TALLINNA PEDAGOOGIKAÜLIKOOL
Matemaatika-loodusteaduskond
Informaatika osakond

Sander Zeemann

Tarkvaraprojektide probleeme ja nende võimalikke lahendusi

Bakalaureusetöö

Juhendaja: prof. Peeter Normak

Autor:..... “.....”.....2003
Juhendaja:..... “.....”.....2003
Osakonna juhataja:..... “.....”.....2003

Tallinn 2003

“Ainus asi, mis on keerulisem
kui tuleviku ennustamine,
on mineviku muutmine”

Allikas teadmata

Sisukord

Sissejuhatus

Käesoleva töö eesmärk on anda ülevaade ja analüüsida tarkvaraprojektide põhilisi probleeme, nende olemust, tekkepõhjuseid ning samuti välja pakkuda võimalikke viise nende probleemide ennetamiseks ja kui võimalik, siis ka viise nende kõrvaldamiseks või mõju vähendamiseks.

Tarkvara parim omadus on selle paindlikkus - seda saab programmeerida tegema ükskõik mida. Tarkvara halvim omadus on samuti selle paindlikkus – omadus ükskõik mida on teinud tarkvaraarenduse raskesti planeeritavaks, jälgitavaks ja juhitavaks (Walker Royce) [1].

Üha enam sõltuvad kõikvõimalikud majandusharud tarkvarast. Tarkvara loojate jaoks on see muidugi väga meeldiv, kuna turg laieneb, kliente tuleb juurde, raha voolab sisse jne. Ettevõtete äriliste eesmärkide saavutamine on tarkvaraga nii tihedalt seotud, et uute süsteemide väljatöötamisel nõutakse üha paremat kvaliteeti, suuremat funktsionaalsust, paindlikkust, kasutajasõbralikkust, turvalisust – ja seda kõike aina lühenevate tähtaegade jooksul.

Tähtaegadest kinnipidamine on muutunud tõsiseks katsumuseks, levinud on isegi ütlus “Parem olgu see vigane, kui et kogu projekt viibib. Me saame selle alati hiljem ära parandada.” [4] Probleemi teravdab asjaolu, et kvalifitseeritud tarkvaraspetsialistide juurdekasv ja nõudlus ei ole tasakaalus; eriti suur puudus on veel laia silmaringiga ja märkimisväärsete kogemustega visiooniga inimestest. Tulemus on see, et tarkvara arendamine ja hooldamine on raske ja muutub järjest raskemaks. Kvaliteetse tarkvara arendamine ka järgnevate projektide puhul, samas suutes ette ennustada nii tööde mahtu kui ka vajaminevaid ressursse, on aga veelgi raskem.

Enne kui hakata uuele protsessile üle minema, tuleks kõigepealt mõista senist olukorda ja miks selline olukord üldse tekkida sai. üheksakümnendate keskel tehtud põhjalikumad analüüsid tarkvaraarenduse hetkeseisust jõudsid kõik samadele üldistele järeldustele:

- Tarkvaraprojektide edenemine on ikka veel väga ennustamatu – ainult umbes 10% projektidest valmib tähtajaks ning eelarve piirides.
- Erinevad juhtimise distsipliinid mängivad projekti õnnestumise ja ebaõnnestumise osas suuremat rolli kui tehnoloogia areng.
- Tarkvara ümbertegemiste (rework) maht ja praagi osakaal viitab ebaküpsele protsessile. [1]

Selline seis on valitsenud tarkvaraarenduse sektoris juba viimased kolm aastakümmet ja sellest räägitakse kui "tarkvaratootmise kriisist".

Tarkvarafirmad on hakanud lõpuks aru saama, et asi ei saa samamoodi enam edasi kesta. Statistikud näitavad meile numbreid, mida on lausa valus vaadata. 1998. aastal USA's läbi viidud uurimuse tulemusena selgus, et 175000 infotehnoloogiaprojektist, mille eelarve oli kokku 250 miljardit dollarit, täideti edukalt tähtajaks ja eelarve piirides vaid 26% [15]; paar aastat varem oli see näitaja veel väiksem.

Üks tuntumaid ja levinumaid tarkvaraprotsesside küpsuse hindamise mudeleid on Software Engineering Institute'i (SEI) poolt välja töötatud suutvusküpsuse mudel (Capability Maturity Model ehk CMM). Selle rakendamine on end enamikel juhtudel ka õigustanud, kuna asjaga on tegelenud kompetentsed ja pühendunud inimesed. Tarkvaraprotsessi arendamine on pikk ja kulukas protsess, mis nõuab pühendumist ka kõige kõrgemal organisatsiooni tasemel – juhtkonna täielik toetus on üliloluline. Nagu juba öeldud, protsessi muutmine võtab aega ja selle tegelike tulemuste nägemine ei pruugigi kohe välja paista. Et progressis veenduda, tuleb erinevaid näitajaid Protsessi muutus mõjutab inimesi ja organisatsiooni rohkem sügavuti, kui tehnoloogia või vahendite muutmine. Seda tuleb plaanida ja juhtida hoolikalt. Tuleb tuvastada muudatuste väljavaated ja oodatav kasu, teha see huvitatud osalistele selgeks, tõsta nende teadlikkuse taset ja siis järk-järgult liikuda seniselt tööpraktikalt uuele. Kui tarkvaraprotsessi arendamise tegevusi väga täpselt ei määratleta, siis ei ole mõtet ka soovitud tulemit loota.

Aastate jooksul on tarkvaraprojektide läbiviimisel ilmnenud probleemid, mis kuidagi ei taha kaduda, pigem korduvad ka järgnevate projektide käigus. Milles on probleem - kas juhtimises, kogemustes või hoopis force majeure? Olles uurinud mitmeid

tarkvaraprojektide alaseid raamatuid ja artikleid, võib väita, et suuremas osas on põhjus just selles, et ei õpita lihtsalt enda ning loomulikult ka teiste vigadest. Samuti ei panda piisavat rõhku arendusprotsessile, et seda täiustada, küpsemaks muuta ning väga tihti alahinnatakse juhtimise osakaalu projekti õnnestumisel.

Et antud tööd rohkem Eesti konteksti tuua, olen ma uurinud kodumaiste tarkavara tootmisega tegelevate ettevõtete probleeme tarkvaraprojektide osas ning kuidas nad üritavad nendega toime tulla. Kuna firmad reeglina oma tegelikest miinustest just võhivõõrale rääkida ei soovi, siis pidin adekvaatsemate andmete saamiseks ära kasutama tarkvaraarenduse vallas töötavaid sõpru/tuttavaid, kuid see eeldas vaikimisvande andmist (töös konkreetsete firmade mainimine ei ole lubatud). Uuringud põhinesid enamasti vabas vormis suulistel küsitlustel, kuid ka MSN Messenger oli täiesti omal kohal. Konkreetset küsimustikku koostatud ei olnud, kuigi kohati oli mingil määral aluseks võetud CMM'i teise taseme küsimustik. Eesmärgiks oli küsitletav vabas vormis rääkima saada andes ette teema ning seejärel esitades suunavaid ja täpsustavaid küsimusi. Eelnevalt olid paika pandud punktid, mida küsitletav peaks kommenteerima lähtuvalt oma praktikast firmas, kus ta praegu tegev on või eelnevalt on olnud. Kokku sai uuritud 14 erineva suurusega töötajaskonnaga tarkvara tootmisega tegeleva ettevõtte telgitaguseid. Väljatoodud probleemide osas ongi suurem rõhk just uuritud Eesti tarkvarafirmadel.

Suureks abiks töö koostamisel oli 2002.a. sügissemestril toimunud valikseminar "IT-arenduse aktuaalseid probleeme", mille käigus külastasime mitmeid tarkvara-arendusega tegelevaid ettevõtteid, et uurida, millega nad täpsemalt tegelevad, kuidas nad oma tööd teevad, mis probleeme neil ette tuleb, millised on nende tulevikuplaanid jne. Samuti olen palju kasulikku informatsiooni saanud 2003.a. kevadel Ahto Kalja loengus "Tarkvara projektijuhtimine", mille praktiline suunitlus, grupitööd ja põhinemine reaalsel olukorral lisasid tavapärasele teoreetilisele osale täiesti uue dimensiooni. Ka Marion Lepasaare poolt loetud "Tarkvaraprotsesside parandamine" oli väga heaks infoallikaks antud töö koostamisel. Töö koostamisel on toetutud materjalidele [1] – [15].

Töö struktuur

Töö on jaotatud nelja peatüki vahel. Esimene peatükk on pühendatud tarkvaraprotsessi parandamise ja hindamise mudelile Capability Maturity Model (CMM), tähtis osa selles on CMM'i teisel tasemel, mis keskendubki suures osas just projekti tasemele. Esimene peatükk tutvustab CMM'i üldist struktuuri, kirjeldab selle viite taset, samuti on iga taseme juures välja toodud nende võtmepiirkonnad (Key Process Areas). Antud peatükis on ka CMM'i analüüsitud ja välja toodud selle probleemid ja puudused. Suurema tähelepanu all on CMM'i teine tase, mille puhul on välja toodud ka iga võtmepiirkonna eesmärk.

Teine peatükk annab ülevaate tarkvaraprotsessi arenduse kasulikkusest organisatsiooni jaoks ning mida selle korrektne rakendamine organisatsioonis kaasa on toonud lähtuvalt; aluseks on võetud Software Engineering Institute'i poolt läbi viidud uurimused.

Kolmas peatükk käsitleb uuringu käigus selgunud Eesti ettevõtete tarkvaraprojektide levinumaid probleeme, nende tekkepõhjuseid, iseärasusi ning samas on iga probleemi juures välja toodud mitmeid viise nende ennetamiseks ja/või lahendamiseks. Kokku on välja toodud 13 probleemigruppi.

Viimases, neljandas peatükis käsitlen valikut Alan M. Davise poolt kirja pandud konventsionaalse tarkvaraarenduse põhiprintsiipidest, mis on täiesti aktuaalsed ka tänapäeval.

Lisade osas on muu hulgas ka välja toodud CMM'i teise taseme küsimustik iga võtmepiirkonna jaoks, projekti ajagraafiku näidis MS Projectis, projektiplaani soovituslik sisu, muudatusettepaneku näidis, online-intervjuu näidis ja osa uurimuse tulemustest koos esinemiste arvuga erinevate firmade lõikes.

I Tarkvaraprotsessi küpsuse hindamise mudel SW-CMM

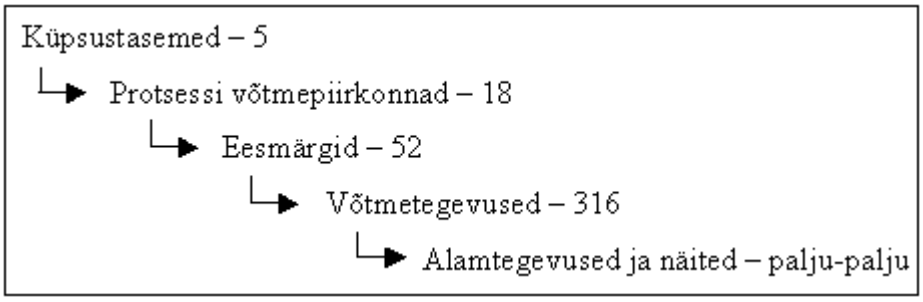
Tarkvaraprotsessi kvaliteedi hindamisel kasutatakse sageli Carnegie Mellon University Software Engineering Institute'i (CMU/SEI) poolt välja töötatud maailmatasemel tunnustatud SW-CMM mudelit. Käesolev peatükk keskendubki just SW-CMM'ile (versioon 1.1). CMM'ist ülevaate kirjutamisel on suures osas aluseks olnud Walker Royce'i 1999.a. kirjutatud raamat "Software Project Management: A Unified Framework" [1] ning "Capability Maturity Model for Software, Version 1.1" [2]. Kuigi Euroopas on levinud just SPICE ja ISO 900X standard, annan oma töös siiski ülevaate CMM'ist, mis on laialdaselt kasutusel maailma tähtsaimates tarkvara tootmisega tegelevates riikides (USA ja India) ning kuna põhimõtteliselt on ka CMM'i võimalik kohandada väikeste tarkvaraarendusfirmade jaoks - hea näide selles osas on näiteks Dynamic CMM.

I-I SW-CMM'i ajalugu

Novembris 1986 hakkas Software Engineering Institute (edaspidi SEI) koos Mitre Corporation'iga välja töötama tarkvaraprotsessi küpsuse hindamise ja arendamise raamsüsteemi, et aidata organisatsioonidel tõsta oma tarkvaraprotsessi taset. Asja vastu hakkas esimesena huvi tundma USA valitsus, et oleks mingi meetod, mille abil saaks hinnata nende jaoks mingit tarkvara välja töötavate erafirmade suutlikust.
Septembris 1987 avaldas SEI esimese versiooni protsessi küpsuse hindamise raamsüsteemist ja sellega koos kasutamiseks ka seda toetava küsimustiku .
Nelja järgneva aasta jooksul arendati seda raamsüsteemi edasi; suureks abiks oli erinevatelt organisatsioonidelt ja ka valitsuselt saadud vastukaja. Aastaks 1991 valmis CMM versioon 1.0. Pidev arendustöö CMM'i kallal jätkus ja selle tulemusena valmis 1993. aastal versioon 1.1 [2], millel põhineb ka antud peatükk.

I-II Ülevaade SW-CMM'ist

SEI poolt välja töötatud suutvusküpsuse mudel (Capability Maturity Model, edaspidi CMM) on levinud mall tarkvaraarendusprotsessi küpsuse hindamiseks. See on üks peamisi ja ka tuntumaid küpsusmudeleid firmade tarkvaraprotsessi hindamiseks ja arendamiseks.



Joonis 1: CMM'i üldine struktuur.

CMM defineerib arendusprotsessi küpsusele viis taset, mis näitavad, milliseid protsessi võtmepiirkondasid (Key Process Area, edaspidi KPA) vaadeldav organisatsioon toetab.

Tase 1. Algne tase – KPA'sid pole. Protsess on ebaküps, määratlemata, kui mingi protsess on isegi paigas, siis seda reeglina ei järgita.

Tase 2. Protsess on korratav. 2. taseme KPA'd: nõuete haldus, projekti planeerimine, tarkvaraprojekti jälgimine ja ülevaadete tegemine, alltöövõttude haldus, tarkvara kvaliteedikontroll, tarkvara konfiguratsioonihaldus.

Tase 3. Protsess on defineeritud. 3. taseme KPA'd: organisatsiooni fookus protsessil kui tervikul, organisatsioonipoolne protsessi defineerimine, koolitusprogrammid, terviklik tarkvaraarenduse juhtimine, tarkvaratoote arendus, gruppidevaheline koordineerimine, vastastikused läbivaatused.

Tase 4. Protsessi juhitakse. 4. taseme KPA'd: kvantitatiivne protsessi haldus, kvaliteedijuhtimine.

Tase 5. Protsessi optimeeritakse. 5. taseme KPA'd: tehnoloogia muutuste haldamine, protsessi muudatuste haldus, defektide ennetamine.

Enamik edukusele pretendeerivaid tarkvarafirmasid peaksid endale eesmärgiks seadma protsessi kolmanda CMM taseme. Kuigi leidub mitmeid vastuargumente, mis on tihti ka täiesti põhjendatud, et seades eesmärgiks mingi kindla CMM'i taseme

saavutamise, hakkad sa samas ka iseendale “hauda kaevama”, kuna arendusprotsessi läbiviimine on äärmiselt kulukas ja aeganõudev tegevus ning see ei pruugi end lõppkokkuvõttes ära tasuda, kuna eelnevalt polnud kõik veel piisavalt põhjalikult läbi mõeldud. Eriti käib see madalamatele CMM’i tasemetele kuuluvate organisatsioonide kohta, kes püüavad CMM’i ettekirjutusi üks-ühele järgida – see aga võib vabalt lõppeda pankrotiga. [3]

Arendusprotsessi suutlikkuse hindamise (Software Capability Evaluation, edaspidi SCE) käigus tavaliselt tuvastatakse, kas “organisatsioon ütleb, mida ta teeb ja teeb, mida ta ütleb”, uurides selleks organisatsiooni sisepoliitikat ja projektide praktikat. Neid hinnatakse KPA raamistiku järgi. Hindamisprotsess ei ole küll perfektne, aga siiski hea suhteline indikaator. [1]

Tüüpiline SCE kasutab SEI CMM’i küsimustikku (SEI Maturity Questionnaire) kui osa põhjalikust auditist. Küsimustikku kasutatakse reeglina enne hindamise juurde asumist konteksti tekitamiseks. CMM’i 2. taseme küsimustik on toodud välja ka Lisade osas. Hindamisprotsess sisaldab endas detailseid analüüse, intervjuusid ja teisi hindamise viise. [1]

Tarkvaraorganisatsioonide jaotuse kohta CMM’i tasemete järgi on tehtud palju erinevaid uuringuid. Tabel 1 annab umbkaudse pildi tarkvaraorganisatsioonidest 1995. aastal. Need on muidugi umbkaudsed arvud, kuna kõiki tarkvaraarendusega tegelevaid firmasid pole hinnatud. SEI andmete järgi (mai, 2001) hindamise läbinud firmadest 9% on 4. ja 5. tasemel (kokku 132; nendest umbes pooled paiknevad Indias), umbes 20% kuulub 3. tasemele.

