Adaptive life cycle	4	3	3	4	3	3,4	1	
Continuous attention to technical excellence and good design	ASD: Learn: Quality Review: Software Inspections	4	4	4	4	4	4	1	
Regular team retro-spectives	ASD: Learn: Quality Review: Postmortems	4	3	3	4	3	3,4	1	

 v.03

Evaluating Agile Methods Workboo

Key
blank: No rating
0: Can't do it
1: Unlikely to work
2: Difficult to do
3: Probably can do
4: Easy to do
5: Already do it

Dynamic Systems Development Method (DSDM)

Principles	Practices	Considerations about: Culture	Customers	Projects	Tools & Processes	Staff	Summary: Fit	Strength	Notes
DSDM Summary:		4	3,6	3,6	3,8	3,5	3,7	1	
Motivated individuals – Give them support and trust	DSDM: 2) Teams must be empowered	5	4	4	4	4	4,2	1	
Self-organizing teams	DSDM: 2) Teams must be empowered	4	3	4	4	3	3,6	1	
Early and continuous delivery	DSDM: 5) Iterative and incremental development	4	4	4	4	4	4	1	
Deliver working software frequently	DSDM: 3) Frequent Delivery	4	4	4	4	4	4	1	
Working software is the primary measure of progress	DSDM: 4) Fitness for business purpose	3	4	2	2	3	2,8	1	
All stake-holders must work together daily	DSDM: 1) Active user involvement	5	4	4	4	4	4,2	1	
	DSDM: 9) Collaborative and co-operative approach	5	4	4	4	4	4,2	1	
Welcome changing require-ments	DSDM: 6) All changes are reversible	4	4	4	4	3	3,8	1	
	DSDM: 7) Requirements are baselined at a high level	2	1	2	4	2	2,2	1	
Continuous attention to technical excellence and good design	DSDM: 8) Testing throughout the lifecycle	4	4	4	4	4	4	1	

 v.03

Evaluating Agile Methods Workboo

Key
blank: No rating
0: Can't do it
1: Unlikely to work
2: Difficult to do
3: Probably can do
4: Easy to do
5: Already do it

Extreme Programming (XP)

Principles	Practices	Considerations about: Culture	Customers	Projects	Tools & Processes	Staff	Summary: Fit	Strength	Notes
XP Summary:		4,2	3,9	4	4,1	3,8	4	1	
Motivated individuals – Give them support and trust	XP: Collective ownership	5	4	4	4	4	4,2	1	
Self-organizing teams	XP: The planning game	4	3	4	4	4	3,8	1	
Face-to-Face Communi-cation	XP: Facilities Strategy	4	4	4	4	3	3,8	1	
	XP: Pair programming	4	4	4	4	3	3,8	1	
	XP: Whole team	4	3	4	4	4	3,8	1	
	XP: The planning game	4	4	4	4	4	4	1	
Sustainable pace	XP: Sustainable pace	5	5	4	5	4	4,6	1	
The unstated principle: Appropriate processes and tools	XP: Continuous integration	4	4	4	4	4	4	1	
Early and continuous delivery	XP: Small releases	4	4	4	4	4	4	1	
Deliver working software frequently	XP: Small releases	4	4	4	4	4	4	1	
Working software is the primary measure of progress	XP: Continuous integration	4	4	4	4	4	4	1	
All stake-holders must work together daily	XP: whole team	5	4	4	4	4	4,2	1	
Welcome changing require-ments	XP: System metaphor	4	4	4	4	4	4	1	
	XP: Design improvement (Refactoring)	4	4	4	4	3	3,8	1	
Continuous attention to technical excellence and good	XP: Test first	4	4	4	4	4	4	1	
	XP: Coding standards	5	4	4	4	4	4,2	1	
Simplicity: maximize work not done	XP: Simple design	3	4	4	4	3	3,6	1	

 v.03

Evaluating Agile Methods Workboo

Key
blank: No rating
0: Can't do it
1: Unlikely to work
2: Difficult to do
3: Probably can do
4: Easy to do
5: Already do it

Feature-Driven Development (FDD)