CMM tase	**Sagedus**	**Käitumise tase**
1. Algne	70%	Ennustamatu, kõrge risk
2. Korratav	15%	Raskustes, aga jäävad ellu
3. Defineeritud	<10%	Stabiilne, ennustatav, arenev
4. Juhitav	<5%	Väga kindel, usaldusväärne
5. Optimeeritav	<1%	Kestvalt arenev

Tabel 1: Tarkvaraorganisatsioonide jaotus CMM-i skaalal aastal 1995. [1]

I-III CMM'i probleemid ja puudused

Üks peamisi SEI CMM'i puudusi on see, et KPA'd keskenduvad peamiselt konventsionaalses protsessis eksisteerivatele dokumendi kujul tehistele nagu disain, nõuded, projekti käigu jälgimise dokumendid, samuti ka allhangete lepingud, lepingud, plaanid ja raportid. Väga vähe KPA'sid puudutavad arenevaid väljatööte tehiseid nagu nõuete mudelid, disaini mudelid, lähtekood, käivitatavad programmid.

Teine CMM'i puudus on viis, kuidas see kirjeldab konfiguratsioonihaldust ja kvaliteedikontrolli, mitte teiste protsessi tegevuste juurde kuuluvatena, vaid kui kõikidest teistest protsessi tegevustest eraldiseisvaid distsipliine. [1]

I-IV Protsessi küpsus ja selle indikaatorid

Praktikas on protsessi küpsuse tegelikuks indikaatoriks projekti tulemuslikkuse ennustatavuse tase. CMM-i viie taseme vaheline korrelatsioon peaks näitama järgmisi trende:

Tase 1. Protsess on juhuslik ja käitub ennustamatult. Isegi kui mingi protsess on paigas, ei peeta sellest siiski eriti kinni. Probleemide lahendamine põhineb peamiselt "tulekahjude kustutamisel". Tähtaegadest kinni pidada on väga raske, aga kui seda siiski suudetakse, siis on tegu pigem hea juhusega ja meeskonnaliikmete kangelastegudega.

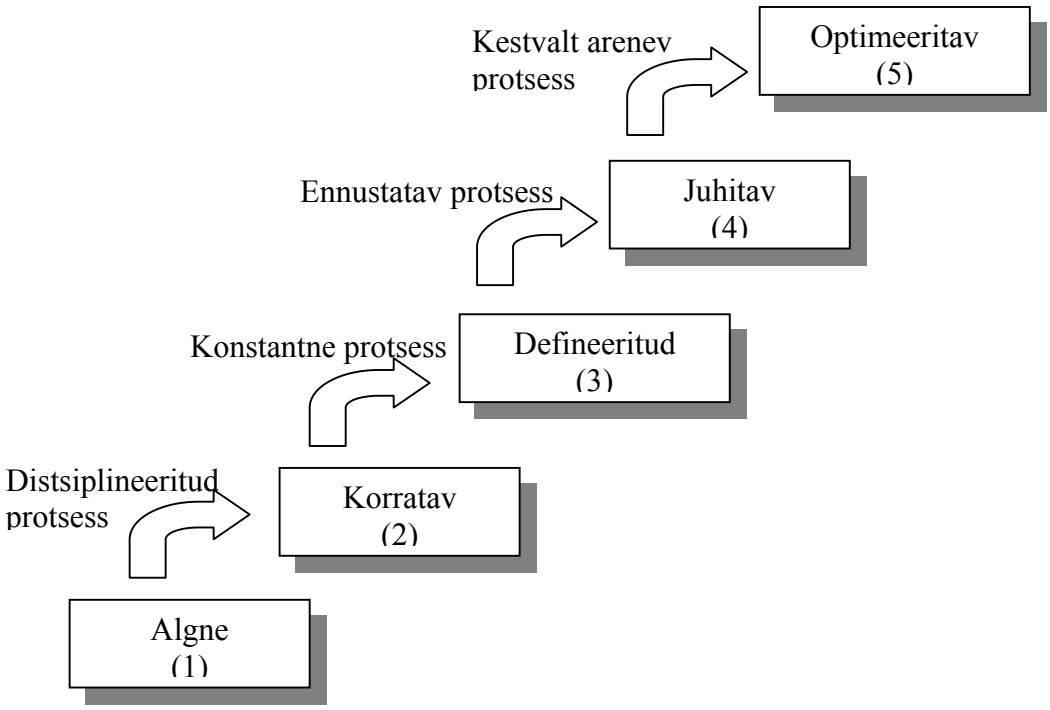
Tase 2. Protsess on erinevates projektides korratav. Sarnaste projektide tase on suhteliselt konstantne. Teisel tasemel on sõltuvus üksikisikust tunduvalt vähenenud.

Tase 3. Protsess on defineeritud, projektid käituvad aja möödudes hinna, ajagraafiku või tarkvarakvaliteedi suhtes vähehaaval järjest paremini. Kolmandal tasemel on organisatsiooni standardprotsess tarkvara arendamiseks kehtestatud ja dokumenteeritud.

Tase 4. Protsessi juhitakse. Aja möödudes projekti käitumine paraneb kas ühe näitaja suhtes oluliselt või mitme näitaja suhtes korraga (näiteks hinna ja tarkvara kvaliteedi suhtes). Välja on töötatud spetsiaalne meetrikasüsteem. Kõikumised tootlikkuses

ja/või tähtaegades on enamasti tühised. Neljanda taseme organisatsioon suudab tagada tarkvara kõrge kvaliteedi.

Tase 5. Protsessi optimeeritakse. Iga järgnev projekt käitub paremini igas mõttes. Viienda taseme organisatsioon täiustab pidevalt oma protsesse. [1][2]

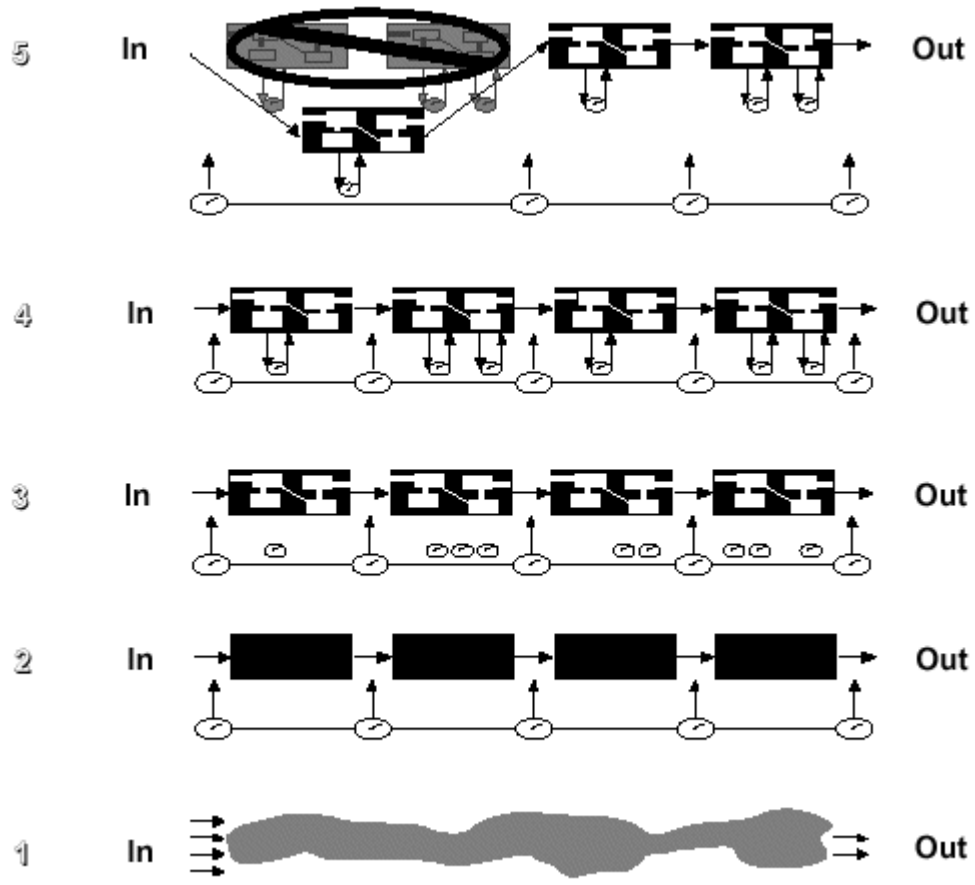


Joonis 2: CMM'i viis taset.

Mitmed organisatsioonid suudaksid kindlasti tekitada endale näo, mis võimaldab seda organisatsiooni hinnata CMM'i skaalal kolmanda taseme väärilisek, kuigi protsess tegelikkuses ei pruugi nii hea olla. Seevastu tõeliselt hea protsess peaks CMM'i kolmanda taseme hinde siiski kergesti saama. Omades tosinate protsesside hindamise kogemust, on Walker Royce identifitseerinud veel mõned indikaatorid, mis vastavad tõeliselt küpsele protsessile:

- Objektiivne arusaamine protsessi praegusest küpsusest.
- Objektiivne arusaamine projekti kvantitatiivsest käitumisest hinna ja kvaliteedi suhtes.
- Tegelik projekti tulemuslikkuse paranemine.
- Hindamise ettevalmistamiseks vaja minev aeg on minimaalne.

Tarkvaraarendajatel on väiksemate ja keskmise suurusega projektide puhul detailne ülevaade projekti hetkeseisust, suuremate projektide puhul piirdub see aga ainult nende enda vastutusvaldkonnaga antud projektis. Projektist väljaspool seisvatel gruppidel on aga väga raske saada pidevalt adekvaatset ülevaadet projekti seisust, kuna nende informatsioon põhineb perioodilistel ülevaatlustel.[2] Joonis 3 annab visuaalse ülevaate sellest, mil määral on juhtkonnal võimalik "näha tarkvaraprotsessi sisse" erinevatel CMM'i küpsustasemetel.



Joonis 3: Juhtkonna "nähtavus tarkvaraprotsessi sisse" erinevatel CMM'i tasemetel.

Piisavalt "küpse" tarkvaraprotsessiga organisatsioon ja projektid teavad protsessi ja järgivad seda. Nad ei vaja auditi ettevalmistamiseks aega. Kui te arvate, et teie organisatsioon on CMM'i skaalal vähemalt kolmandal tasemel, siis vastake sellele küsimusele: "Kas te taluksite üllatusauditit?".[1]

Vastavus mõne küpsusmudeli tasemele, näiteks CMM-ile, ei too tingimata kaasa kvaliteetsete toodete arendamist, küll aga tõeliselt kõrge kvaliteediga protsess, mille käigus valmivad kvaliteetsed tooted, hinnatakse küpseks. [1]

Vale arusaam on see, et küps protsess on kallis - küps protsess ei ole kallis - pikemas perspektiivis see hoopis säästab raha. Parandades ebaküpset protsessi, organisatsioon sageli tajub muutustega seotud kulutusi. Protsessi arendamine juba jooksval projektil, millel on lühiajalised kulukaalutlused, on väga raske. Samas on protsessi arendamine pikaajaliste äriliste püüdlustega organisatsioonis ja pikaajalistes projektides, mis on alles planeerimise staadiumis, kergem. [1]

I-V CMM'i 2. tase

Teisel tasemel on määratletud tarkvaraprojekti juhtimise poliitikad ja nende poliitikate elluviimise protseduurid. Uute projektide plaanimine ja juhtimine toetub kogemustele analoogsete projektidega. Protsessid on praktikas läbiproovitud, dokumenteeritud, ja kohustuslikud, on läbi viidud koolitused ning on loodud tingimused protsesside edasiseks täiustamiseks.

Teise taseme organisatsioonides läbiviidavatel projektidel on paika pandud põhilised juhtimise mehhanismid. Projektijuht jälgib projekti põhilisi näitajaid (kulud, ajagraafik, funktsionaalsus). Projektide planeerimisel kasutatakse eelnevatest projektidest saadud kogemusi ja andmeid. Probleemid identifitseeritakse, kui need esile kerkivad. Tarkvara nõuded ja nende rahuldamiseks toodetud tulemid salvestatakse alusdokumentidena ning kontrollitakse. Tarkvaraprojektide standardid on defineeritud ning organisatsioon tagab nende järgimise. Allhankijatega (kui neid on) luuakse tugev tellija-teostaja suhe.

Teise taseme organisatsioonide tarkvaraprotsessi võimekust saab kokkuvõtvalt defineerida kui distsiplineeritud, kuna projektide planeerimine ja jälgimine on

stabiilne ja eelnevate projektide edu on võimalik korrata. Projekti protsess on juhtimise mehhanismi efektiivse kontrolli all järgides realistlikke plaane. [2]

I-VI 2. taseme võtmepiirkonnad ja nende eesmärgid

Nõuete haldus

Nõuete halduse eesmärgiks on paika panna ühine arusaam projekti ja kliendi nõuete osas. Nõuete haldus hõlmab endas nii tehnilisi, kui ka mittetehnilisi nõudeid. Selle käigus koostatud dokumendid on aluseks hinnangute tegemisel; nõuete muutumisel sellega seonduvaid dokumente uuendatakse. [2]

Tarkvaraprojekti planeerimine

Eesmärgiks on paika panna mõistlikud plaanid nii arendustööde, kui ka projekti haldamiseks. See sisaldab endas hinnangute tegemist, kohustuse määratlemist ja tööde planeerimist. [2]

Tarkvaraprojekti jälgimine ja ülevaadete tegemine

Eesmärgiks on tagada piisav "nähtavus" projekti sisse, et juhtkond saaks kõrvalekallete esinemisel vastavad meetmed kasutusele võtta. Selle käigus jälgitakse projekti tegelikku käitumist ja võrreldakse seda plaanidega; erinevuste esinemisel plaane ajakohastatakse. [2]

Alltöövõttude haldus

Alltöövõttude eesmärgiks on valida kvalifitseeritud allhankija ning hallata neid efektiivselt, et tagada nii arendusprotsessi, kui ka toote vastamine kehtestatud kvaliteedinormidele. [2]

Tarkvara kvaliteedikontroll

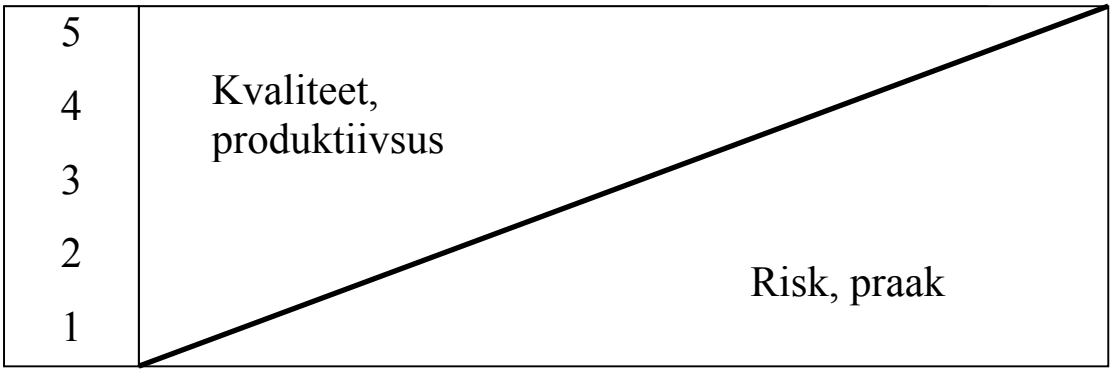
Eesmärgiks on tagada juhtkonna jaoks sobilik ülevaade projekti poolt kasutatavast protsessist, tehtavatest töödest ja valmistatavast tootest. See hõlmab endas läbivaatusi vms, et tagada protsessi, toote nõuetele vastavus. [2]

Tarkvara konfiguratsioonihaldus

Tarkvara konfiguratsioonihalduse eesmärgiks on paika panna ja tagada protsessi toimimine, mille alusel garanteeritakse terve projekti elutsükli käigus tekkivate tulemite täielikkus (puutumatus, rikkumatus - *integrity*). [2]

II Tarkvaraprotsessi arendamise tulemusi

Mida tähendab organisatsiooni jaoks küps protsess? Mis muutub, kui organisatsioon hakkab tegelema protsessi arendamisega? Nendele küsimustele annab põgusa vastuse joonis 3. Antud peatüki andmestik pärineb SEI poolt läbi viidud uurimusel. [5]



Joonis 4: Kvaliteedi ja produktiivsuse kasv ning riski ja praagi osakaalu vähenemine liikudes CMM'i madalamatelt tasemetelt kõrgemate suunas.

Kuigi eelnev joonis kajastab CMM'i järgimisel tekkivat kvaliteedi ja produktiivsuse kasvutrendi, on samad näitajad omased ka teiste SPI (Software Process Improvement) mudelite kasutamisele.

Küpse protsessi hind ei ole midagi kolossaalset, küps protsess ei maksa rohkem raha. Otse vastupidi, pikas perspektiivis see säästab raha. Küpse protsessi juurutamine juba käimasoleva projekti juures on äärmiselt keeruline ja ebaotstarbekas. Samas on protsessi juurutamine pikaajaliste äriliste püüdlustega organisatsioonis ja pikaajalistes projektides, mis on alles planeerimise staadiumis, kergem.

Küpset protsessi juurutades suureneb ka produktiivsus. Tarkvaratootjate puhul on võimalik seda mõõta mitmeti, näiteks kirjutatud koodiridade arv (SLOC) mingi ajaühiku kohta või varajases staadiumis avastatud defektide arv. Varajases staadiumis avastatud defektide parandamine on palju odavam, kui nende parandamine näiteks testimisfaasis või üldse peale toote valmimist ja kliendile üleandmist. Erinevate uurimuste ja hinnangute järgi küündib hinnavahe isegi mitme tuhande dollarini. Mida varasemas staadiumis defekt avastatakse, seda odavam tuleb ka selle parandamine.

Toote valmimise aeg väheneb. Kui SPI'd rakendada organisatsioonis, siis tavaliselt esimese aasta jooksul võib toote valmimise aeg olla isegi pikem, kuna protsess pole veel piisavalt omaks võetud või siis pole protsess piisavalt optimeeritud. Peale kahte-kolme aastat see aeg aga väheneb olulisel määral võrreldes ajaga, mis kulus enne SPI rakendamist. Sarnaste projektide puhul see aeg väheneb aastatega kuni teatud maani, mille järel muutub see enam-vähem konstantseks ehk on saavutatud optimaalne tase antud tingimuste juures.

Kvaliteet paraneb. Kvaliteet viitab tarkvara seisundile, kui see tellijale üle anti või turule toodi. Üks näitaja, mille abil kvaliteeti hinnata saab on defektide avastamine peale toote üleandmist/väljastamist.

Rentaablus. See tähendab põhimõtteliselt seda, mitu krooni teenitakse tagasi iga investeeritud krooni pealt. Sageli öeldakse selle kohta ka ROI (Return On Investment). Ühe uurimuse põhjal oli see näitaja 5, mis on väga hea tulemus, kuna on vähe investeerimisvõimalusi, mis pakuvad samasugust tulusust. Rentaablus on aga üks näitaja, mida on väga raske ennustada enne SPI rakendamist.

Tabel 1 toob välja ühe uurimuse tulemused, mis viidi läbi SEI poolt 13 firma seas, mis olid rakendanud erinevaid SPI metoodikaid. Antud firmad ei kuulu ainult CMM'i 4. ja 5. tasemele, vaid nende hulgas on ka madalamate tasemete esindajaid. Peale protsessi parendamise alustamist on olulisel määral paranenud järgmised näitajad: produktiivsus, defektide avastamine varases staadiumis, toote valmimise aeg ja kvaliteet.

Kategooria	**Hinnavahemik**	**Mediaan**
SPI rakendamise maksumus aastas	$49000 - $1202000	$245000
SPI'ga tegeldud aastate arv	1a.-9a.	3,5a.
SPI maksumus ühe inimese kohta	$490 - $2004	$1375
Produktiivsuse kasv aastas	9% - 67%	35%
Varases staadiumis avastatud defektide arv	6% - 25%	22%
Toote valmimise aja vähenemine aastas	15% - 23%	19%
Peale toote valmimist ilmnenud defektide arv	10% - 94%	39%

Rentaablus	$4,0 – $8,8	$5,0

Tabel 2: Kokkuvõte SPI rakendamise tulemustest. [5]

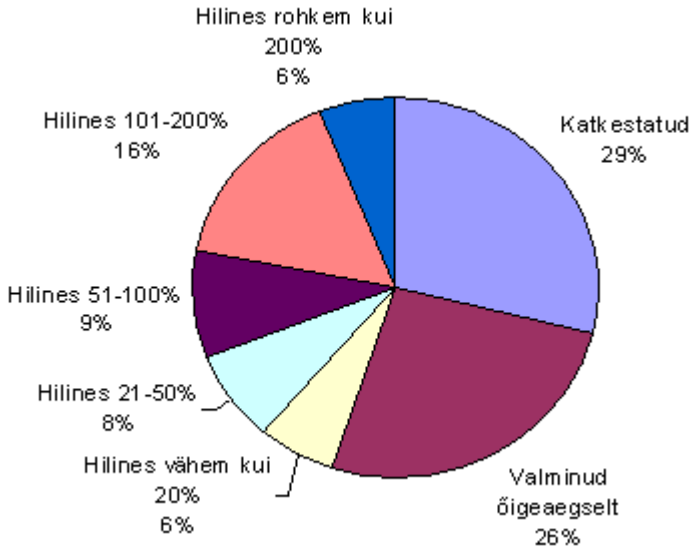
SEI on veel välja toonud rea positiivseid näitajaid, mida SPI rakendamine organisatsioonis endaga kaasa toob:

- Kodeerimisele ja testimisele kuluv aeg lüheneb.
- Toetudes olemasolevale faktide ja muu info kogumile, on võimalik projektide õnnestumist ette ennustada.
- Meeskonna vaim, uhkus ja moraal on tõusnud.
- Uut tehnoloogiat on lihtsam kasutusele võtta.
- Töötingimused on paranenud: vähem ületunde, stressi ja närvesöövaid probleeme.
- Kriisiolukordi tuleb väga harva ette.

III Tarkvaraprojektide probleeme ja nõuandeid nende lahendamiseks

Tarkvaraprojektidel esineb igasuguseid probleeme, alustades tüüpilisest ajagraafiku edasi nihkumisest, kuni väga abstraktsete ja tehnilisteni välja. Antud peatükis keskendun uuringu (uuringu täpsemaid andmeid on toodud Lisade osas, vaata Lisa 7; online-intervjuu näidis, vaata Lisa 6) käigus just kõige enam esile kerkinud üldistele probleemidele, mis uuritud firmade tarkvaraprojektides (mingi osa on iseloomulik ka teiste valdkondade projektidele) ikka ja jälle esile on kerkinud. Samuti püüan iga probleemi puhul välja tuua selle põhjuse(d) ning välja pakkuda ka võimalikke lahendusvariante. Kuna mõnede probleemide allikad ühtivad ja/või need on omavahel sõltuvuses, siis võib esineda ka lahenduste kattuvusi.

Probleemide lahendamine on tarkvaraprojektide juures üks vältimatu tegevus, mis lõppkokkuvõttes võib projekti kulgu muuta, sellest tulenevalt on kasulikum siiski püüda probleeme ennetada ning kui võimalik, siis ka vältida analüüsides ja arendades organisatsiooni tarkvaraprotsessi. Joonis 4 esitab Standish Groupi'i poolt 1999. aastal läbiviidud uurimuse tulemused tarkvaraprojektide tulemite osas. [14]



Joonis 5: Tüüpiline tarkvaraprojektide tulem. Allikas: Standish Group'i uurimus, 1999. [14]

III-I Hinnangud ajagraafiku, töömäärade, maksumuse ja tarkvara suuruse osas

Tüüpilised hinnangud, mis projekti algul tehakse on ajagraafiku, töömäära, maksumuse ja loodava tarkvara suuruse osas. Suurim viga, mis nende juures võidakse teha on see, et neid hinnanguid võetakse nö puhta kullana. Projekti algfaasis tehtud hinnangud on siiski eelhinnangud ning neid ei tohiks vaadata kui absoluutset tõde. Seda just sellepärast, et algfaasis tehtud hinnangud on väga ebatäpsed, kuna loodavast tarkvarast puudub veel selge ettekujutus. Seepärast pole kasulik paika panna kindlaid numbreid/tähtaegu, vaid pigem mingi vahemik, kui suur see vahemik peaks olema, sõltub juba antud projekti iseärasustest, kogemustest, intuitsioonist jne. Järgnevalt toon välja tüüpilisi vigu, mida tehakse ajagraafiku, töömäärade, maksumuse ja tarkvara suuruse hindamisel.

Ajagraafiku hindamise probleeme:

- Ülesanded pole piisavalt defineeritud: puudub ettekujutus tehtavatest töödest, nende sõltuvustest, kestvusest jne., järelikult ei saa ka adekvaatset graafikut koostada.
- Aega on vähe: tähtajad on väga lühikesed ja lihtsalt pole aega, et täpselt kõike paika panna – tulemuseks on äärmiselt pealiskaudne ajagraafik, millest tööde teostamisel ja jälgimisel erilist abi ei ole.
- Võimete ülehindamine: graafiku koostaja on liiga optimistlik või tal puuduvad andmed projekti täitjate kohta, nende teadmiste, kogemuste kohta.
- Ajagraafik pole piisavalt tükeldatud: isegi kui plaanis on tööd paigas, siis graafikus ei näidata ära nende sõltuvusi, järjestatust jne – ei teki piisavalt head ettekujutust projekti etappide sisust.
- Ei arvestata mittetootliku ja projektiväliste töödega: eeldatakse (arvatakse, loodetakse), et tööl viibides tegeleb inimene koguaeg tootliku tööga.

Töömäära hindamise probleeme:

- Selge ettekujutuse puudumine loodavast tarkvarast: visioon on puudulik ja/või spetsifikatsioon on ebatäpne.

- Töömäärahinnang sobitatakse ajagraafikusse: kui graafik on eelnevalt koostatud või kui on projektil kindel tähtaeg, siis ei jää töömäärade hindajal muud üle, kui tehtud hinnanguid kohandada, et vähemalt paberil asi toimiks.
- Vale inimene hindab või hindab ainult üks inimene: hinnangute andmisel ei arvestata töö teostajatega, ei konsulteerita nendega. Antakse hinnang, mis võib põhineda ainult oletustel.
- Ajalooandmestiku puudumine: ei saa ära kasutada eelnevate projektide näitajaid töömäärade osas, kuna need lihtsalt ei eksisteeri.
- Ei võeta arvesse mittetootlikke ja projektiväliseid töid.

Projekti maksumuse hindamise probleeme:

- Ajalooandmestiku puudumine või selle ebamäärasus: puuduvad konkreetsed andmed eelmiste projektide kohta, mis hõlbustaks hinnangute tegemist.
- Number võetakse "laest": ei baseeru reaalsusel.
- Töömäärad ja ajagraafik pole eelnevalt paika pandud: kuna pole teada projekti kestvust, ülesannete sõltuvust, ega töömäärade hinnanguid, siis ei saa ka adekvaatset eelarvet koostada.
- Ei arvestata kõikide kululiikidega: (tarkvara)projektide kulud ei piirdu ainult igakuiste püsikuludega.
- Soovitakse võimalikult palju vahelt teenida: kui palju eelarvele juurde liita, et kliendil ei tekiks "nöörimise" tunnet.

Loodava tarkvara suuruse hindamise probleeme:

- Pealiskaudne spetsifikatsioon: ettekujutus loodavast tarkvarast on lünklik.
- Ei kasutata vastavat tarkvara: ilma hinnangu andmist toetava tarkvarata võib hinnang olla väga ebatäpne.
- Ülevaade võimalikest korduvkasutatavatest komponentidest puudub: analüüs, spetsifikatsioon puudulik.
- Puuduvad oskused, kogemused.
- Tegu on väga suure projektiga: aastaid kestev tarkvaraprojekt, mille suurust on raske hinnata juba ainuüksi selle unikaalsuse tõttu.

Nõuandeid ajagraafiku osas:
Ajagraafiku koostamisel peaks kindlasti aluseks olema töömäärade hinnang. Projekti alguses on mõistlik kasutada top-down meetodit ajagraafiku koostamiseks [7]. Järgnevalt toongi välja ühe võimaliku variandi [6]:
Kõigepealt jaga projekt osaprojektideks ja see veel omakorda loogilisteks lõikudeks. Järgnevalt jaga need lõigud ülesanneteks ja osaülesanneteks ning seejärel jaota ülesanded inimeste vahel ära (igal ülesandel peab olema üks täitja; kui täitjaid on mitu, siis üks nendest peab olema vastutaja). Hindamisel kasuta kindlasti teostajate hinnanguid ja/või kolmandaid isikuid, samuti on hindamisel suureks kasuks ajalooteabe olemasolu. Järgnevalt tuleks paika panna ülesannete järgnevused ja sõltuvused. Graafiku koostamisel tuleb jälgida, et mõningaid inimesi üle ei koormataks. Selles osas on suureks abiks projektijuhtimistarkvara, näiteks Microsoft Project. Lisade osas on välja toodud MS Projectiga koostatud ajagraafiku näidis.

Kuidas toimida, kui projekt (kogu projekt, osaprojekt) hilineb? Selle probleemi lahendamiseks on palju võimalusi, ka väga loovaid. Kõigepealt tuleks välja uurida hilinemise põhjus, ja seda mitte selleks, et süüdlasi karmilt karistada, vaid ikka selleks, et leida sobivaim lahendus ning kogemustest õppida.

Kui hilinemise põhjuseks on klient (näiteks pidevalt uute muudatusettepanekute esitamine või nende esitamine projekti lõppfaasis), siis ei tohiks klient ka nuriseda, kui projekti ei lõpetata algul kokku lepitud tähtajaks, kuid see põhineb siiski vastastikkusel kokkuleppel. Muudatuste juhtimisest on juttu lk 34.

Kui projekti hilinemine on projektijuhi või mõne muu projektiliikme süü, siis tuleb asjale teise nurga alt läheneda. Oleks vale arvata, et aeg, mis antud etapis üle läks tehakse järgnevate etappide jooksul tasa. Maailmapraktika siiski näitab, et seda ei juhtu, hoopis vastupidi – projekt hilineb veelgi rohkem. Kui loodavale tarkvarale esitatavad nõuded on grupeeritud (kriitiline, oluline, soovituslik...), siis on kõige tähtsam siiski saavutada kriitiliste nõuete täitmine ja alles seejärel hakata teisi nõudeid rahuldama. Kui projekti ajagraafik ja teostajad on mõistlikud, siis suudetakse tagada kriitiliste ja ka oluliste nõuete saavutamine õigeks tähtajaks, ilma et peaks tähtaega edasi lükkama [6][7] ja seega kliendile otsest kahju tegema ning oma firma reputatsiooniga riskima.

Kui kõikide nõuete täitmine on ühtmoodi oluline, siis tuleb ajagraafikut muuta pikemaks (loomulikult tuleb sellest ka klienti teavitada). Graafiku pikendamisel ei tohiks lisada lõppu vaid hetkel üle läinud päevade arvu, vaid iga järgneva etapi lõppu tuleks lisada üle mindud päevade suhtarv iga etapi pikkusega võrrelduna [7]. Juhul kui tegu oli spetsiifilise probleemiga, mis järgnevates etappides kindlasti ei saa ette tulla, siis pole iga etapi pikendamine esialgu otstarbekas. Ajagraafikut tuleb täpsustada projekti käigus, kuna iga päevaga ilmneb projekti osas uut informatsiooni, mis seda mõjutada võib. Ajagraafik tuleks üle vaadata periooditi ja ka toimunud sündmustest lähtuvalt ning siis vastavad muudatused sisse viia [1][6][7].

Järgnevalt toon välja mitmeid viise, kuidas projekt tagasi graafikusse saada:

- Töötundide arvu suurendamine: negatiivne külg asja juures on see, et töötajate produktiivsus langeb ja kulud suurenevad.
- Ressursside lisamine, ka allhankijate kaasamine: suurendab jälle kulusid.
- Funktsionaalsuse vähendamine või selle realiseerimine hiljem: klient ei pruugi kohe saada tarkvara, mis teda 100% rahuldab.
- Üritada töötajate motivatsiooni tõsta lubadustega: katmata lubadusi ei maksa siiski jagada. Kogenud tegijate jaoks ei pruugi raha olla enam piisav motivaator.
- Muudatuste sisseviimine peatatakse kuni esialgu ettenähtud tööd on valmis: kui muudatus on kliendile piisavalt tähtis, siis ei tagata kliendi ärivajaduste rahuldamist.

Nõuandeid töömäärade osas:

Mida varem teha töömäärade hinnang, seda tõenäolisem on, et see on ebareaalne. Hinnangu andmisel tuleb arvestada mitmete näitajatega: kas kasutatakse uusi tehnoloogiad, meetodeid, kas arenduskeskkond on äkki uus või inimesed? Mida täpsem on spetsifikatsioon, seda täpsem tuleb ka töömäärade hinnang. Ka töömäärade jaoks hinnangute tegemisel on abiks vastav tarkvara. Sisestades nõutavad andmed arvutisse, arvutab programm välja umbkaudse töömäära. Internetis saadaval on ka vastava tarkvara vabavara versioonid. Kui võimalik, siis tuleks hinnangut küsida ka töö sooritaja enda käest.

Töömäärade hindamiseks veel mitmeid võimalusi [6]:

- Matemaatilised mudelid: Function Point Analysis (FPA), Constructive Cost Model (COCOMO).
- Eksperdi hinnang: üks või mitu eksperti annavad oma sõltumatu hinnangu.
- Analoogia: antud juhul võetakse aluseks eelnevad samatüübilised projektid.
- Parkinsoni seadus: "Töö täidab alati temale varutud aja".

Tegureid töömäärade hindamiseks [6]:

- Realiseeritud funktsioonide arv mingis ajaühikus sõltuvalt raskusastmest.
- Ekraanivormide arv.
- Kasulikule tööle töökohas kulunud aeg.
- Meeskonna suurus.
- Keskmine tööpanus ühe klassi kohta.
- Projektiliikme kogemus antud valdkonnas.

Nõuandeid maksumuse osas:

Projekti maksumuse hindamisel tuleb jällegi aluseks võtta töömäärade hinnang (rahas seekord). Samuti tuleb maksumuse sisse arvata maksud, keskkonna püsikulud, esinduskulud, koolitus, võimalik ületöö eritasu, allhanked, seadmete ja/või tarkvara hankimine ning ei maksa ka reservi ja kasumit unustada. Üks võimalus projekti eelarve koostamiseks on võtta aluseks varasemad analoogsed projektid [6], samas arvestades inflatsiooniga.

Kui projekt on töös ning selle hind on fikseeritud ja finantsid on otsakorral, siis tuleb võtta kasutusele reserv (kui see eksisteerib), kärpida vähemtähtsate kuluartiklite eelarvet või kasumi osa, viimane variant oleks küsida kliendilt raha juurde, mis aga ei tule jällegi firma mainele kasuks.

Nõuandeid tarkvara suuruse hindamise osas:

Tarkvara suuruse hindamise aluseks on valmis spetsifikatsioon. Enne kui spetsifikatsioon pole valmis, on hinnang väga ebatäpne. Hinnangu andmise teevad lihtsamaks korduvkasutatavate komponentide kasutamine, eelnevad kogemused

analoogsete projektidega, ajalooandmestiku olemasolu, spetsiaalne tarkvara (näiteks Construx Estimate).