Principles	Practices	Considerations about: Culture	Customers	Projects	Tools & Processes	Staff	Summary: Fit	Strength	Notes
	FDD Summary:	3,1	3	2,9	2,6	2,5	2,8	1	
Motivated individuals – Give them support and trust	FDD: Class (code) ownership	1	1	1	1	1	1	1	
Self-organizing teams	FDD: Feature teams	3	3	3	3	3	3	1	
The unstated principle: Appropriate processes and tools	FDD: Configuration management	3	3	3	2	2	2,6	1	
Early and continuous delivery	FDD: Developing by feature	4	3	2	1	2	2,4	1	
Deliver working software frequently	FDD: Regular build schedule	4	4	4	4	3	3,8	1	
Working software is the primary measure of progress	FDD: Reporting/ visibility of results	3	3	3	3	3	3	1	
Welcome changing require-ments	FDD: Domain object modeling	3				2	2,5	0,4	
Continuous attention to technical excellence and good design	FDD: Inspections	4	4	4	4	4	4	1	

 v.03

Evaluating Agile Methods Workboo

Key
blank: No rating
0: Can't do it
1: Unlikely to work
2: Difficult to do
3: Probably can do
4: Easy to do
5: Already do it

Lean Software Development (LD)

Principles	Practices	Considerations about: Culture	Customers	Projects	Tools & Processes	Staff	Summary: Fit	Strength	Notes
	LD Summary:	3,7	3,6	3,7	3,4	3,3	3,6	1	
Motivated individuals – Give them support and trust	LD: Empower the team 14) Motivation	5	4	4	4	4	4,2	1	
Self-organizing teams	LD: Empower the team 13) Self-determination	4	3	4	4	3	3,6	1	
The unstated principle: Appropriate processes and tools	LD: Amplify learning 5) Synchronization	4	4	4	3	4	3,8	1	
	LD: Deliver as fast as possible 10) Pull systems 11) Queuing theory 12) Cost of delay	4	3	4	3	3	3,4	1	
	LD: See the whole 21) Measurements	4	4	4	4	3	3,8	1	
Deliver working software frequently	LD: Amplify learning 4) Iterations	4	4	4	4	4	4	1	
Working software is the primary measure of progress	LD: Amplify learning 3) Feedback	3	4	2	2	3	2,8	1	
All stake-holders must work together daily	LD: Build integrity in 17) Perceived integrity 18) Conceptual integrity	4	4		3	3	3,5	0,8	
	LD: See the whole 22) Contracts	4		4		4	4	0,6	
Welcome changing require-ments	LD: Decide as late as possible 7) Options thinking 8) Last responsible moment 9) Making decisions	3	3		4	3	3,3	0,8	
	LD: Build integrity in 19) Refactoring	4	4	4	4	3	3,8	1	
Continuous attention to technical excellence and good design	LD: Amplify Learning 6) Set-Based Development	1	3	2	3	3	2,4	1	
	LD: Empower the team 15) Leadership 16) Expertise	4	4	4	4	4	4	1	
	LD: Build integrity in 20) Testing	4	4	4	3	3	3,6	1	
Simplicity: maximize work not done	LD: Eliminate waste 1) Seeing waste 2) Value Stream Mapping	3	3	4	3	3	3,2	1	

 v.03

Evaluating Agile Methods Workboo

Key
blank: No rating
0: Can't do it
1: Unlikely to work
2: Difficult to do
3: Probably can do
4: Easy to do
5: Already do it

Scrum

Principles	Practices	Considerations about: Culture	Customers	Projects	Tools & Processes	Staff	Summary: Fit	Strength	Notes
	Scrum Summary:	4	3,6	3,8	4	3,6	3,8	1	
Self-organizing teams	Scrum: Scrum teams	4	4	4	4	3	3,8	1	
Face-to-Face Communi-cation	Scrum: Daily scrum meetings	4	4	4	4	4	4	1	
Early and continuous delivery	Scrum: Sprint	4	4	4	4	4	4	1	
Deliver working software frequently	Scrum: Sprint	4	4	4	4	4	4	1	
Working software is the primary measure of progress	Scrum: Sprint Review	4	4	4	4	4	4	1	
All stake-holders must work together daily	Scrum: Product backlog	5	4	4	4	4	4,2	1	
Welcome changing require-ments	Scrum: Sprint planning meeting	3	1	3	4	3	2,8	1	
Regular team retrospect ives	Scrum: sprint retrospective	4	3	3	4	3	3,4	1	
Continuous attention to technical excellence and good design	Scrum: Scrum master	4	4	4	4	3	3,8	1	

 v.03

Evaluating Agile Methods Workboo

Key
blank: No rating
0: Can't do it
1: Unlikely to work
2: Difficult to do
3: Probably can do
4: Easy to do
5: Already do it