Capers Jones'i 10 pöidlareeglit hinnangute tegemiseks [9]

Suure nõudluse tulemusena pakkus Capers Jones välja 10 nö pöidlareeglit hinnangute tegemiseks. Nendega saavutatavad näitarvud ei ole väga täpsed, vastupidi – tulemused võivad erineda reaalsusest päris palju. Kuna nende kasutamisel täpseid tulemusi ei saa, siis ei maksa neid tulemusi ka otseselt aluseks võtta, seda näiteks pakkumiste tegemisel.

Ta võrdles 50 *käsitsi* tehtud hinnangut 50 spetsiaalse tarkvara poolt genereeritud hinnanguga. Tulemustest selgus, et 75% juhtudest olid *käsitsi* tehtud hinnangud valed, täpsemalt liiga optimistlikud. Tarkvara poolt genereeritud hinnangud olid aga seevastu oluliselt täpsemad ja konservatiivsemad, ehk nad ennustasid pisut suuremaid ajagraafikuid ja kulusid.

Järgnevalt toon välja ülevaate Capers Jones'i poolt välja pakutud nö pöidlareeglitest hinnangute tegemiseks:

1. 1 funktsioonpunkt (function point) = 100 loogilist koodirida. Tegu on keskmise suurusega, kuna erinevates programmeerimiskeeltes on see suurus erinev.
2. Võttes funktsioonpunktide arvu astmesse 1,15, saame umbkaudse lehekülgede arvu, mis projekti jooksul koostatakse.
3. Nõuded muutuvad keskmiselt 1% kuu jooksul, ehk 2-aastase projekti funktsionaalsus on tarnimise ajaks suurenenud 24%.
4. Võttes funktsioonpunktide arvu astmesse 1,2, saame teada umbkaudse test case'ide arvu.
5. Võttes funktsioonpunktide arvu astmesse 1,25, saame teada umbkaudse defektide potentsiaali uue tarkvaraprojekti jaoks. Defektide potentsiaali all mõistab Capers vigade arvu neljas põhitulemis: nõuded, disain, kodeerimine, kasutajadokumentatsioon, pluss veel vead, mis tekitatakse eelnevate vigade parandamisel.
6. Iga ülevaatuse, inspektsiooni või testi käigus leitakse ja parandatakse umbes 30% olemasolevatest defektidest.

7. Võttes funktsioonpunktide arvu astmesse 0,4, saame teada umbkaudse aja kuudes, mis kulub tarkvara tootmiseks alates spetsifikatsioonist kuni tarneni.
8. Jagades funktsioonpunktide arvu 150ga, saame teada umbkaudse arvu palju läheb projekti jaoks vaja inimesi.
9. Jagades funktsioonpunktide arvu 500, saame teada umbkaudse arvu palju inimesi läheb vaja tarkvara hooldamiseks/uuendamiseks projekti lõppedes.
10. Töömäära teada saamiseks tuleb korrutada projekti kogukestvus kuudes projektiliikmete arvuga.

Need "pöidlareeglid" võivad küll populaarsed olla, kuid neid ei tohiks keegi puhta kullana võtta.

III-II Motivatsioon

Motivatsioon on muutunud tõsiseks probleemiks, seda eriti just kogenud tegijate seas, keda pelgalt rahaga enam ei motiveeri (eks kõigil ole tegelikult oma hind). Algajate jaoks on raha reeglina piisav motivaator. Kui töötaja ei ole piisavalt motiveeritud, siis ei ole ta ka produktiivne ja selle all kannatab kogu projekt ja seeläbi ka organisatsioon tervikuna. Oleks vale arvata, et kevade saabumisega inimesed jälle motiveeritumaks/reipamaks muutuvad ainult selle tulemusena, et Päike end rohkem näitama hakkab ja ilmad soojemaks muutuvad – kui on märgata, et produktiivsus ja/või motivatsioon on langenud, tuleks põhjusi otsida esmajärjekorras töökohas.

Motivatsiooni puudumise põhilised põhjused:

- Üksluine töö: töö ei paku piisavaid väljakutseid, suur hulk projektidest on liiga sarnased või projekt kestab lihtsalt liiga kaua ja edasiminekut on raske märgata.
- Töö röövib liiga suure osa ajast: inimesel pole aega eraelu jaoks, paratamatult tekib mõte: "Mille nimel ma üldse elan?" → stress, depressioon...
- Tunnustuse puudumine: tunnustamine hõlmab endas nii materiaalset, kui ka moraalset külge. Tunnustamise rõhuasetus pole aga tihti paigas, ehk piirdutakse šampuse ja õlapatsutusega.

- Liiga karm keskkond: inimese osakaalu alaväärtustatakse, eesmärkide õigeaegne saavutamine on prioriteet number 1 ja selle saavutamiseks viise ei valita. Samas võib karm keskkond viidata ka näiteks konditsioneeri puudumisele.
- Usu puudumine loodavasse tootesse: ei tulda millegi uuega välja vms.

Mis motiveerib tarkvaraarendajaid [6]:

- Personaalne tähelepanu, tunnustus, innustamine: oluline on just see, et see tunnustus tuleks ülevamalt poolt, isegi tippjuhtkonnalt.
- Materiaalne premeerimine: tihti just raha, kuid raha ei pane siiski kõiki rattaid käima.
- Uued väljakutsed: ei teki mandumise tunnet.
- Uhked ametitiitlid, diplomid: juhtivprogrammeerija, märtsi parim töötaja vms.
- Isiklik sõltuvus: inimene väga armastab oma tööd.
- Pinge: mõõduka pinge all töötamine aitab tagada eesmärkide saavutamist.
- Vastutus: inimene tunneb, et temast sõltub rohkem.
- Uued ja paremad töövahendid.
- Lisapuhkus.
- Aktsiad: inimene on firma positiivsest käekäigust rohkem huvitatud ja sellega saab ka head inimest firmaga paremini siduda.
- Loominguvabadus: aitab kaasa uute innovaatiliste lahenduste leidmisele.
- Ühtekuuluvustunne: meeskond=1.
- Soodustused: näiteks võimalus osta firma kaudu odavamalt sülearvuti või auto, samuti tasuta lõunad vms.
- Firmaüritused: piknikud, suvepäevad, reis kogu kollektiiviga soojale maale või suusatama jne.

Töötajate motiveerimiseks võib leida peaaegu loendamatul arvul võimalusi. Üks hea võimalus hoida töötajaid väikese pinge all, on jagada projekt väikesteks etappideks, millel igaühel on oma graafik – see ei lase töötajal tekkida “aega on küllaga” tunnet [6]. Nagu ka eelpool öeldud sai, ei ole raha alati peamine motivaator, tihti piisab ka tunnustusest, mis peab aga tulema ülevaltpoolt!

III-III Jälgitavus

Projekti jälgitavus tähendab projekti läbipaistvust – kui suur osa tööst on tehtud, kui suur on veel tegemata. Suurim jälgitavuse probleem on see, et see lihtsalt ei toimi või kui toimib, siis väga pealiskaudselt. Aruanded progressist põhinevad vaid ütlustel, mida ei kontrollita või ei olegi võimalik alati kontrollida. Väga levinud on ütlus "90% valmis" [7]. Samuti teeb jälgimise raskeks info väga aeglane liikumine – et vajalik informatsioon jõuaks õige inimeseni, läbib see enne pika kadalipu, mille jooksul võib see moonduda, kas siis kogemata või halvemal juhul ka tahtlikult. Järgmine üldlevinud põhjus miks jälgimine raske on, on see, et dokumentatsioon ei vasta tegelikkusele, on aegunud, dokumentatsiooni ei uuendata peale muudatuste tegemist. Üks põhilisi probleeme on veel see, et projekti etapid on liiga pikad ja/või nad pole piisavalt madalal tasemel defineeritud. Kuna mitmed projektid hõlmavad ka allhankijaid, siis on pidevaks probleemiks ülevaate saamine nendele usaldatud tööde progressist, mis aga võib terve projekti õnnestumise ohtu seada.

Üks hea viis projekti "läbipaistvamaks" muutmiseks on teatud kontrollmehhanismi (KMeh) sisseviimine ja loomulikult ka detailne planeerimine. Planeerimise käigus tuleb projekt jaotada võimalikult väikesteks etappideks (üks ülesanne, üks moodul vms.). Tööd peavad olema täpselt ära jaotatud teostajate vahel. Kui keegi väidab, et asi on valmis, siis tuleb see KMeh'i alla esitada, kus see üle kontrollitakse – kontrollitakse vastavust spetsifikatsioonile, nõuetele kvaliteedi osas jne. Kui antud tehis läbib KMeh'i edukalt, siis on see tõepoolest valmis, mitte on 90% valmis. Kui tehis KMeh'i edukalt ei läbi, siis peab selle autor veel vaeva nägema. Selline jäik ja karm kontroll võib küll projekti kulgu pikendada, kuid see tasub end siiski ära, kuna vastasel juhul võibki projekt lõppmatuseni 90% valmis olla. [7]

Jälgitavuse suurendamiseks tuleb kasuks ka anonüümne tagasiside kanal, mis võib olla nii projekti veebilehel üks vormike, kui ka kõikidele kasutamiseks mõeldud antud projekti jaoks loodud meiliaadress. Antud juhul on väga tähtis just anonüümsus, et tagada info kiirem laekumine ning ka adekvaatsema info laekumine. [7]

Allhankijatega toime tulemiseks tuleb nõuda neil sarnase KMeh'i rakendamist (vähemalt antud osaprojekti raames). Samuti tuleb allhankijatel kindlate eelnevalt paika pandud intervallide järel esitada eduraporteid, mis peaksid endas sisaldama vähemalt järgnevaid punkte:

- Tööde seis: millega hetkel tegeletakse, kas ollakse graafikus.
- Tulemid: näiteks kasutusjuhud, dokumentatsioon jne.
- Riskianalüüs.
- Probleemid ja lahendusettepanekud.
- Muudatused ajakavas, tööde ulatuses ja ressursikasutuses.

Jälgitavus tõenäoliselt siiski on ja jääb mingil määral probleemiks, kuna uusi mehhanisme on keeruline juurutada ning inimloomust on ka raske muuta. Inimestele ei meeldi, kui neid pidevalt kontrollitakse, nende tegemisi jälgitakse – tekib valvamise tunne, mis aga pole iseenesest sugugi meeldiv.

III-IV Dokumenteerimine

Dokumenteerimine on üks tüüpilisemaid probleeme, mis tarkvaraarenduses esineb, täpsemalt selle vähesus, kuid mõningatel juhtudel ka selle liigne tekitamine. Dokumenteerimise osatähtust veel päris täpselt eriti ei mõisteta või siis ei tahetagi mõista. Kusagilt on külge jäänud arusaam "mida vähem, seda parem". Kust? Kas asi on viitsimises, oskustes, teadmistes, seda ei peeta lihtsalt tähtsaks või tuleb hoopis agiilmetoodikaid süüdistada? Kuid dokumentatsiooni vähesusega probleemide ring veel otsa ei saa. Suur probleem on dokumentatsiooni pealiskaudsus: spetsifikatsioon pole piisavalt detailne, kohustused pole kirjalikult paigas ja täpselt formuleeritud, info liikumise skeem pole paigas. Pealiskaudsus on peamiselt tingitud ajanappusest, kogemuste puudumisest ja dokumendivormide puudumisest. Samuti ei kasutata piisavalt koodisisest kommenteerimist, mis teeks koodi mõistmise teiste arendajate jaoks oluliselt lihtsamaks ja arusaadavamaks (näiteks: int i; /*muutuja i*/ pole piisav!). Koodisisene kommenteerimine on kasulik ka arendaja enda jaoks, et hiljem oleks lihtsam näiteks muudatusi sisse viia. Tüüpiline on veel see, et dokumentatsiooni

ei uuendata, mille tulemusena ei vasta see ka 100% reaalsusele. Dokumentatsiooniprobleemide osas annavad tooni veel selle vormitus ehk mitteformaalsus ning hajusus. Liigne dokumentatsiooni tekitamine ei tohiks Eestis veel probleemiks olla, seepärast ka antud teemal ma pikemalt ei peatu.

Dokumentatsioonialaseid probleeme on targem ennetada kui neid hiljem püüda lahendada. Kui planeerimise käigus või isegi organisatsiooni tasemel need asjad paika panna, siis ei tohiks ka need probleemid nii tõsiselt esile kerkida. Dokumentatsiooni koostamise alane koolitus tuleks kindlasti kasuks. Samuti on oluliselt lihtsam dokumenteerimisega tegeleda, kui on olemas spetsiaalsed dokumendivormid – dokumenteerimine on organisatsioonis standardiseeritud. Projekti kestel tehtavad muudatused peavad kajastuma ka projektiplaanis ja ka kõikides teistes asjaga seotud olevates dokumentides. See mida projektiplaan sisaldada võiks on kirjas Lisade osas. Mida dokumenteerida? Dokumenteeri seda, mis on oluline! (M. Karu) Mis on oluline? See on tõenäoliselt projekti –ja organisatsioonispetsiifiline asi. Määrav osa selles on projekti suurusel, kriitilisusel ja tulevikul (kas antud tarkvara kavatsetakse tulevikus veel laiendada).

Dokumenteerimine on tarkvaraprojekti vältimatu osa, küsimus on vaid selles mida dokumenteerida ja mil määral. Kui dokumentatsiooniga üle pingutada, siis kokkuvõttes pikeneb projekti kogupikkus ja suureneb mittetootliku töö maht. Kui aga dokumentatsiooni osakaal on liiga väike, siis võib tekkida probleeme projekti haldamisega, jälgitavusega, arendatavusega.

III-V Kommunikatsioon

Kommunikatsioon on iga projekti kohustuslik osa. Kui info ei liigu, siis jäävad osapooled paljustki ilma, mille all aga kannatab jälle projekt. Kommunikatsiooni all võib mõista nii koosolekuid, aruandlust, läbivaatusi, kohvinurka, info jagamist e-maili, veebilehekülje kaudu. Kommunikatsioon on nii positiivne, kui ka mõningatel juhtudel negatiivne. Suurimad miinused selle juures on selle vähesus (infosulg) ja ka selle rohkus (müra). Tõsine probleem, mis mu uurimusest välja tuli on see, et

informatsiooni jagatakse nö *on-need-to-know basis*, mis takistab täitjatel projekti täielikult sisse elada (puudub ülevaade projektist kui tervikust) ja sellega ka endast maksimaalset anda. Antud nähtust nimetan ma "raudseks eesriideks". Tõsiseks probleemiks on veel nö lobamokad. Need on inimesed, kellel on äärmiselt suur suhtlemisvajadus – sellega nad aga segavad teiste inimeste tööd (näiteks programmeerijatele ei meeldi absoluutselt kui neid mõttepauside ajal segatakse – häirib nende keskendumisvõimet). Tööl käiakse reeglina ikka tööd tegemas. Kui programmeerijad ise end kaitsta ei suuda, siis peab selle probleemi ära lahendama keegi teine, näiteks projektijuht.

Koosolekud on projektide puhul üks levinumaid asjade arutamise kohti, kuid ka nende juures tehakse palju vigu:

- Päevakord pole eelnevalt paigas.
- Kohale ei kutsuta ainult asjasse puutuvad inimesed, vaid ka nö valed inimesed.
- Arutletakse tühiste probleemide üle, mille lahendamine ei nõua meeskonna kokku kutsumist, vaid näiteks ainult veidi lisatööd projektijuhilt või analüütikult.
- Projektimeeskonna liikmed ei tule kohale.
- Ei protokollita piisavalt detailselt.

Et näiteks antud probleeme ei esineks, tuleks päevakord eelnevalt paika panna ja sellest teavitada kõiki asjasse puutuvaid isikuid. Koosoleku kokku kutsumiseks peavad olema tõsised põhjused, mida peab arutama koos teiste asjaga seotud olevate inimestega. Kui arutusele tulevate probleemide ring on väga oluline, siis peab koosolek olema ka kohustuslik ning ka sellest tuleb osalejaid teavitada; mitteilmumisel on mõningatel juhtudel mõistlik rakendada ka sanktsioone. Et vastuvõetud otsustest ja kohalviibinutest ka jälg jääks, tuleb koosoleku tulemid protokollida piisava detailsusega, et ka hiljem oleks võimalik teada saada, mis täpselt ära otsustati ja ka kes otsuse täideviimise eest vastutav on.

Projekti aruandlusskeem peab olema väga täpselt paigas: millal, mida, kes, kellele. Korralik aruandlusskeem tagab projekti parema organiseerituse ja koordineerituse. Aruandluse käik peab olema kirjas ka projektiplaanis.

Kogu projekti puudutav informatsioon peab olema projektiliikmete jaoks kättesaadav. Üks mugav viis selleks on luua projekti intranetilehekülg, kus on kirjas kõik projektiga seonduv: plaanid, hetkeseis, muudatused, kohustused, riskid jms. See tagab selle, et kõik projekti liikmed on projekti seisuga, iseärasustega kursis, mis aga omakorda tagab parema lõpptulemi.