Agile Unified Process

Principles	Practices	Considerations about: Culture	Customers	Projects	Tools & Processes	Staff	Summary: Fit	Strength	Notes
	AUP Summary:	3,7	3,7	3,4	3,7	3,3	3,6	1	
Motivated individuals – Give them support and trust	AUP: Your staff knows what they're doing	5	4	4	4	4	4,2	1	
Early and continuous delivery	AUP: Deployment	4	4	3	3	3	3,4	1	
Working software is the primary measure of progress	AUP: Project Management Discipline	4	4	4	4	4	4	1	
All stake-holders must work together daily	AUP: Test discipline		4	3	5	3	3,8	0,8	
Welcome changing require-ments	AUP: Implementation	3	4	4	4	3	3,6	1	
Continuous attention to technical excellence and good design	AUP: implementation	3	3	3	3	3	3	1	
Simplicity: maximize work not done	AUP: Simplicity	3	3	3	3	3	3	1	

 v.03

Evaluating Agile Methods Workboo

Key
blank: No rating
0: Can't do it
1: Unlikely to work
2: Difficult to do
3: Probably can do
4: Easy to do
5: Already do it

Crystal Clear

Principles	Practices	Considerations about: Culture	Customers	Projects	Tools & Processes	Staff	Summary: Fit	Strength	Notes
	CC Summary:	4,2	3,7	3,8	4	3,7	3,9	1	
Motivated individuals – Give them support and trust	CC: Personal safety	5	4	4	4	4	4,2	1	
Self-organizing teams	CC: Focus	4	3	4	4	3	3,6	1	
The unstated principle: Appropriate processes	CC: Technical enviroment	4	4	4	4	4	4	1	
Face-to-Face Communi-cation	CC: Osmotic communication	4	4	4	4	4	4	1	
Early and continuous delivery	CC: Frequent delivery	4	4	4	4	4	4	1	
All stake-holders must work together daily	CC: Easy access to expert users	5	4	4	4	4	4,2	1	
Regular team retro-spectives	CC: Reflective improvement	4	3	3	4	3	3,4	1	

Lisa 2. Agiilsele arendusmetoodikale ülemineku riskianalüüs

Riski kirjeldus	Tagajärg	Riski tekkimise tõenäosus	Riski mõju projekti tulemusele	Riski tähtsus (kahe eelmise korrutis)	Vältimise võimalus	Tegevused, mida võtame ette, kui risk realiseerub?
Suured muudatused meeskonna tegutsemises korraga, meeskonnaliikmed hakkavad muudatusele vastu töötama.	Agiilne arendusmetoodika ei juurdu, projektid ebaõnnestuvad.	1	3	**3**	Edulugude tutvustamine, värbamispoliitika, vajadusel väliste konsultantide kaasamine. Metoodika muutust mitte ette võtta teiste suurte muudatustega ettevõttes samaaegselt.	Kaasame välised konsultandid, tegeleme koos meeskonnaga vastutöötamise põhjuste läbiarutamisega ja lahendamisega, leiame meeskonna seest "jutlustajad", kellele agiilsed põhimõtted lähedased on.
Meeskonnaliikmed ei taha võtta vastutust.	Agiilne arendusmetoodika ei juurdu, projektid ebaõnnestuvad.	1	3	**3**	Selgitada XP pakutavaid praktikaid, mis teevad arendajale vastutuse kandmise oluliselt kergemaks.	Selgitamisse panustamine, välise konsultandi kaasamine.
Meeskonnaliikmete liigne õhin.	Meeskonnaliikmed lähevad liigse õhinaga muudatusega kaasa ja probleemide tekkides suurendavad panust - ületöötamine, kiirustatult tehtud ja valed otsused	1	2	**2**	Selgitada meeskonnale antud probleemi tekkimise võimalust.	Probleemide kiire avastamine ja avameelne meeskonnas läbiarutamine. Eelduseks usalduslik õhkkond meeskonnas.
Meeskonnaliikmete vahel ei teki sünergiat, inimestevahelised vastuolud või koostöötahte puudumine.	Meeskond ei hakka tööle vajaliku efektiivsusega, agiilne arendusmetoodika ei saa täielikult	1	3	**3**	Meeskonnaliikmete valimine nende isikuomadusi ja sotsiaalseid oskusi arvestades, meeskonnas usaldusliku	Selgitamisse panustamine, välise konsultandi kaasamine. Võimalusel ja vajadusel meeskonna

	juurduda, projektid ebaõnnestuvad.				õhkkonna loomine.	koosseisu muutmine.
Agiilse metoodika juurutuse käigus esilekerkivate probleemide mittemärkamine ja nende eskaleerumine.	Agiilne arendusmetoodika ei juurdu, projektid ebaõnnestuvad.	2	3	**6**	Juurutusprotsessi juhil olla kursis võimalike tekkida võivate probleemidega ja nende sümptomitega, olla kursis meeskonna tööga. Meeskond ise on teadvustatud sellisest ohust ja jälgib samuti.	Probleemide esile kerkides lahendame need koos meeskonnaga.
Arendaja asub uue looga tegelema enne, kui eelmine valmis ja vigadeta integreeritud on.	Vead kuhjuvad, lahendamine on keerulisem ja võtab täiendavat aega.	2	2	**4**	Iseorganiseeruv meeskond jälgib üksteise tegevust ja kutsub varakult korrale.	Lahendatakse jooksvalt, sellest õppust võttes.
Arendaja eelistab kiiret häkki hea koodi põhimõtetele.	Koodi kvaliteet langeb ja hoolduskulud kasvavad.	1	1	**1**	XP paarisprogrammeerimise praktika aitab sellist probleemi vältida.	XP pidev koodi refaktoorimise praktika aitab varasemaid häkke elimineerida.
Juhtkond ei ole kaasatud.	Juhtkonna toetus võib jääda nõrgemaks.	1	2	**2**	Jälgida, et juhtkond oleks kaasatud, näiteks neile ülesandeid andes.	Selgitada agiilsete metoodikate eeliseid kasutades edulugusid.
Huvipoolte, sh juhtkonna jaoks, pole mõõdikud piisavad.	Tekib vastuseis agiilsele arendusele, nõutakse sellest loobumist.	1	3	**3**	Mõõdikute püstitamine selliselt, et rahuldaksid osapoolte vajadusi, kuid samas ei tekitaks meeskonnale väärtusetut lisatööd.	Tegevused samad, mis vältimisel, lisaks meenutame jällegi edulugusid, miks agiilne arendus hea on.
Puuduvad automaattestid või nende vähene hulk	Testimise ajakulu pidurdab projekti kulgu	2	2	**4**	Automaattestide kirjutamiseks moodultestide tasemel juurutatakse XP testipõhise arenduse praktika, läbivate protsesside tasemel kasutaja vaates kirjutab automaateste testija.	Läbivate protsesside tasemel kasutaja vaates kirjutab automaateste testija ja tema ressursi jälgitakse - kui vaja, abistab ülejäänud meeskond.

Lugude kvaliteet pole piisav, puuduvad aktsepteerimiskriteeriumid.	Arendajal pole selge, mida teha vaja on. Ajakulu täiendavatele täpsustustele või ümbertegemisele.	3	2	**6**	Tootejuhi koolitamine, koos meeskonnaga kogemustest õppimine.	Tootejuhi koolitamine, koos meeskonnaga kogemustest õppimine. Väliste konsultantide kaasamine.
Lugude hindamisel suured eksimused.	Planeerimine pole piisava täpsusega - kas ei jõua valitud lugusid valmis või tuleb pidevalt uusi lugusid lisaks nõutada, mis vähendab äritellija usaldust.	2	2	**4**	Hindamisel kasutada kogenud meeskonnaliikmete abi või algusfaasis olla valmis kaasama väliseid konsultante.	Kaasame välised konsultandid.
Möödalaskmised nõuete olulisuse hindamisel.	Arendatakse väheolulisi omadusi, projekti pakutav väärtus pole tasemel, liigne aja- ja rahakulu.	1	2	**2**	Meeskonnas lugude läbiarutamine, et kõik mõistaksid realiseerimiseks valitud lugude väärtust. Kuigi lõplik vastutus toote tööde nimekirja haldamise eest on tootejuhil, on meeskonnal õigus ja kohustus kaasa rääkida, kui peetakse vajalikuks.	Tegevused samad, mis vältimisel.
Nõuete ja kommentaaride segiminek.	Arendatakse väheolulisi omadusi, projekti pakutav väärtus pole tasemel, liigne aja- ja rahakulu.	1	2	**2**	Sellise ohu meeskonnale selgitamine ja protseduuri selgitamine, kuidas käib lugude iteratsiooni valimine.	Tegevused samad, mis vältimisel.
Tootejuhi vähene koolitus.	Tootejuhi roll projektis ei saa täidetud piisavalt kvaliteetselt (lugude kirjeldamine ja prioritiseerimine), projekt ei edene mõistliku kiirusega.	2	3	**6**	Tootejuhtide varajane ja põhjalik koolitamine.	Tootejuhtide koolitamine, kogenud meeskonnaliikmete tugi, välise konsultandi kaasamine.
Meeskonnaliikmete vähene koolitus, oskuste puudumine metoodikate rakendamiseks	Vähene efektiivsus.	1	3	**3**	Valitud arenduspartner on pikaajalise agiilse arenduse kogemusega, Elioni kaasatud arendajad saavad läbi XP paarisprogrammeerimise ja koodi ühisomandi praktikate tuge	Elioni arendajate täiendavad koolitused.