Informatsiooni filterdamine on tarkvaraprojektide puhul väga oluline tegevus, mille eest peab hoolt kandma projektijuht. Filterdamise eesmärk seisneb selles, et projektiliikmeid ei koormata kõikvõimaliku ebaolulise informatsiooniga, mis võib nende tööd mõjutada. Näiteks ei tohiks teostajateni jõuda informatsioon kliendi muudatusettepanekutest enne, kui need on juhtrühma poolt kinnitatud. [6]

III-VI Muudatuste juhtimine

Projekti kestel ettetulevad muudatused on iga halvasti planeeritud tarkvaraprojekti "õudusunenäod". Eriti kriitiliseks läheb asi siis, kui muudatusi hakatakse nõudma projekti lõpufaasides. Kuna pole eelnevalt paika pandud kuidas muudatusi adresseerida, siis muutub projekti kulg hiljem kriitiliseks. Muudatustega tuleb arvestada juba planeerimise faasis. Juhitamatud muudatused on tüüpiline põhjus mis projektid hilinevad. [7]

Tüüpilised vead mida tehakse:

- Arvatakse (loodetakse), et spetsifikatsioon on lõplik.
- Muudatuste mõju ei hinnata.
- Üritatakse mahutada uusi töid olemasolevasse ajagraafikusse: graafik muutub veel ebareaalsemaks.
- Dokumentatsioon ei kajasta muudatusi: dokumentatsioon != reaalsus.
- "Unustatakse ära", et kui omadusi lisatakse, siis ajagraafik pikeneb.
- "Unustatakse ära", et kui ajagraafikut ei saa pikendada, siis peab midagi ära jätma või hiljem tegema.

Spetsifikatsioon koostatakse reeglina projekti algul. Projekti alguses ei ole aga kõik nõuded siiski teada, kuna klient veel isegi päris täpselt ei tea, mis funktsionaalsust loodav tarkvara peab omama – on ainult visioon. Spetsifitseerimise etapis üritatakse seda küll täpsustada, kuid 100% nõuete välja selgitamist ei suudeta siiski tagada. Paratamatult tekivad uued nõuded, mis on kliendile olulised. Kuna loodav tarkvara on tavaliselt väga oluline faktor kliendi ärieesmärkide saavutamisel, tuleb muudatusi ka piisava hoolega hallata. Spetsifikatsioon on "elav" dokument!

Muudatuste juhtimiseks tuleb luua kindel süsteem [6]:

- Kõik muudatusettepanekud tuleb teha kirjalikult kasutades selleks spetsiaalselt ettenähtud dokumendivorme.
- Muudatusettepanek esitatakse projektijuhile, kes hindab selle mõju ajagraafikule, kuludele ning juba tehtud ja ka tulevastele töödele.
- Projektijuht viib hinnatud muudatusettepaneku juhtrühmale, kes otsustab nende teostamise üle.

Kui muudatus otsustatakse ka teostada, tuleb kindlasti uuendada ka projektiplaani ja teisi sellega seotud olevaid dokumente. Näidis muudatusettepanekust on Lisade osas, lk 54.

Kuna reeglina mõjutavad muudatused nii ajagraafikut, kui ka muid ressursse (näiteks maksumus), siis on väga oluline, et seda teadvustaks ka tellija. Tellija peab juba algfaasis tunnistama, et muudatused võivad nõuda temalt lisaressursse ja et ajagraafik võib nende tulemusena pikeneda. Muudatuste sisseviimise kord peaks olema kirjas ka lepingus.

III-VII Koordineerimatus

Koordineerimatuse all mõistan ma projekti juhtliikmete vähest kontrolli teostajate üle, mille põhjuseks on pealiskaudne planeerimine ja/või nõrk juht. Projektiplaanis pole kirjas rollid ja vastutused, faasid ja ajakava, ülesannete sõltuvussuhted, juhtimine ja jälgitavus, töömeetodid, aruandlus ja alluvussuhted, samuti on projektisisene koostöö

väga pealiskaudne. Aruandlusest on kirjas eespool, seetõttu sellel antud punktis ei peatu pikemalt.

Üks suuremaid probleeme, mis koordineerimatuse tagajärjel tekkida võib on programmeeritud komponentide ühildumatus. Seetõttu tuleb töid ümber teha ning testimisele ja integreerimisele kulub planeeritust rohkem aega. Et seda ei juhtuks on koostöö väga oluline ning ka pidev komponentide integreerimine [7] on üks viis edu tagamiseks. Tööde välja jagamine ja koordineerimine on projektijuhi kohustus (kui pole määratud teisiti). Rollid ja kohustused ning aruandluse kord peavad olema kirjas projektiplaanis ning kõik projektiliikmed peavad nendega kursis olema ning neid aktsepteerima.

III-IIX Automatiseerimine

Automatiseerimise all mõistan ma konkreetse tarkvara kasutamist mingi ülesande lahendamisel või selle toetamisel. Hea tarkvara ja koolitatud ning kogenud kasutaja suudavad koostöös tagada eesmärkide kiirema ja kvaliteetsema saavutamise, kuid kogenematu kasutaja koos spetsiifilise "tööriistaga" võib tekitada korvamatut kaost. [1]

Eesti tarkvaraarendajate jaoks ei ole automatiseerimine mingi uudis, kuigi selle õige rakendamine võib siiski probleeme tekitada. Mingil määral on levinud piraattarkvara ja tarkvara prooviversioonide pidev kasutamine. Alternatiiv sellele oleks tarkvara ostmine, kuid väiksemad tarkvaraarendusfirmad ei suuda neid endale lubada ülikõrge hinna tõttu, samuti oleks võimalik kasutada vabavara, kuid selle valik ja ka kvaliteet ei ole enamasti samal tasemel, aktsepteeritaval tasemel.

Järgnevalt toon välja võimalike vahendite loetelu, mida on võimalik kasutada tarkvaraprojekti juures:

- MS Office perekond: dokumentatsiooni koostamiseks, presentatsioonide tegemiseks jne.

- MS Project: ajagraafikute koostamiseks ja projekti käigu visualiseerimiseks ja võrdlemiseks baasvariandiga. Samuti raportite jaoks, töökoormuse hindamiseks jms.
- MS SharePoint: dokumentide haldamiseks.
- Rational RequisitePro: nõuete haldamiseks.
- Rational ClearCase: muudatuste haldamiseks.
- Rational Rose: analüüsi ja disaini vahend.
- JBuilder: programmeerijate töökeskkond.
- WinCVS: tarkvara lähtetekstide versioonihalduseks.

Antud loetelu ei ole lõplik, vahendeid on äärmiselt palju ja neid tuleb valida vastavalt firma rahakotile, tulevikuplaanidele, projektide iseärasustele.

Tarkvaraprojekti automatiseerimine on samm tuleviku ja kvaliteedi tagamise suunas, kuid seda ei tohiks "pimesi" teha. Enne kui mõnda vahendit soetada, tuleb seda põhjalikult uurida ja võrrelda oma tulevikuplaanidega ja vajadustega ning loomulikult kaaluda alternatiive. Ainuüksi hea vahend ei taga veel kvaliteetset tööd, vajalik on ka kasutajate professionaalsed teadmised vahendist, selle kasust, võimalustest.

III-IX Kvaliteedi juhtimine

Kvaliteedi juhtimine kaootilise protsessi juures, kasutades selleks vaid olemasolevaid ressursse, on praktiliselt võimatu [2]. Antud juhul suudavad projekti õnnestumise tagada vaid hea õnn ja projektiliikmete ennastohverdavad kangelasteod. Arvestades Eesti tarkvarafirmade noorust, võib oletada, et suurem osa nendest CMM'i skaalal üle esimese taseme veel lähitulevikus ei tõuse, ja seda isegi siis, kui CMM'i kohandada nii, et seda saaks ka väikefirmade osas hõlpsasti rakendada. Teise taseme saavutamine oleks paljudel juhtudel piisav, et tagada kvaliteetsema tarkvara loomine [2], kuid see ei ole reegel. CMM'i mingi taseme saavutamine ei taga veel iseenesest kvaliteetset tarkvara [1], kuid protsess, mille käigus see valmib, on kuulutatud küpseks (vastava taseme vääriliseks).

Kvaliteedi tagamine piirdub paljudel juhtudel ainult testimisega ja komponentide integreerimisega. Korrektne kvaliteedi tagamine on aga hoopis keerulisem protsess [7]. Kvaliteedi tagamine peab olema iga projektiliikme töökohustuste hulka integreeritud [1]. Eelpool kirjeldatud kontrollmehhanism (KMeh) on vaid üks võimalus, kuidas tarkvara kvaliteeti tagada. Arendajate jaoks on Watts S. Humphrey välja töötanud Personal Software Process'i (PSP) [11], mille kasutamine pikemas perspektiivis parandab oluliselt nii kvaliteeti, kui ka produktiivsust [12].

III-X Projekti realisatsioonietapp ei ole piisavalt "tükeldatud"

Suuremate ja keskmiste tarkvaraprojektide puhul on üks pikk realisatsioonietapp kui "libedal jääl kõndimine". Kui realisatsioonietapp ei ole täpselt defineeritud, siis puudub ka võimalus jälgida tööde käiku ja hinnata progressi – "4 moodulit 7-st on valmis" ei ole veel piisav indikaator.

Kasutada tuleks astmelist arendust või siis iteratiivset (iteratiivne arendusmeetod on otstarbekas siiski vaid suuremate projektide puhul). Iga etapi alguses täpsustatakse spetsifikatsiooni, koostatakse detailne disain, realiseeritakse antud etapis ettenähtud funktsionaalsus, testitakse/integreeritakse, koostatakse/uuendatakse dokumentatsiooni ja sooritatakse teised vajalikud tegevused. Esimese etapi käigus realiseeritakse kõige suurema tähtsusega funktsionaalsus, järgnevate jooksul vähemtähtsad omadused [7]. Ka antud meetodi juures on kasulik kasutada eelpool kirjeldatud KMeh'i.

Realiseerimisetapi "tükeldamine" nõuab rohkem tööd ja aega, kuid see-eest on see ka kasulik mitmete näitajate osas – jälgitavus, kvaliteet [7]. Väga väikeste projektide puhul ei ole astmeline arendus muidugi eriti otstarbekas, kuna tehtavate tööde hulk on piiritletud ning teada.

III-XI Riskianalüüs

Olen uurinud erinevate projektide riskianalüüse, ja nendest on jäänud mulje, et need on tehtud ainult sellepärast, et see on olnud dokumendi kohustuslik osa – standardis ette nähtud. Riskianalüüsil oli kasutatud vaid kõige tüüpilisemaid riske ning ka nendel oli enamasti hinnanguks antud "riski esinemise tõenäosus on olematu või madal". Teine probleem oli riskidele reageerimine. Plaanis ei olnud kirjas, kuidas antud riski adresseerida – oli vaid "riski ilmnemisel planeerime vastavad meetmed", ehk põhimõtteliselt oleks siis tegu "tulekahju kustutamisega".

Riskide ennetamine, nende mõju hindamine ja vastavalt ka nende esinemiseks valmistumine on iga tarkvaraprojekti äärmiselt tähtis osa, nii et selle koostamisel tuleb sügavuti projekti uurida, üritada tulevikku ette näha ja uurida ka eelnevate analoogsete projektide käekäiku.

Riskianalüüsi tegemiseks ma soovitan kasutada SWOT-maatriksit. See toob välja projekti tugevused, nõrkused, võimalused ja ohud ning näitab ära, kuidas nendele läheneda. Ülevaate SWOT-maatriksist annab tabel 2.

	Sisemised tugevused (internal strengths - **S**)	**Sisemised nõrkused** (internal weaknesses - **W**)
Välised võimalused (external opportunities - **O**)	**SO-strateegia**: väliste võimaluste ära kasutamine kasutades sisemisi tugevusi	**WO-strateegia**: kuidas üle saada sisemistest nõrkustest, et väliseid võimalusi ära kasutada.
Välised ohud (external threats – **T**)	**ST-strateegia**: kuidas neutraliseerida väliste ohtude mõju sisemiste tugevuste abiga	**WT-strateegia**: planeerida kaitseplaan, et sisemised nõrkused ei oleks väliste ohtude jaoks eriti haavatavad.

Tabel 2: SWOT-maatriks.

III-XII Ajalooandmestik

Ajalooandmestik hõlmab endas andmeid möödunud projektide kohta. Minu teada tegelevad Eestis vaid tuntud suuremad tegijad süstemaatiliselt ajalooandmestiku kogumisega, kuigi see on väga kasulik ka väiksematele tarkvaraarendusfirmadele.

Ajalooandmestik võiks sisaldada näiteks järgmisi andmeid [6]:

- Üldine ajagraafik: planeeritud/tegelik.
- Ressursid: planeeritud/tegelik.
- Töömäär: planeeritud tegelik.
- Esile kerkinud probleemid.
- Avastatud defektid: kogus, vea aste (kriitiline, tõsine, iluviga vms.), mis faasis leiti.
- Muudatused: kogus, kuidas mõjutas projekti kulgu vms.
- Loodud tarkvara liik, suurus, kasutatud tehnoloogiad, programmeerimiskeeled.

Nimekiri oleks väga pikk. Nende andmete salvestamiseks peaks olema eraldi süsteem, mille abiga saaks ka järgnevatel kordadel uute sarnaste projektide osas hinnanguid teha. Adekvaatsete andmete olemasolu võimaldab järgnevaid analoogseid projekte oluliselt täpsemalt ennustada.

III-XIII Allhankijad

Projektide raporteid lugedes võib sealt pidevalt leida umbes sellise rea: “Projekt hilines, kuna allhankija...”. Et neid probleeme vältida, tuleks enne lepingu sõlmimist hinnata, kas allhankija on üldse võimeline tegema seda tööd kvaliteetselt olemasoleva ajagraafiku ja kuluhinnangu kohaselt. Et seda teha, tuleks küsitleda allhankija praeguseid ja ka eelmiseid kliente; küsida näha eelmiste projektide dokumente; selgitada välja allhankija oskused, kogemused just antud projekti valdkonnast lähtuvalt; selgitada välja, milline on allhankija kvaliteedisüsteem ja mil määral seda järgitakse [6].

Projekti algul tuleb ära määrata viisid, kuidas muudatusi sisse viia, progressi jälgida, tulemeid dokumenteerida ja kvaliteeti hinnata, samuti tuleb paika panna aruandluse kord. Allhankija projektiplaani peab tellija üle vaatama, et see vastaks tema poolt esitatud nõuetele. Allhankija ülesanded peavad olema tellija poolt väga täpselt spetsifitseeritud ja tellija peab olema veendunud, et allhankija meeskond on ka nendest õigesti aru saanud. Projekti ajal peab allhankija esitama kindlate intervallide järel tööde edenemisraporteid ja soovitavalt ka ülejäänud dokumentatsiooni, mis antud töödega seotud on, et tellija saaks need ka üle kontrollida ning nende kvaliteedis veenduda. Allhankija poolt kirjutatud programmi lähtekoodi peaks tellija meeskond üle vaatama ning valmis programmi testima [6].

Õige allhankija leidmine võib problemaatiliseks muutuda, kui nõudmised piisavalt kõrged on, kuid samas järeleandmisi ei tohiks ka teha, kuna sellega kaasnevad uued riskid. On täiesti iseenesestmõistetav, et allhankijalt tuleb nõuda sama kõrget kvaliteeti kui endalt.

Kui mõne projekti puhul peaksid ilmnema kõik eelpool välja toodud probleemid, siis on tegu ikka väga halvasti planeeritud projektiga. Oleks huvitav endalgi teada, mis sellest projektist lõpuks sai. Et need probleemid kontrolli alla saada, tuleks hakata kindlasti mõtlema protsessi arendamise peale. Kuna olemasolevate mudelite ja metoodikate hulk on päris suur, siis esialgu võib nendes orienteerumine lõputu katsumusena näida. Neil kõigil on omad head ja vead. Milline on sobiv just Sinu firmale?

Käesolevas peatükis käsitlesin levinumaid probleeme, milledest võib järeldada, et paljudel Eesti tarkvaraarendusega tegelevatel ettevõtetel esineb projektide osas mitmeid ühiseid probleeme, milledest valdav osa tuleneb vähesest planeerimisest ja juhtimise osakaalu alahindamisest projekti õnnestumisel.

IV Konventsionaalse tarkvaraarenduse põhiprintsiibid

Aastate jooksul on kirja pandud väga palju tarkvaraarenduse põhiprintsiipe erinevate inimeste poolt. Mingi osa nendest on juba ammu aegunud, teisi on kohandatud, et need ka tänapäeval paremini kehtiksid, ülejäänud on aga aegumatud. Järgnevalt esitan valiku Alan M. Davise poolt kirja pandud konventsionaalse tarkvaraarenduse põhiprintsiipidest [1][8], mis ka tänapäeval kehtivad ja aktuaalsed on.

- Kõrge kvaliteediga tarkvara on võimalik teha. Meetodid, mis on end õigustanud kui kvaliteedi parandajad, on tellija kaasamine, prototüüpimine, disaini lihtsustamine, inspektsioonid, parimate inimeste palkamine.
- Anna toode varakult tellija kätte. Ükskõik kui väga sa ka ei püüa spetsifitseerimise faasis kõiki nõudeid kirja saada, on kõige efektiivsem viis kasutaja tõelisi nõudmisi välja uurida siiski andes kasutajale tarkvara kätte, et ta saaks sellega "mängida".
- Kasuta sobivat protsessimudelit. Iga projekti jaoks tuleb valida protsessimudel, mis on selle jaoks kõige mõistlikum, võttes aluseks loodava tarkvara valdkonna, nõuete muutuvuse, määra, milleni nõuded on paigas juba.
- Minimeeri intellektuaalset distantsi. Tarkvara struktuur peab olema võimalikult lähedane pärismaailma struktuurile.
- Tee see korda enne, kui hakkad seda kiiremaks muutma. On palju lihtsam muuta töötavat tarkvara kiiremaks kui üritada kiiret programmi tööle saada. Ära esmase kodeerimise juures optimiseerimise pärast muretse.
- Hea juhtimine on tähtsam kui hea tehnoloogia. Isegi parim tehnoloogia ei kompenseeri halba juhtimist – hea juhtimine seevastu aga suudab tagada eesmärkide saavutatuse ka nappide ressursside korral. Hea juhtimine motiveerib inimesi andma endast parimat, kui kahjuks ei ole mingit universaalselt "õiget" juhtimisstiili.
- Inimesed on edu alus. Suurte oskustega inimesed, kellel on nii kogemusi, kui ka talenti on edu aluseks. Õiged inimesed ebapiisavate vahendite ja nõrga protsessiga suudavad ikka edu saavutada. Valed inimesed sobivate vahenditega ja protsessiga tõenäoliselt hävivad.