					oskuste omandamisel.	
Arendaja hirm oma puuduste osas (agiilsed metoodikad toovad oskusi ja nende puudusi selgemalt esile).	Arendaja moraal langeb, efektiivsus väheneb ning ta võib ka meeskonnast lahkuda.	2	3	**6**	Meeskonnasisese usaldusliku meeleolu loomine, läbi mille väheneb hirm ning võimaluste pakkumine oma oskuste parandamiseks. Tagasiside nelja silma all. Isiklik mentor uutele meeskonnaliikmetele. Paarisprogrammeerimisel paaride jagunemine nii, et uued liikmed on paaris kogenud liikmetega.	Arendajaga arutamine, mida ta vajab, et end kindlamalt tunneks (koolitus).
Agiilsete metoodikate suundumus, et arendaja peab olema meister mitmel alal.	Töötajaid raske leida või kallis ning aeganõudev koolitada.	1	2	**2**	Paarisprogrammeerimine aitab teadmistel meeskonnas levida. Meeskonnaliikmete julgustamine võtmaks uusi ülesandeid uutest valdkondadest, millega pole ise varem kokku puutunud. Spetsiifiliste rollide kasutamine, kui selleks on vajadus. Näiteks on meil analüütik eraldi rollina planeeritud, et arendaja ettevõttespetsiifilised teadmised saaksid olla nõrgemad.	Ootame teadmiste levimist ja julgustame. Pakume koolitust.
Suurenenud sõltuvus arendaja sotsiaalsetest oskustest.	Nõrgemate sotsiaalsete oskustega inimesed ei suuda panustada oma tavalisel tasemel, efektiivsus langeb ning võivad ka lahkuda.	2	3	**6**	Koolitusprogrammides tuua sisse meeskonna enda poolt pakutud teemad, mis aitavad arendada sotsiaalseid oskusi. Kasutada piisavat (kuid mitte liigset) dokumentatsiooni,	Värbame sobivaid inimesi.

					mis toetaks just arendusmeeskonna enda vajadusi.	
Arendajatel puuduvad teadmised ärivaldkondades.	Tellija kaotab arendusmeeskonna vastu usalduse.	2	2	**4**	Arendajate koolitamine vajalikes ärivaldkondades, värbamisel arvestada nii tehniliste kui äriliste teadmiste vajadusega. Analüütiku (kui Elioni äri hästi tundva rolli) kaasamine.	Arendajate koolitamine.
Arendajatel puuduvad teadmised ärivaldkondades.	Arendaja mõistab valesti vajadusi või teeb eeldusi valesti ja võtab neid edaspidises töös aluseks.	2	2	**4**	Arendajate koolitamine vajalikes ärivaldkondades, äritellijaga tiheda suhtlemise vajaduse rõhutamine ja selle võimaldamine. Analüütiku rolli kasutuselevõtmine.	Arendajate koolitamine. Tehtud vigade parandamine.
Meeskond ei näe agiilsete praktikate taga tegelikke agiilseid väärtusi ja põhimõtteid, mille saavutamise poole püüelda.	Meeskond ei tööta efektiivselt, puudub agiilsest arenduseset saadav kasu.	1	3	**3**	Pidev koolitusprotsess (sh seotud konverentsidel osalemine), agiilse arenduse väärtustele ja põhimõtetele keskenduv arendav juhtimine, treenimine.	Pidev koolitusprotsess (sh seotud konverentsidel osalemine), agiilse arenduse väärtustele ja põhimõtetele keskenduv arendav juhtimine, treenimine.
Meeskonnaliikmete motivatsioonipuudus uue metoodikaga kohanemiseks.	Meeskond ei tööta efektiivselt, puudub agiilsest arendusest saadav kasu.	1	3	**3**	Edulugude otsimine ja meeskonnas levitamine, et uue metoodika eelised selgeks saaksid ning meeskonna poolt omaks võetaks. Mitme meeskonnaliikme meelitamine agiilse arenduse mõtteviisi pooldajaks ja "jutlustajaks".	Edulugude otsimine ja meeskonnas levitamine, et uue metoodika eelised selgeks saaksid ning meeskonna poolt omaks võetaks. Mitme meeskonnaliikme meelitamine agiilse arenduse mõtteviisi pooldajaks ja "jutlustajaks".

Otsustusprotsesside muutumine juhi poolt otsustamiselt meeskonna poolt otsuste vastuvõtmiseks ei hakka tööle.	Otsustamatus, halvad otsused, projekti kontrolli alt väljumine.	2	3	**6**	Selge meeskondliku otsustuskorra väljatöötamine ja meeskonnaga kokkuleppimine, et vältida kaost. Demokraatlik hääletamine. Juht ei ole meeskonnatöös juhi rollis, vaid võrdväärne osaline. Neljasima vestluse aeg igal liikmel oma juhiga igal nädalal.	Sama, mis vältimisel, tuleb rohkem panustada.
Meeskonnaliikmete hindamisel agiilseks arenduseks vajalike omadustega mittearvestamine.	Hindamine ei toimu vajalikel alustel, tulemuslikkuse hinnang võib olla ebaõige, arengueesmärgid vääralt püstitatud.	1	3	**3**	Hindamine arvestab oskustega nii sügavuti kui ka erinevate oskuste hulgaga. Arvestada veel mentoriks olemise, vabatahtliku lisatööga ja teiste agiilses arenduses väärtust omavate tegevustega.	Muudame hindamise aluseid.
Ettevõtte värbamispõhimõtted ei arvesta agiilseks arenduseks vajalike isikuomaduste ja oskustega.	Tööle satuvad ebasobivad inimesed, kes ei ole tulemuslikud, liigne koolituskulu.	1	3	**3**	Värbamispõhimõtted üle vaadata, lisada praktikad, mis annaksid ülevaate agiilseks arenduseks vajalikest omadustest ja oskustest. Selleks sobib, kas praktiline töö koos meeskonnaga või sobiva praktilise ülesande täitmine.	Muudame värbamispõhimõtteid.
Vähene kommunikatsioon projektiliikmete vahel.	Projektides palju vigu, jäävad maha plaanidest.	1	3	**3**	Kasutatavad praktikad tagavad hea kommunikatsiooni projektiliikmete vahel.	Analüüsime põhjuseid ja elimineerime meeskonnaga koos tuvastatud probleemid.
Pideva integreerimise keskkonna loomine on keeruline, kui tegemist on erinevate platvormide ja süsteemidevaheliste sõltuvustega.	Suur ajakulu või ei saa üldse toimima.	1	3	**3**	Hetkel skoop ainult e-kanalitega seotud Java rakendused, ohud süsteemidevahelistе sõltuvustega seoses on minimeeritud SOA/EDA platvormi kasutusega.	Panustame rohkem, kaasame väliseid spetsialiste.

Agiilsel arendusel ei ole võrreldes monumentaalmetoodi kateg a arhitektuuri disain samavõrra fookuses.	See võib viia halbade süsteemidisaini otsusteni.	1	3	**3**	Hoida samaaegsete meeskondade arv väike ja selle suurenedes arvestada vajadusega rohkem panustada koordineerimisse. Üldine arhitektuuriplaan. XP paarisprogrammeer imise praktika.	Disainime süsteemi uuesti heaks ja panustame vältimise võimalustes kirjeldatud tegevustele.
Agiilsete arendusmeeskondade omavahelise koordineerimise ja kommunikeerimise korraldamine on keeruline	Projekti ajakavast mahajäämus, meeskondadevah elisest koordineerimatus est väheefektiivse töö tekkimine (erinevad meeskonnad lahendavad seda teadmata sama probleemi vms).	1	3	**3**	Hoida samaaegsete meeskondade arv väike ja selle suurenedes arvestada vajadusega rohkem panustada koordineerimisse. Üldine arhitektuuriplaan. XP paarisprogrammeer imise praktika.	Meeskondadega koos põhjuste läbiarutamine ja töötavate koostöömudelite leidmine.
Paarisprogrammeerim ist peetakse ebaefektiivseks ja ressurssi raiskavaks	Konfliktid tellijaga, nõutakse praktikast loobumist või konflikt meeskonnaga, programmeerija pole nõus praktikat omaks võtma.	1	3	**3**	Edulugude tutvustamine ja läbi selle eelarvamuse muutmine.	Edulugude ja uurimuste tutvustamine, teise poole seisukoha põhjuste uurimine ja vastuväidetega tegelemine.
Meeskonnaliikmed ei ole kõrge kvalifikatsiooniga.	Projekt ei edene planeeritud kiirusega, tulem on ebakvaliteetne.	1	3	**3**	Arenduspartneriks kaasame pikaajaliste kogemustega ja väljapaistvate oskustega töötajatega ettevõtte ning kasutatavate praktikate (ühine koodi omandus, paarisprogrammeer imine) saavutame oskuste ülekandumise oma töötajatele.	Ettevõtte arendajate täiendav koolitamine.
Meeskonna normaalne omadus on olla kõrge sidususega.	Meeskondadevahe line infoliikumine on pärsitud.	3	2	**6**	Kõiki osapooli kaasavad ärilised ja tehnilised infokoosolekud, kus räägitakse teemadest üle kõigi projektide.	Kui selgub, et koosolekud pole piisavad, siis arutame koos meeskondadega, mis infovahetust pärsib ja kuidas sellest vabaneda.