- Püüa mõista tellija prioriteete. On võimalik, et tellija lepiks sellega, et 90% funktsionaalsusest jõuab temani hiljem, kui ta saab 10% kätte õigel ajal.
- Disaini mõeldes muutuste peale. Arhitektuurid, komponendid, spetsifitseerimise meetodid, mida sa kasutad peavad olema tulevikus lihtsalt muudetavad.
- Kapselda. Kapseldamine on lihtne, järele proovitud kontseptsioon, mille tulemusena on tarkvara lihtsam testida ja hallata.
- Inimesed ja aeg ei ole vaheldatavad. Projekti "mõõtmine" võttes arvesse vaid inimkuid ei oma erilist mõtet.
- Eelda täiuslikkust. Sinu alluvad saavad palju paremini hakkama, kui sa ootad neilt palju.

Kokkuvõte tööst

Antud töö toob välja levinumad probleemid, mis tarkvaraprojektide puhul esile kerkivad ning nõuandeid nende ennetamiseks ja/või lahendamiseks. Uuritud on 14 Eesti tarkvaraarendusega tegeleva ettevõtte projektide levinumaid probleeme ning nende tagamaid. Uuringud põhinevad suures osas vabas vormis läbi viidud küsitlustel, milledest osa sai läbi viidud ka Interneti vahendusel kasutades MSN Messengeri.

Esimene peatükk tutvustab laialt levinud tarkvaraprotsessi küpsuse hindamise mudelit Capability Maturity Model (CMM), selle ajalugu, struktuuri, negatiivseid külgi; täpsemalt on kirjeldatud CMM'i teist taset, mis keskendub suures osas projekti tasemele.
Teine peatükk vaatleb tarkvaraprotsessi arendamise kasulikkust organisatsiooni jaoks, mida selle korrektne rakendamine endaga kaasa toob lähtuvalt Software Engineering Institute'i poolt läbi viidud uurimusest.
Kolmandas peatükis on välja toodud minu poolt läbi viidud uurimuse tulemused. Nendest võib järeldada, et paljudel Eesti tarkvaraarendusega tegelevatel ettevõtetel esineb projektide osas mitmeid ühiseid probleeme, milledest valdav osa tuleneb vähesest planeerimisest ja juhtimise osakaalu alahindamisest projekti õnnestumisel.
Viimane peatükk käsitleb valikut Alan M. Davis'e konventsionaalse tarkvaraarenduse põhiprintsiipidest, mis kehtivad ning on ka aktuaalsed tänapäeval.

Lühidalt võiks töö võiks kokku võtta järgnevalt: probleemide ilmnemisel tuleb need muidugi kõrvaldada, kuid pikemas perspektiivis on kasulikum siiski nende ennetamine ja tarkvaraprotsessi arendamine.
Isegi kui katastroofi tegelikud põhjused suudetakse leida, siis lahendus ei pruugi seisneda ainult nende "lappimises". Ilma põhjaliku plaanita protsessi järk-järguline paremaks muutmine oleks nagu viie liitri vee surumine kolmeliitrisesse purki. [3]

Töö edasiarendamise võimalusi tulevikus

Kuna mul on plaanis õpinguid jätkata magistriõppes, oleks antud tööd võimalik edasi arendada ka magistritööks, mis haaraks endas suuremat osa Eesti tarkvarafirmadest, see aga annaks parema võimaluse üldistuste tegemiseks Eesti tarkvarafirmade hetkeseisust neil esinevate probleemide vallas. Teine võimalus oleks võtta aluseks mingi konkreetne tarkvaraprojekt ning koostada selle juhtumianalüüs.

Tänusõnad

Käesoleva töö koostamisel on suur abi olnud kolmest inimesest. Järgnevalt tahangi tänada oma juhendajat, prof. Peeter Normakut, kelle nõuanded ja kriitiline meel aitasid seda tööd kujundada. Samuti tahan tänada Marion Lepasaart (Tampere Tehnikaülikool, doktorant) ja Tarmo Robalit (Tallinna Tehnikaülikool, magistrant) nende ülimalt väärtuslike soovituste eest kogu töö valmimise käigus.

Lisad

Lisa 1: CMM 2. taseme küsimustik

[13][10]

Nõuete haldus

1. Kas tarkvarale esitatavaid süsteeminõudeid kasutatakse tarkvara arenduse ja selle juhtimise alusena?
2. Kui tarkvarale esitatavad süsteeminõuded muutuvad, kas siis tehakse ka vastavad muudatused tarkvaraarendusplaanidesse, tegevustesse ja toodetesse?
3. Kas projekt järgib tarkvarale esitatud süsteeminõuete haldamiseks organisatsiooni poolt kehtestatud korda?
4. Kas projektis nõudehaldusega tegelevad inimesed on vastavate protseduuride läbiviimiseks koolitatud?
5. Kas nõudehaldusega seotud tegevuste seisundi hindamiseks kasutatakse mingeid mõõtusid (nt. muutmistaotluste koguarv, avatud, heaks kiidetud ja arendusalusesse liidetud muutmistaotluste arv)?
6. Kas projekti nõudehaldusega seotud tegevused on allutatud tarkvara kvaliteedikontrolli (Software Quality Assurance – SQA) läbivaatustele?

Tarkvaraprojekti planeerimine

1. Kas ennustused (nt. projekti suurus, hind ja ajagraafik) on planeerimiseks ja projekti jälgimiseks dokumenteeritud?
2. Kas projekti plaanid dokumenteerivad neid tegevusi, mida tehakse, ja korraldusi, mis on antud projektiga seoses antud?
3. Kas kõik grupid ja isikud on nõus neid mõjutavate tarkvaraprojektiga seotud kohustustega?
4. Kas projekt järgib kirja pandud organisatsiooni tarkvaraprojekti planeerimise poliitikat?
5. Kas tarkvaraprojekti plaanimisel on kasutada adekvaatsed ressursid (st. finantsvahendid ja kogenud isikud)?

6. Kas tarkvaraprojekti tegevuste plaanimise seisu tuvastamiseks kasutatakse mõõtmisi (st. tähtpunktidesse jõudmisel võrreldakse projekti plaanimise tegevusi plaaniga)?
7. Kas projekti juht vaatab läbi tarkvaraprojekti plaanimise tegevusi nii perioodilisel alusel kui ka sündmuste alusel?

Tarkvaraprojekti jälgimine ja ülevaadete tegemine

1. Kas projekti tegelikke tulemusi (graafik, suurus ja hind) võrreldakse hinnangutega tarkvara plaanides?
2. Kui tegelikud tulemused erinevad projekti tarkvaraplaanidest märkimisväärselt, kas siis võetakse ette parandamistegevusi.
3. Kas muutused tarkvarale esitatavate nõuete osas kooskõlastatakse kõikide nende muutuste poolt mõjutatavate gruppide ja isikutega?
4. Kas projekt järgib kirja pandud organisatsiooni poliitikaid nii tarkvara väljatöötamise tegevuste jälgimise kui juhtimise osas?
5. Kas spetsiaalselt tarkvara väljatöötamise tegevuste ja toodete (st. tööpanused, graafik ja eelarve) jälgimise eest on keegi vastutav?
6. Kas tarkvara väljatöötamise jälgimise ja järelvaatuse oleku määramiseks kasutatakse mõõtmisi (st. kogu jälgimise ja järelvaatuse tegevuste peale kulutatud tööpanused)?
7. Kas tarkvaraprojekti jälgimise ja järelvaatuse tegevusi vaadatakse perioodiliselt läbi tippjuhtkonnaga (st. projekti suutvus, avatud probleemid, riskid ja astutavad sammud)?

Alltöövõttude haldus

1. Kas alltöövõtjate töö tegemise võimekuse järgi valimiseks kasutatav protseduur on dokumenteeritud?
2. Kas muutused alltöövõttudes toimuvad peatöövõtja ja alltöövõtja kokkuleppel?
3. Kas alltöövõtjatega viiakse läbi perioodilisi tehnilisi kooskõlastusi?
4. Kas tarkvara alltöövõtja tulemusi ja suutvust jälgitakse talle pandud kohustuste suhtes?

5. Kas projekt järgib tarkvara alltöövõttude haldamisel kirja pandud organisatsiooni poliitikat?
6. Kas inimesed, kes vastutavad tarkvara alltöövõttude haldamise eest, on tarkvara alltöövõttude haldamiseks koolitatud?
7. Kas tarkvara alltöövõttude haldamise tegevuste seisu määramiseks kasutatakse mõõtmisi (st. kalenderplaani seis plaanitud üleandmiskuupäevade suhtes ja alltöövõtu haldamiseks kulutatud tööpanus)?
8. Kas tarkvara alltöövõtja tegevusi vaadatakse läbi projektijuhiga nii perioodilisel alusel kui ka sündmuste alusel?

Tarkvara kvaliteedikontroll

1. Kas SQA tegevusi plaanitakse?
2. Kas SQA tegevused võimaldavad objektiivselt kontrollida et tarkvaratooted ja tegevused vastavad rakendatavatele standarditele, protseduuridele ja nõudmistele.
3. Kas neid gruppe ja isikuid, keda SQA läbivaatused ja auditid puudutavad, varustatakse SQA läbivaatuste ja auditite tulemustega (st. neid kes töö teostasid ja neid, kes töö eest vastutavad)?
4. Kas projektis lahendamata jäänud mittevastavuste küsimusi (st. Kõrvalekalded rakendatavatest standarditest) vaadeldakse tippjuhtkonna tasemel?
5. Kas projekti järgib SQA teostamisel organistatsiooni kirja pandud poliitikat?
6. Kas SQA tegevuste sooritamiseks on saadaval piisavalt ressursse (st. Rahaline ressurss ja määratud juht, kes tegeleb tarkvara mittevastavuse küsimustega)?
7. Kas SQA sooritatavate tegevuste seisu ja hinna määramiseks kasutatakse mõõtmisi (st. tehtud töö, tööpanused ja kulutatud rahaline ressurss plaaniga võrreldes)?
8. Kas SQA tegevusi vaadatakse tippjuhtkonna poolt läbi perioodilisel alusel?

Tarkvara konfiguratsioonihaldus

1. Kas tarkvara konfiguratsiooni halduse tegevusi plaanitakse?
2. Kas projekt tuvastas, haldas ja tegi tarkvara väljatööted saadavaks konfiguratsioonihaldust kasutades?
3. Kas konfiguratsiooni elementide/ühikute muutmise haldamiseks järgib projekt dokumenteeritud protseduuri?

4. Kas tarkvara arendusaluse standardaruandeid (st. tarkvara muutmisnõukogu protokollid, muutmistaotluste koond ja seisuhinnangud) jagatakse neile gruppidele ja isikutele, keda need puudutavad?

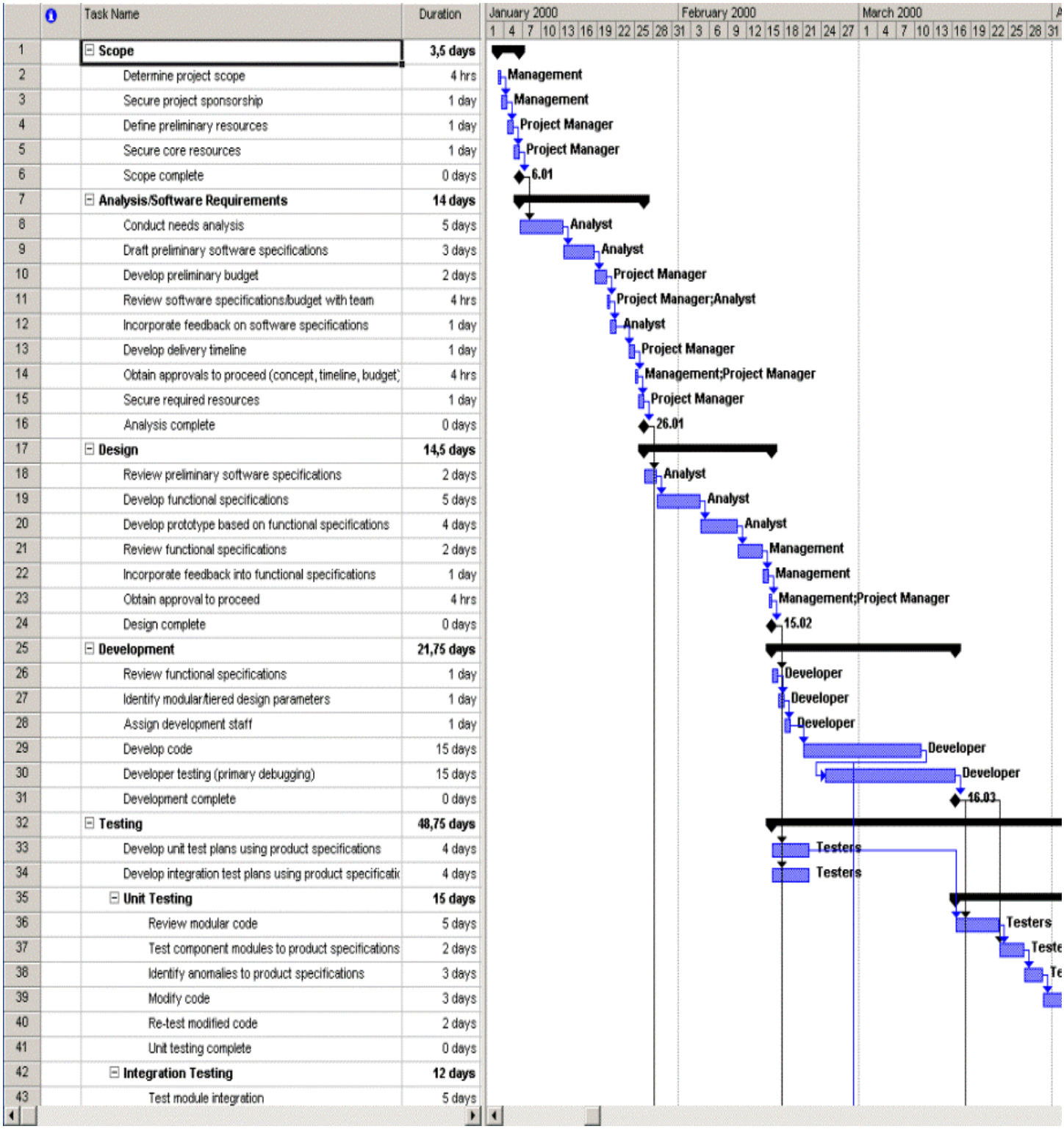
5. Kas projekt järgib tarkvara konfiguratsiooni halduse tegevuste teostamisel kirja pandud organisatsiooni poliitikat?

6. Kas projekti personal on koolitatud sooritama neid tarkvara konfiguratsiooni halduse tegevusi, mille eest nemad vastutavad?

7. Kas tarkvara konfiguratsioonihalduse tegevuste oleku määramiseks kasutatakse mõõtmisi (st. tarkvara konfiguratsioonihalduse tegevustele kulutatud tööpanused ja rahalised ressursid)?

8. Kas tarkvara arendusaluse vastamist seda defineerivale dokumentatsioonile kontrollitakse perioodiliste auditite ga (st. SCM grupi poolt)?

Lisa 2: Projekti ajagraafiku näidis kasutades MS Project 2000 projektihaldustarkvara



	Task Name	Duration
1	**Scope**	**3,5 days**
2	Determine project scope	4 hrs
3	Secure project sponsorship	1 day
4	Define preliminary resources	1 day
5	Secure core resources	1 day
6	Scope complete	0 days
7	**Analysis/Software Requirements**	**14 days**
8	Conduct needs analysis	5 days
9	Draft preliminary software specifications	3 days
10	Develop preliminary budget	2 days
11	Review software specifications/budget with team	4 hrs
12	Incorporate feedback on software specifications	1 day
13	Develop delivery timeline	1 day
14	Obtain approvals to proceed (concept, timeline, budget)	4 hrs
15	Secure required resources	1 day
16	Analysis complete	0 days
17	**Design**	**14,5 days**
18	Review preliminary software specifications	2 days
19	Develop functional specifications	5 days
20	Develop prototype based on functional specifications	4 days
21	Review functional specifications	2 days
22	Incorporate feedback into functional specifications	1 day
23	Obtain approval to proceed	4 hrs
24	Design complete	0 days
25	**Development**	**21,75 days**
26	Review functional specifications	1 day
27	Identify modular/tiered design parameters	1 day
28	Assign development staff	1 day
29	Develop code	15 days
30	Developer testing (primary debugging)	15 days
31	Development complete	0 days
32	**Testing**	**48,75 days**
33	Develop unit test plans using product specifications	4 days
34	Develop integration test plans using product specificati	4 days
35	**Unit Testing**	**15 days**
36	Review modular code	5 days
37	Test component modules to product specifications	2 days
38	Identify anomalies to product specifications	3 days
39	Modify code	3 days
40	Re-test modified code	2 days
41	Unit testing complete	0 days
42	**Integration Testing**	**12 days**
43	Test module integration	5 days

Joonis 6: Ajagraafiku näidis MS Projectis.