Arendajatele suurema mõjuvõimu andmine esmalt hirmutab juhte.	Juhtide ebakindlus, praktikate juurdumisele ebapiisav toetus, ebakindlus ja vead projektide ajaplaani koostamisel ja sellest kinnipidamisel.	2	2	**4**	Organisatsioonikultuur juba toetab spetsialistidele suurema mõjuvõimu andmist, lisaks juhtide koolitused.	Arutelud juhtidega.
Rakenduse implementeerimine algab varases projekti faasis, seega tulevad implementeerimisega seotud probleemid välja varakult.	Projekti juhtrühma hirm, et projekt on probleemne.	1	1	**1**	Selgitada agiilse arenduse põhimõtteid ja toimimise protsesse ning kirjeldada eeliseid.	Lisaks vältimise juures kirjeldatuga tegeleda hirmudega.
Tellija esindaja on pidevalt kaasatud ja peab oma panuse andma kogu projekti kestel, mis suurendab tema koormust võrreldes monumentaalmetoodikates kasutatud praktikatega ja suurendab stressi.	Tellija esindaja ei saa piisavas mahus osaleda või stressi tekkides ei suuda anda oma panust.	1	3	**3**	Ressursikokkulepped tellija esindaja juhiga, et projektis osalemiseks on aeg planeeritud ja broneeritud.	Arutelud ja kokkulepped tellija esindaja ja tema juhiga, kuidas tagada piisav kaasatus.
Paarisprogrammeerimine pole rakendatav, kui üks osapool on palju paremate oskustega kui teine.	Vähekogenud programmeerijad ei anna projektile väärtust, kuna peavad töötama paaris sama taseme programmeerijaga, teadmiste ülekandumist ei toimu, meie ettevõttes töötavad arendajad ei õpi süsteemi tundma, et projekti lõppedes ise väiksemate täiendustega hakkama saada.	1	3	**3**	Piisav koolitus väiksemate oskutega arendajate järeleaitamiseks.	Piisav koolitus väiksemate oskutega arendajate järeleaitamiseks.
Tellija pole rahul meeskonna tööga.	Koostöö mittetoimimine, oht agiilse lähenemise peatamisele.	1	3	**3**	Scrumi/XP kooslus annab tellijale rohkem võimalust projekti käiku mõjutada, pidev koostöö ja probleemide arutamine ning lahendamine.	Arutada koos meeskonnaga läbi, mis probleemid tellija vaates esinevad ning kokku leppida, kuidas neid koos lahendada.

Tellija poolse pühendumise puudumine, tellijal pole aega meeskonna jaoks.	Meeskond ei saa liikuda soovitud tempos, peab ootama tellija järel.	1	3	**3**	Ajaplaneerimine ja broneerimine sarnaselt arendusmeeskonda des kasutatavale metoodikale, ühine projektide prioritiseerimine nii, et töösolevad projektid on ka tellija vaates kõige olulisemad.	Meeskond selgitab koos tellijaga olukorra põhjuseid. Kui probleemi põhjuseks on projekti madal tähtsus tellija jaoks, kas siis on antud persoon õige tellija esindaja? Kui põhjuseks halb ajaplaneerimine, siis arutada koos tellija juhiga läbi võimalused edasiseks projekti kestuseks piisava ajaressursi broneerimiseks.
Otsuste tegemiseks läheb palju aega.	Projektid jäävad plaanist maha.	1	3	**3**	Iseorganiseeruvatel e meeskondadele anda piisavad volitused ise otsusi vastu võtta.	Meeskond selgitab olukorra põhjuseid, korrigeerib ja kui pühjused jäävad meeskonna mõjualast väljapoole, teeb ettepanekuid olukorra lahendamiseks.
Organisatsioonikultuuri sobimatus agiilseks arenduseks	Meeskond ei võta agiilset lähenemist omaks	1	3	**3**	Värbamisel arvestada agiilseks arenduseks vajalike omadustega.	Organisatsiooniku ltuuri muutmine terves ettevõttes võtab aastaid, kuid kuna arenduspartneri organisatsiooniku ltuur toetab agiilset arendust, siis Elioni töötajatega koos meeskonnas olles kandub see üle ka Elioni töötajatele.
Projekti kulgemise mõõtmine toimub valesti.	Ei saa aru projekti edenemise seisust, meeskonnaliikmet ele läbimõtlematult/v alesti määratud mõõdikud suunavad kõrvale kokkulepitud eesmärkide saavutamisest.	2	3	**6**	Läbimõeldud, meeskonnaga läbi arutatud, tegelikke eesmärke toetavad ja soovitavalt meeskondlikud mõõdikud.	Mõõdikute korrigeerimine.

Ei käivitu väljalasete planeeritud tempos valmimine.	Esimest väljalaset tuleb oodata plaanitust kauem, järgmised väljalasked hilinevad võrreldes plaaniga.	2	3	**6**	Plaan tuleb koostada reaalsetele võimaluste vastav, mitte nimetada unistust plaaniks. Regulaarsed meeskonna arutelud (sh igapäevased koosolekud ning tagasivaatekoosolekud), kus probleemide kerkimisel neist räägitakse.	Meeskonnaga koos arutada põhjuseid, objektiivsete põhjuste korral korrigeerida plaani, meeskona töös kitsakohtade leidmisel lahendada neid koos meeskonnaga ühiselt.
Ei saavutata dokumentatsiooni hulga tasakaalu.	Kas jääb puudu vajalikust infost (liiga vähe dokumentatsiooni) või arenduseks vajalikku aega raisatakse ebavajaliku (liigse) dokumentatsiooni kirjutamisele.	2	2	**4**	Selgitada välja osapooled, kellel ja milleks dokumentatsiooni vaja on ja kirjutada seda ainult tegelikeks vajadusteks vastavalt.	Vaadata üle dokumenteerimise kokkulepped ja neid korrigeerida.
Pideva integreerimisjärgse testimise käivitamine ei õnnestu.	Vead hakkavad kuhjuma.	2	3	**6**	Probleemi tekkimise võimaluse fookuses hoidmine. Meeskond jälgib oma liikmete tööd (iseorganiseeruv meeskond).	Vajadusel panustada rohkem, kaasata abijõude. Selle riski realiseerumist ei tohi mingil tingimusel lubada.
Mittefunktsionaalsed nõuded ei saa piisavalt tähelepanu.	Mittefunktsionaalsete nõuete mittearvestamine toob lisatööd ja tekitab segadust.	2	3	**6**	Jälgida mittefunktsionaalsete nõuete õigeaegset lisamist tööde nimekirja.	Arutada koosmeeskonnaga läbi, kuidas sellist probleemi edaspidi vältida saab. Hetkel üleval olevad probleemid lahendada.
Pikad viivitused tagasiside ahelas.	Arendaja ei saa kiiret tagasisidet realiseeritud loo kohta, kas see on aktsepteeritud.	2	3	**6**	Iteratsiooni lugude valimise eel vajalike tegevuste tegemine (analüüsid, sh kasutatavuse analüüs, fookusgruppide kasutamine), iteratsiooni planeerimine nii, et pooled lood on märgitud kui kindlasti realiseerimist vajavad ja teine pool on töid, mis	Leppia koos meeskonnaga (sh tellija esindaja) kokku protsess, kuidas tagada pika tagasiside ahela juures esmane aktsepteerimine ja kuidas edasisi muudatusi teostatud lugudes käsitleda (uute lugudena?) ja see juurutada.

					kiiremate lugude (eelnevate iteratsioonide lugude muudatuste/täienduste) ilmumisel võivad edasi lükkuda.	