Lisa 3: Projektiplaani soovituslik sisu

1. Sissejuhatus: annab üldpildi kogu dokumendist
   1.1 Dokumendi eesmärk: Millist eesmärki dokument teenib, kellele on dokument ette nähtud.
   1.2 Toode: toote nimi, tähendus, eesmärgid/kasud.
   1.3 Spetsifikatsioon, terminid, lühendid, lühinimed.
   1.4 Viited ja lisadokumendid.
2. Ülesande püstitus
   2.1 Taust.
   2.2 Eesmärk: kirjeldatakse projekti eesmärk või eesmärgid.
   2.3 Ülesanded: Lühike projektiplaani kuuluvate ülesannete kirjeldus.
   2.4 Piirid / piirtingimused: ülesanded, mis ei kuulu projekti alla; tingimused, mille puhul projekt realiseerub.
   2.5 Kliendi kohustused: ülesadned, mis ei kuulu antud projekti meeskonna vastutusalasse.
3. Projekti tükeldamine
   3.1 Etappideks jaotamine: meetodid, mille abil projekti eesmärk püütakse saavutada
   3.2 Etapid: näiteks spetsifitseerimine, projekteerimine jne.
4. Projekti organiseerimine: spetsifitseeritakse projektorganisatsiooni osad, nende vastutus ja ülesanded.
5. Liidesed projektist väljaspoole: millistesse teistesse projektidesse või organisatsioonidessse see projekt kuulub ja mis moodi.
6. Fikseeringute ja muutuste käsitlus: kuidas etapitasemed fikseeritakse ja kuidas juhitakse ülesannete muudatuste ettepanekuid/muudatusi.
7. Dokumenteerimise plaan: projekti dokumenteerimistööd: mis dokumendid, millal, mil viisil, kelle jaoks, kuidas arhiveeritakse jne.
8. Kvaliteediplaan: kuidas kvaliteedi tagamine projektis tagatakse: meetmed, tööriistad, inspektsioonid vms.
9. Riskide juhtimise plaan: projekti riskid tähtsuse järjekorras, nende kirjeldus, mil viisil nende eest hoiduda, kuidas neid adresseerida.

10. Koolitusplaan: kuidas projektiga seotud koolitus organiseeritakse: mida, keda, kus, millal, kes jne.
11. Juhtimisplaan: projekti edenemise jälgimine, aruandlus, juhtimismeetodid.
12. Informeerimisplaan: kuidas organiseeritakse projektikohane informeerimine – kellele, mida, kuidas, millal.
13. Kättetoimetamisplaan: kuidas, millal, kuhu jne.

Lisad: Ajagraafik (kogu projekt, osaprojektid)
Ülesannete loend ja töömäärahinnangud
Eelarve ja kuluarvutused
Dokumentide loetelu
Inimeste loetelu

Versiooninumber:
Kuupäev:
Kommentaarid:

* Projektiplaani soovitusliku sisu koostamisel on aluseks võetud A. Kalja "Tarkvara projektijuhtimine" loengumaterjalides olev näide. [6]

Lisa 4: Muudatusettepaneku näidis

MUUDATUSETTEPANEK

Projekt:_____
Muudatuse nr: _____

Täidab muudatuse esitaja:
Muudatuse esitaja:_____
Kuupäev:_____
Muutusesitus ja muutuse põhjused:

_____
_____
_____
_____
_____
_____
_____

Täidab projektijuht:

<table>
<tr><td colspan="2">Muutuse hinnang:</td></tr>
<tr><td>Mõjud Projektiplaanile:
Kuludele: +/- _____ Krooni
Töömääradele: +/- _____ Tundi
Ajagraafikule: +/- _____ Päeva</td><td>Kiirus:
___ Kiire
___ Teha, kui jõutakse
___ Muu: _____</td></tr>
<tr><td colspan="2">Muutus mõjutab järgnevaid</td></tr>
<tr><td>Dokumente:
_____
_____
_____</td><td>Mooduleid/programme:
_____
_____
_____</td></tr>
<tr><td colspan="2">Muud kommentaarid ja PJ heakskiit muutuse otsustamiseks:
_____
_____
_____</td></tr>
</table>

<table>
<tr><td>Otsus, kuupäev: _____
___ Ei teostata
___ Teostada nüüd
___ Teostada järgmises redaktsioonis
___ Muu: _____
_____
_____
_____</td><td>Muutuste mõjud uuendatud:
___ Ajagraafikusse
___ Töömääradesse
___ Eelarvesse</td><td>Muudatusettepaneku seis:
___ Vastuvõetud
___ Hinnatud
___ Käsitletud
___ Ei teostata
___ Ootab teostust
___ Teostub
___ Muu: _____
_____</td></tr>
</table>

* Muudatusettepaneku näidise koostamisel on aluseks võetud A. Kalja “Tarkvara projektijuhtimine” loengumaterjalides olev näide. [6]

Lisa 5: Töös kasutatud akronüümid

CMM – Capability Maturity Model

COCOMO – COnstructive COst MOdel

FPA – Function Point Analysis

ISO – International Organization for Standardization

KMeh – Kontrollmehhanism

KPA – Key Process Area

PSP – Personal Software Process

ROI – Return on Investment

SCE – Software Capability Evaluation

SCM – Software Configuration Management

SEI – Software Engineering Institute

SLOC – Source Lines of Code

SPI – Software Process Improvement

SPICE – Software Process Improvement and Capability dEtermination

SQA – Software Quality Assurance

SW-CMM – Software Capability Maturity Model / Capability Maturity Model for Software

Lisa 6: Online-intervjuu näidis

Intervjuud toimusid kõik täiesti vabas vormis. Mul endal olid esialgu ainult märksõnad, mille kohta uurida. Eesmärgiks oli siiski lasta vastajal rääkida omasoodu, mina esitasin suunavaid ja täpsustavaid küsimusi. Küsitletava nimi on asendatud nimega "küsitletav", teatud faktid firma kohta on asendatud XXX-iga. Allolev online-intervjuu viidi läbi MS Messenger keskkonnas. Antud online-intervjuu sai välja valitud, kuna see on erakordselt halb näide sellest, kuidas üks projekt võib toimida.

.--.
| Session Start: 7 May 2003
| Participants:
| sander (san2a@hotmail.com)
| küsitletav (xxx@xxx.xxx)
.--.
[17:14:40] sander: meili lugesid? tutvusid reegliteda?:) tere ka
[17:15:22] küsitletav: heh ... lugesin jah ... juba mitu tundi tagasi ...
aga ma pidin ära käima - ema juures :) ... just tulingi
siia et sa ka vast siin :)
[17:15:27] küsitletav: ja tre kah ;)
[17:15:41] sander: nii et las käia siis:)
[17:15:49] sander: räägi kõik välja:)
[17:16:19] küsitletav: lasen siinsama jutuka aknas vabas vormis ?
[17:16:23] sander: jep
[17:16:31] küsitletav: ok ...
[17:16:36] sander: võid muidugi meili ka saata kui tahad
[17:17:02] küsitletav: heh ... üritan vast vabas vormis :)
[17:17:20] sander: teeme nii
[17:17:25] küsitletav: niii ... esmalt siis natukene taustast - firma ja
struktuurid ...
[17:18:02] küsitletav: firma on suht koht pisike ja ei ole hetkel veel
keskendunud otseselt tarkvara tootmisele - kuigi plaanide
kohaselt on see sihiks võetud
[17:19:04] küsitletav: alguse sai kogu asi siiski ühest suuremast projektist

- web'i lehest - ja kõik firma programmeerijad on siiani tegelenud põhiliselt saidi enda ning selle abivahendite (tausta-apliktsioonide) loomisega

[17:20:16] küsitletav: projekti-stiilist tootmine meil nagu toimiks... - selline süsteem on meil parasjagu arengu-järgus ... ehk siis üritatakse edasi minna täielikult projekti põhiseks

[17:20:33] küsitletav: niisiis oskan ma projektidest kui sellistest ka suht vähe rääkida :(

[17:20:54] küsitletav: ok ... nüüd siis probleemid ...

[17:20:59] sander: no planeerimise poolest, change management, requirements management, problems

[17:21:19] küsitletav: brrr ...

[17:21:21] küsitletav: ?

[17:21:50] sander: ehk muutuste haldamine, nõuete haldamine

[17:21:58] küsitletav: heh ...

[17:22:39] küsitletav: suht raske sellele niimoodi läheneda ... kui tegemist ei ole norm projekti-põhise arendusega siis ei saa ka sellistele küsimustele eriti otseselt vastata :(

[17:22:49] sander: oki

[17:22:56] sander: no pane siis vabas vormis edasi:)

[17:24:29] küsitletav: ok ... mismoodi meil tekib niiöelda "projekt" : XXX (ehk siis pekontoris) tekib firmajuhil/juht-grupil idee midagi teha ... see pannakse suht koht üldisesse sõnastusse - aga ei midagi täpsemat ... ehk on siis ainult teada et midagi sellist tahetakse

[17:25:58] küsitletav: see info saabub siis eestisse kus meie (progejad) peame seda kuskilt otsast järama hakkama ... mõtlema välja asja täpsema loogika jne ... vahepeal üritame seda XXX poolega nö kooskõlastada kuid see toimub tavaliselt läbi eesti ülemuste - ja ei anna just palju tulemusi :(

[17:26:36] sander: ülemused ise nagu selle arenduse tehnilise poole pealt midagi jagavad ka?

[17:26:51] sander: on tegu visionääridega?

[17:26:59] küsitletav: seejärel teeme me valmis mingid versioonid asjast ...

ehk siis nö oma nägemuse ... väga täpseks üritame mitte minna kuna see võib kaasa tuua liiga suured ajakulud kui XXXis siiski leitakse et see ei ole see mida nad tahtsid ...

[17:27:04] küsitletav: eip ...

[17:27:10] küsitletav: nad ei jaga eriti miskit ..

[17:27:16] küsitletav: ainult majaduse osa ...

[17:27:41] küsitletav: aga web-ist ja sealsetest "headest tavadest" ei tea nad eriti midagi :(

[17:28:40] küsitletav: ok ... edasi ... kui XXXidele asi siiki meeldib siis me teeme projekti lõpuni ... ning viimane faas on jälle see et nemad vaatavad üle et kas siis sobib või ei ...

[17:29:12] sander: kuidas koordineerimisega lood on?

[17:30:31] küsitletav: "projekt" kuulutatakse lõpetatuks ... mida ta tegelikult aga kunagi ei ole ... sest ülemused muudavad suht tihti meelt ja tahavad suht palju muudatusi ... kusjuures nende tahtmised pole ka kunagi kindlad ... üks päev ütlevad et muutke ära see ja see ... siis läheb 2 nädalat mööda ja siis kästakse kõik need asjad tagasi muuta ... ehk varukoopiate alles-hoid on suht koht oluline ...

[17:31:14] küsitletav: kehv :(... eesti poolel on tegelikult ainult 2 inimest kes peaksid asja juhtima - progejate ülemus ja saidi kui sellise "projekti-juht" ...

[17:32:14] sander: kuid...?

[17:32:17] küsitletav: reaalsus on nii et töid annab enamasti projekti-juht ... otsene ülemus aga ainult kontrollib aeg-ajalt töö kulgu ... ok ok ... vahest annab töid ka teha ... ja noh .. temale peame ka raporteerima asjade seisust ...

[17:32:53] küsitletav: ehk organiseerimine on peaaegu null ... tihti on nii et keegi ei tea tegelikkuses mida keegi teeb ...

[17:32:54] küsitletav: :)

[17:33:01] sander: on teil mingi kindel raporteerimise kord, vorm või on see nii et kui ta soovi avaldab?

[17:33:05] küsitletav: ja see ei ole just mitte eriti meeldiv ...
[17:33:55] küsitletav: eip ... meil kehtestati suht ammu nö nädala-aruanne ... kus on siis kirjas eelmise nädala tegevused (planeeritud aeg/ tegelik aeg) ning tuleva nädala plaan ...
[17:34:11] sander: kas nendest kinni ka peetakse?
[17:34:19] küsitletav: sellise aruande peab esitama iga progeja esmaspäeviti ...
[17:35:07] küsitletav: noh ... jah ... aga need ei ole väga informatiivsed ... vahest peaks palju rohkem kirjutama ... vahest vähem ... ja noh ... ajaliselt jäädakse sellega ka tihti hiljaks ...
[17:35:57] sander: mingeid "karistusi" ka selle eest on ette nähtud?
[17:36:04] küsitletav: ahjaa ... aja planeerimisest ka niipalju et kui mingi ülessanne antakse siis progeja peab umb-kaudu ütlema et kaua sellega aega võiks minna ...
[17:36:42] sander: kas progeja on ainus kes seda hindab või veel keegi?
[17:38:09] küsitletav: no eks ülemus ikka ka umb-kaudu hindab aja-kulu ....
[17:38:23] küsitletav: aga eks see on suht raske kui ei tea asja spetsiifilist külge ...
[17:39:01] sander: kas te ajalooandmestikku ka kogute?
[17:39:15] küsitletav: mismõttes ?
[17:40:07] sander: möödunud tegevuste kohta et hiljem oleks lihtsam hinnanguid anda näiteks töömäärade osas
[17:40:41] küsitletav: noh ... ega vist mitte eriti ...
[17:40:52] küsitletav: ma hetkel suitsetan .. ja siis jätkan ... :)
[17:41:09] sander: ok
[17:43:22] küsitletav: niii ... tagasi ...
[17:43:40] sander: jätkame siis
[17:43:52] küsitletav: mul nüüd max tund aega su küsimustele vastata siis ma pean elamist koristama hakkama :) ... naine annab muidu peksa :)
[17:44:14] sander: no see on seda väärt

[17:44:51] sander: kuidas teil firmasisese kommunikatsiooniga on?

[17:45:09] küsitletav: mingit erilist satistikat meil ei tehta ... aruandluse põhjal ... ja mis eriti vastik (minu arust) on see et ka kirjutatud kood ei ole selline mida hiljem "taas-kasutataks" :(

[17:45:51] küsitletav: firmasisene kommunikatsioon on üks nõrgemaid lülisid kogu asja juures ... ehk ka see punkt kust enamus teised probleemid alguse saavad ...

[17:46:10] küsitletav: ok ... veel natukene üldist ... ehk siis suurem osa negatiivseid asju :

[17:46:46] küsitletav: meil EI ole reeglistikke - kindlaid reegleid kuidas koodi kirjutada , kuidas seda kommenteerida ja dokumenteerida

[17:47:05] sander: see on suur miinus

[17:47:57] küsitletav: koode EI kirjutata viisil mis võimaldaks nende korduv-kasutamist ... seda üritavad teha üksikud progejad (mina kaasa arvatud) ... kuid ajapikku hakkab see entusjasm üle minema :(

[17:48:24] küsitletav: kui on mingi ülessanne siis kogu selle kood on põhimõtteliselt kirjutatud nullist ...

[17:48:57] küsitletav: kuigi loogiline oleks nii et on mingi koodivaramu - valmis kirjutatud osad mingite toimingute tegemiseks ...

[17:48:58] sander: kas sa ei arva et teil oleks tulevikus lihtsam?

[17:51:04] küsitletav: arvan küll ... aga selleks et sellest kasu oleks peab rohkem kui üks inimene pingutama :(

[17:51:20] sander: kuidas automatiseeritusega lood on?

[17:51:39] sander: riskianalüüsi teete?

[17:51:54] küsitletav: nope ... see on meil nö tundmatu suurus :)

[17:52:20] sander: :)

[17:52:52] sander: mai eksi üldse kui ütlen et nii CMM'i kui ka SPICE'i skaalal olete te esialgsel tasemel

[17:53:14] küsitletav: tegelikult ongi jama selles et meie firmas ei vasta

vist üksi asi sellele mida koolis õpetatakse ...
[17:53:18] sander: kuidas ressursside arvestamine käib?
[17:54:02] küsitletav: heh ... peaks vist jälle ütlema et "mis asi ?" ;)
[17:54:09] sander: :) cool
[17:54:54] sander: no üleüldse kas ta arvestate palju mingi töö jaoks raha, aega, muid ressursse kulub? Milline on tootliku ja mittetootliku töö suhe?
[17:55:06] sander: aega arvestate tuli meelde
[17:55:11] sander: aga muud ressursid?
[17:55:38] sander: ja kas need paika peavad, need arvestused? kui tihti te need hinnangud üle vaatate, kui üldse?
[17:55:56] küsitletav: ok ... võtame siis lihtsalt ... kui loomulik asjade käik on nii et tuleb idee -> analüüs -> disain+dokumentatsioon -> progemine -> produkt -> hooldus ... siis meil on see nö üks suur kera kust ei saa ühtegi eraldi osa välja tuua ...
[17:56:33] sander: ehk siis kõik toimub väga kõrgem tasemel nii öelda
[17:56:51] küsitletav: irw ... jah ... miinus-märgiga ...
[17:56:52] küsitletav: :)
[17:57:14] sander: selles mõttes kõrgel tasemel et asi ei lähe väga detailseks
[17:57:33] sander: allhankijatega teil pole tegemist?
[17:57:37] küsitletav: totaalne anarhia ... aga noh ... kui aus olla siis eks päris kooli-systeemide järgi ei toimi ükski firma ..
[17:57:41] sander: ise ehk olete selles rollis olnud?
[17:58:07] küsitletav: ei ... nagu ennist ütlesin on siiani tegu olnud firma-sisese arendusega ...
[17:58:08] sander: mai räägi siin koolisüsteemidest, mulle neid asju koolis ei õpetata, ikka omapead tegelen sellega suures osas
[17:58:28] sander: kuidas kvaliteedi tagamine toimub?
[17:58:45] küsitletav: aga noh ... plaanitakse asju nö normaliseerima hakata ja sealt edasi siis ka väljapoole töid tegema hakata ...
[17:59:36] küsitletav: heh ... aga enamus terminid on sellised mis lähtuvad

nö korrektsest struktuurist ... ja kuna meil ei ole
struktuuri siis on nendele vastmine pea võimatu ... :(
[17:59:56] sander: no kvaliteet on nii üleüldine termin ju
[18:00:02] sander: või te ei tunnista seda?:)
[18:00:04] küsitletav: kvaliteedi tagamine on nö progeja oma asi ... kui kehvasti teeb siis on tal endal hiljem tööd ...
[18:00:32] sander: kas progeja testib oma tööd ainuisikuliselt või veel keegi?
[18:03:21] küsitletav: enamasti küll ... kuigi vahest vaatab mõni teine ka asja üle ... ise nagu kõiki vigu üles ei leia :(
[18:05:11] sander: see on hea näitaja. saate 1 plussi selle eest:)
[18:07:11] sander: kuidas teil koosolekutega lood on?
[18:07:47] küsitletav: on vahest ... aga nende tulemused on suht väikesed ..
[18:08:09] küsitletav: no parim näide on see et pole suudetud paika panna reegleid ...
[18:08:45] sander: päevakorda pole?
[18:08:50] sander: protokollitakse?
[18:11:00] küsitletav: no ok ... veel üks hea näide sellest kui segased meil need asjad on : web-i saidi juures (nagu meie projekt on) peaks üks suht koht oluline punkt olema see et mis tarkvara me toetame ... ehk mis browseritega meie sait töötab ... ja seda ei ole ära määratud siiani .. kuigi küsimus on üles kerkinud korduvalt :(
[18:11:11] küsitletav: ja päevakava meil küll pole
[18:12:03] küsitletav: koosoleks ise toimub siis kui on mingi probleem ... ja noh .. protokollitakse vast minimaalselt ...
[18:12:34] sander: kas progejad vähemalt üritavad teha asja nii et sait toimiks kõigi levinumate brauseritega?
[18:13:18] küsitletav: mina näen sellega hullu vaeva ... aga ma olen ka vist ainus ... paar tegijat olen veel suutnud mõtlema panna ...
[18:14:03] sander: hea seegi
[18:14:26] sander: kui suur teil muidu ümbertegemiste maht protsentuaalselt võiks olla?

[18:14:42] küsitletav: aga krt .. närvi ajab kui mina näen päevi aega sellega et teha lahendus selliseks et ta iga browseriga töötaks ... ja samas on saidi teine lehekülg selline mis töötab näiteks ainult IE-ga ... kusjuures on tegemist üli-lihtsa lehega ...

[18:14:48] küsitletav: hmmm ...

[18:15:32] küsitletav: suht suur ma arvan ...

[18:16:34] sander: kui nüüd etappide pikksed paika panna protsentuaalselt siis kui pikad nad oleksid?

[18:16:48] küsitletav: kuna meil ette planeerimist eriti ei ole ja ka dokumentatsioon kirjutatakse tagant-järgi siis ... noh ... palju igatahes ... ja sellele lisandub veel see faktor et tellimus muutub pidevalt ...

[18:17:08] küsitletav: heh ... mile-stoned ?

[18:17:23] sander: jep

[18:17:45] küsitletav: ei ole selliseid asju ...

[18:17:59] küsitletav: peaks nagu eelneva jutu põhjal selge olema :D

[18:19:20] küsitletav: seal töötamisel on ainult paar head külge : saad kokku puutuda tarkvara arenduse kõikide faasidega (ehk PEAD kokku puutuma) ... saad kasutada oma loovust ... aga sellega loetelu vist lõppebki :(

[18:19:47] küsitletav: vut ...

[18:20:03] sander: palju pappi saab?:)

[18:20:05] küsitletav: <suitsule>

[18:20:20] sander: no mine tossa siis vahepeal jälle:)

[18:20:24] küsitletav: heh ... minu jaoks normaalselt ...

[18:23:23] küsitletav: tagasi ...

[18:24:12] küsitletav: ok .. veel üks he point : UML on meie firmas tundmatu suurus ....

[18:24:18] küsitletav: he==hea :)

[18:24:22] sander: jätka vabas vormis kui soovid, kuid ma arvan et ma olen suuremas osas juba targemaks saanud.

[18:24:31] küsitletav: heh ...

[18:24:36] sander: mida te UMLi asemel kasutate?

[18:24:43] küsitletav: ja said siis miskit oma tarbeks välja noppida ?
[18:24:44] sander: automatiseeritus oli 1 küsimus
[18:25:05] küsitletav: UML-i asemel ? ... heh ... mälu ...
[18:25:10] küsitletav: irw ;)
[18:25:33] sander: :)
[18:26:33] sander: automatiseeritus puudub?
[18:26:39] sander: mis programme kasutate
[18:26:43] sander: ?
[18:27:00] küsitletav: hmmm ... ok ... ja siis natukene seletust et mix ja millest ... need probleemid tulenevad : kõik töötajad on kohapeal koolitatud ... ehk ei ole ühtegi kellel oleks nö kooli-haridust antud valdkonnas ...
[18:27:18] küsitletav: ja kohapealne koolitus on suht koht vilets :(
[18:27:39] küsitletav: programme ? ... või keeli ?
[18:28:00] sander: programme just
[18:28:09] sander: arenduskeskkond
[18:28:27] küsitletav: koolitusest veel niipalju et mina olen oma oskused nagu suuremalt jaolt ise õppinud seal töötamise ajal ...
[18:28:35] küsitletav: visual-stuudio 6
[18:28:41] küsitletav: programmide jaoks ...
[18:28:53] küsitletav: ja noh ... ASP-i saab kirjutada milles iganes ...
[18:29:03] küsitletav: ja javascripti samuti ...
[18:29:53] küsitletav: ja SQL tuleb MS SQL Server 2000 abi-programmidest - Query Analyser ja Enterprice Managerist
[18:31:17] sander: nüüd mõned faktid veel: mitu inimest kokku on firmas
[18:33:27] sander: .net'i pole plaanis kasutama hakata?
[18:34:43] küsitletav: on .. aga kauges tulevikus ...
[18:35:07] sander: inimesi kokku 4-5?
[18:35:16] küsitletav: see ei ole minu arust küll veel kasutus-kõlblik ... ressursi nõudlikus on meeletu ... ehk siis pointless :)
[18:35:29] küsitletav: progejaid sa mõtled ?
[18:37:34] küsitletav: ametlikult hetkel peaks olema 17 progejat ... heh ... mis peaks jällegi selgitama kui segased meil asjad on ...

kui selline seltskond kõik ilma reegliteta tööd teeb ... ja
koodi kirjutab ... üks ei saa teise koodist aru ... krt ...
ok ... see ajab mind tavaliselt närvi :)

[18:38:07] sander: :)

[18:38:18] sander: 17 progejat ja üldse palju inimesi on?

[18:42:46] küsitletav: krt ... seda ma küll peast ei tea ... meil on palju
lao töötajaid kellega ma üldse kokku ei puutu ... ja XXX
osa suurusest ma ka ei tea ... pluss mingi rahvas
XXXis ... plah tuut ... ja veel majandus inimesed jne
jne ... nii et jään vastuse ses osas võlgu ...

[18:43:03] sander: ainult eestis!

[18:46:57] küsitletav: hmmm ... ikkagi ei tea ...

[18:47:07] küsitletav: miski 50 ringis vast ...

[18:47:41] sander: arendajate osas ma arvan te olete nii eesti
keskmised umbes (koguse osas):-)

[18:49:50] sander: ma arvan et olen nüüd igatahes targemaks
saanud ja sina oma kohvi ka välja teeninud :-)

[18:50:11] küsitletav: heh ... eks ma siis mingi aeg nõuan sisse :)

[18:50:31] sander: teeme nii

[18:50:35] sander: tänud sulle!!!:)

[18:50:50] sander: üritage neid asju seal ikka korda saada

[18:51:07] sander: praegu olete eesti keskmised:) kaootilisuse
osas

[18:51:47] küsitletav: heh :)

[18:52:07] küsitletav: jah ... eks me üritame parandada olukorda ...

[18:53:02] küsitletav: ma hetkel ise üks nn projekti-juhtide teamis ...
meist peaks siis saama projekti juhid - kui kõiki korda
läheb .. siis vast tulevad projektid ka ja kogu muud
paberi-majandus ja korrad sinna juurde :)

[18:54:20] sander: ma siis konsulteerin:)

[19:02:58] küsitletav: heh ... kle ok ... ma siis nüüd koristama :)

[19:03:03] küsitletav: paih ...

Lisa 7: 14 tarkvaraarendusega tegeleva ettevõtte osas läbi viidud uurimuse tulemusi

Tabel 3 toob välja osa uurimuse käigus välja tulnud levinumaid probleeme ja nende esinemiste arvu erinevate firmade lõikes. Olgugi et mõni allolevatest probleemidest on esinenud ainult ühes või kahes firmas, on need siiski välja toodud nende unikaalsuse, omapärasuse, erakordsuse või kriitilisuse tõttu.

Probleem	**Probleemi esinemise arv erinevate firmade osas**
Hinnangute tegemine ajagraafiku, maksumuse, töömäärade ja tarkvara suuruse osas	14
Hinnangud ebatäpsed	14
Hinnangute tegemisel ei kasutata vastavat tarkvara	10
Hinnangute tegemisel ei osale töö tegija ise	5
Hinnanguid ei täpsustata projekti käigus	7
Dokumentatsiooni vähesus	12
Dokumentatsiooni ei uuendata regulaarselt	12
Projekti kulgu ei jälgita piisavalt	12
Riskianalüüs puudub	13
Riskianalüüs pealiskaudne	1
Riskide loetelu ei vaadata projekti jooksul üle	9
Üks pikk realisatsioonietapp	10
Koordineerimatus	9
Komponentide/koodi kokkusobimatus	5
Suur ümbertegemiste maht	8
Muudatustele reageerimine	11
Vähene automatiseeritus	10
Korraliku koolituse puudumine	7
Tarkvara kvaliteet ei vasta püstitatud eesmärkidele	7
Muudatuste haldamine	12

Süstemaatilist ajalooandmestiku kogumist ei toimu	13
Tunnustust ei jagata igale projektiliikmele eraldi	13
Allhankijate tööde seisu ei jälgita piisavalt	3
Kodeerimise standard puudub	10
Koodi/komponentide korduvkasutamist ei toimu	6
Versioonimine	3
"Loll" müügimees	2
Projektijuht on samas ka müügijuht	2
Projektijuht on samas ka programmeerija/disainer vms	8
Projektijuhil on korraga liiga palju projekte käsil	3
Rollid ja kohustused pole täpselt määratletud	7
Mittetootliku töö liiga suur osakaal	4
Kvaliteedi tagamine piirdub tarkvara testimisega	9
Projektialase informatsiooni vähesus	6
Visiooni puudumine või pealiskaudsus	8
Oma koodi testib ainult programmeerija ise	7
Projektijuht ei ole tarkvara alal spetsialist	9
"Raudne eesriie"	1
Klient ei ole arendusprotsessi piisavalt kaasatud	4
Nõuded loodavale tarkvarale ei ole grupeeritud kriitilisuse osas	11
Dokumendivormid puuduvad	12
"Tulekahjude kustutamine"	14
Tööpuudus	2
Tööd on liiga palju	9
Töö on igav, tüütu, üksluine	8

Tabel 4: Uuringu käigus välja tulnud levinumate probleemide loetelu ja nende esinemiste arv erinevate firmade osas.

Lisa 8: Töös olevate tabelite ja jooniste loetelu

Tabelid:

Joonised:

Kasutatud kirjandus

[1] Walker Royce, “Software Project Management: a Unified Framework,” The Addison-Wesley object technology series. 1998, Addison Wesley Longman, Inc.

[2] Mark C. Paulk, Bill Curtis, Mary Beth Chrissis, Charles V. Weber, “Capability Maturity Model for Software, Version 1.1”, http://www.sei.cmu.edu/publications/documents/93.reports/93.tr.024.html, 1993, CMU/SEI.

[3] Ray Dion, “Starting the Climb Towards the CMM Level 2 Plateau”, 1995, Software Process Newsletter, No. 4, Fall 1995, IEEE Computer Society TCSE.

[4] Mark C. Paulk, “Using the Software CMM in Small Organizations”, 1999, http://www.sei.cmu.edu/cmm/papers/cmm-small.pdf, CMU/SEI.

[5] James Herbsleb, Anita Carleton, James Rozum, Jane Siegel, David Zubrow, “Benefits of CMM-Based Software Process Improvement: Initial Results”, http://www.sei.cmu.edu/publications/documents/94.reports/94.tr.013.html, 1994, CMU/SEI.

[6] Ahto Kalja, “Tarkvara projektijuhtimine: loengumaterjalid ja grupitööde tulemid”, 2003, TTÜ/KI.

[7] Steve McConnell, “Software Project Survival Guide: How to Be Sure Your First Important Project Isn't Your Last”, 1998, Microsoft Press.

[8] Alan M. Davis, “Fifteen Principles of Software Engineering”, 1994, IEEE Software, vol 11, nr 7.

[9] C. Jones, “Software Estimating Rules of Thumb”, 1996, IEEE Computer.

[10] David Zubrow, William Hayes, Jane Siegel, Dennis Goldenson, “Maturity Questionnaire”, 1994, CMU/SEI.

[11] Watts S. Humphrey, “The Personal Software Process (PSP)”, 2000, CMU/SEI.

[12] Watts S. Humphrey, “The Personal Software Process: Overview, Practice and Results”, CMU/SEI.

[13] Asko Seeba, “Unifitseeritud tarkvararendamise protsess ja selle rakendamise juhtumianalüüs”, 2001.

[14] "10 Keys to Successful Software Projects: An Executive Guide", 2000, Construx.

[15] Peeter Normak, "Projektijuhtimine: loengumaterjal", 2001, TPÜ.

[16] "Recommended Approach to Software Development, Revision 3", 1992, NASA.

[17] Mitchell J. Bassman, Frank McGarry, Rose Pajerski, "Software Measurement Guidebook, Revision 1", 1995, NASA.

[18] Kellyann Jeletic, Rose Pajerski, Cindy Brown, "Software Process Improvement Guidebook", 1996, NASA.

[19] Robert E. Park, "A Manager's Checklist for Evaluating Software Cost and Schedule Estimates", 1995, CMU/SEI.

[20] Ronald P. Higuera, Yacov Y. Haimes, "Software Risk Management", 1996, CMU/SEI.

[21] Roger L. Van Scoy, "Software Development Risk: Opportunity Not Problem", 1992, CMU/SEI.

[22] "A Guide to Project Management Body of Knowledge" (PMBOK® Guide), 2000, Project Management Institute (www.pmi.org).

[23] Watts Humphrey, "Managing the Software Process", 1989, Addison-Wesley.

[24] S. Zahran, "Software Process Improvement - Practical Guidelines for Business Success", 1998, Addison-Wesley.

[25] Lars Mathiassen, http://www.mathiassen.eci.gsu.edu/research.htm.

[26] Terttu Orci, Astrid Laryd, "Dynamic CMM for Small Organisations - Implementation Aspects", 2000, Umeå University/Stockholm University.

[27] J. B. Dreger, "Function Point Analysis", Prentice Hall.

[28] Bill Gates, "Äri @ mõttekiirusel", 2000, K-Kirjastus.

[29] E. M. Bennatan, "On Time, Within Budget: Software Projects Management Practices and Techniques", 1992, QED Publishing.

[30] S. Bouchier, I. Procter, "Software Tools for Project Management", UNICOM.

[31] F. P. Brooks Jr., "The Mythical Man-Month", 1995, Addison-Wesley.

[32] A. Davis, "Trial by Firing: Saga of a Rookie Manager", 1994, IEEE Software.

[33] C. Jones, “Patterns of Software System Failure and Success”, 1996, International Thomson.

[34] Steve McConnel, “How to Defend an Unpopular Schedule”, 1996, IEEE Software.

[35] J. J. Rakos, “Software Project Management for Small to Medium Sized Projects”, 1990, Prentice-Hall.

[36] D. Youli, “Making Software Development Visible. Effective Project Control”, 1990, Wiley.

[37] Mark C. Paulk, Charles V. Weber, Suzanne M. Garcia, Mary Beth Chrissis, Marilyn Bush, “Key Practices of the Capability Maturity Model, Version 1.1”, 1993, CMU/SEI.

[38] H. W. Smith, “The 10 Natural Laws of Successful Time and Life Management. Proven Strategies for Increased Productivity and Inner Peace”, 1994, Nicholas Brealey.

[39] K. Whitaker, “Managing Software Maniacs. Finding, Managing and Rewarding a Winning Development Team”, 1994, Wiley.

Summary

The aim of this thesis is to describe the problems that software projects are facing, find out whay those problems have arised and then provide helpful guidelines to fix those problems or prevent them from happening at all.

For the thesis to be more accurate in the context of Estonia's software businesses, I investigated some 14 firms of different sizes. Because most businesses don't really want to reveal their problems or methods of doing things, I had to use all the help I could get from my acquaintances and friends who work in the software business.

First chapter describes the Capability Maturity Model for Software: overview, purpose, structure, and in greater detail level 2 of the CMM.
Second chapter describes the benefits of software process improvement. The following chapter is the main chapter, it describes the problems that software companies are facing, and provides guidlines for fixing them or preventing them from arising at all. Last chapter presents a selection of Alan M. Davis's "Fifteen Principles of Conventional Software Engineering".