Tallinna Pedagoogikaülikool
Matemaatika-loodusteaduskond
Informaatika osakond

TARKVARA ARENDUSPROTSESSI ÜLDPÕHIMÕTTED TPÜ-s

Bakalaureusetöö

Triinu Gross

Juhendaja: Marek Kusmin

Autor:	.....	"....."	 2004
Juhendaja:	.....	"....."	 2004
Osakonna juhataja:	.....	"....."	 2004

Tallinn 2004

SISUKORD

1 SISSEJUHATUS

1.1 Teema valik

Kuna arvutite ning neil põhinevate infotehnoloogiliste lahenduste kasutamine igas eluvaldkonnas muutub üha enesestmõistetavamaks, siis suureneb järjest enam ka uute tarkvararakenduste loomise vajadus. Samasugune olukord valitseb ka Tallinna Pedagoogikaülikoolis (edaspidi TPÜ), kus järjest enam soovitakse kõikvõimalikke toiminguid paberilt arvutisse viia. Üha suurenev tarkvaraprojektide hulk on toonud kaasa vajaduse luua uusi töökohti analüütikute, programmeerijate jt. tarbeks. Kahtlemata eksisteerib TPÜ infotehnoloogia osakonnal (edaspidi IT osakond) vajadus lisatööjõu järele, kuid samas oleks kasulik vaadata üle praegune olukord ning analüüsida, kas oleks võimalik ka olemasolevat meeskonda produktiivsemalt tööle rakendada? Kuna tarkvaraprojektide arv ja ka meeskond kasvab, siis oleks mõistlik hakata välja töötama põhimõtteid ja nõudeid, mis reguleeriksid tarkvaraarenduse protsessi ning aitaksid muuta nii IT osakonna kui ka tellija tegevust tulemusrikkamaks.

Kui mõõta TPÜ tarkvara arendustegevust SW-CMM (*Capability Maturity Model for Software*)[1] skaalal, siis võib kahtluseta väita, et hetkel on see algtasemel. Enamus tegevusest toimub suhteliselt määratlematult nii tellija kui tarnija poolt ning see on toonud kaasa mitmeid probleeme. On esinenud juhtumeid, kus tarkvaraprojektid on kas osaliselt või täielikult läbi kukkunud. Kindlasti ei saa siin kogu vastutust IT osakonnale panna. Ka tellija tegevus projekti algatamisel ja selle edasisel realiseerimisel peaks olema selgelt reguleeritud. Vastasel juhul võib taas tekkida olukord, kus tellija sõna otseses mõttes annab vaid loodava tarkvararakenduse pealkirja ning astub seejärel arendusprotsessist kõrvale, lükates kogu töö ja vastutuse IT osakonnale. Seetõttu on käesolevas töös lisaks tarkvaraarendusmeeskonna tegevustele pööratud tähelepanu ka tellija kohustustele.

1.2 Eesmärk

Käesoleva töö eesmärk on kokku võetud ka infotehnoloogia standardi sissejuhatuses: "Tarkvara väljatöötamiseks ja halduseks ilmub üha enam standardeid, protseduure, meetodeid, keskkondi. Niisugune rohkus on tekitanud raskusi tarkvara halduslikul ja tehnilisel käsitsemisel, eriti toodete ja teenuste integreerimisel. Tarkvaradistsipliin peab siirduma selliselt rohkuselt mingile ühisele raamstruktuurile." [ISO-Tep]

1 SW-CMM (*Capability Maturity Model for Software*) on SEI (*Software Engineering Institute*) poolt 1993. aastaks välja töötatud mudel selleks, et hinnata tarkvara arendamise ja haldamise kvaliteeti. Nimetatud mudel eristab tarkvaraprotsessi küpsuses viit taset. [Piho2003]

Niisiis on töö eesmärgiks süstematiseerida tarkvara arendusega seotud tegevused ning moodustada esialgne pakett põhimõtetest, mis ühtlustaksid projektide elluviimisel läbitavaid etappe ning parandaksid töö efektiivsust. Ühtlasi on eesmärgiks luua taotluste jm. dokumentide vormid, mis aitavad luua ühtse dokumentatsioonisüsteemi kõikidele tarkvarapakettidele.

Kuigi algne idee oli luua kindla raamistikuga "TPÜ tarkvaraarenduse eeskiri", siis töö käigus ilmnes sellise mõtte ebasoovitavus. Pole eriti mõistlik koostada ja vastu võtta eeskirju, mis on kirjutatud pigem teooriast lähtuvalt ning mis ei pruugi reaalse olukorraga üldse sobituda. Kuna iga organisatsioon on omanäoline ja selle kujundavad osalt jaolt ka inimesed, kes selles töötavad, siis ei ole olemas ühest retsepti, mis tagaks projektide õnnestumise. Seega sai töö eesmärgiks luua esialgsed ettepanekud ja nõuanded, millest peaks edasiste tarkvaraprojektide puhul lähtuma. Seejärel on võimalik reaalsetest olukordadest ja kogemustest lähtuvalt väljapakutud süsteemi muuta, täiendada ning järk-järgult liikuda töö kvaliteedi parandamise suunas.

1.3 Töö jaotus

Töö koosneb kolmest osast. Esimene osa (2. - 4. peatükk) moodustub infotehnoloogia standardi ISO 12207 "Tarkvara elutsükli protsessid" ja kahe tarkvaraarendusmetoodika - XP (*Extreme Programming*) ja RUP (*Rational Unified Process*) - kirjeldusest ja analüüsist. Tuginedes varasematele kogemustele on püütud analüüsida kolme eelpoolnimetatu kasutamise võimalikkust (ja võimatust) TPÜ tarkvaraarendusprotsessides. Kuna töö eesmärgiks on koostada kogu tarkvara elutsüklit puudutavad nõuded, siis algatusprotsessi ja tarkvaratoote mahavõttu puudutavad tegevused saab koostada tuginedes ainult standardile, kuna XP ja RUP neid valdkondi väga üksikasjalikult ei puuduta. Samas puudutab standard lisaks muule ka haldus- ja juhtimisprotsesse, mida töös väga põgusalt käsitletakse, kuna peatähelepanu on koondatud siiski tarkvaratootele. Nii standard, XP kui ka RUP toonitavad mitmel pool, et nende poolt esitatud põhimõtteid ei ole mõistlik täht-tähelt järgida. Tarkvara tootva organisatsiooni eesmärgiks peaks olema teha selgeks, millised nõuded on vajalikud ja kasulikud ning ainult neid kohandada tarkvaraarendusprotsessi raamidesse. Eesmärgiks peab olema tarkvara kvaliteedi parandamine mitte ühe või teise metoodika juurutamine. Töö teises osas (5. peatükk) on esitatud põhimõtete pakett, millest peaks edaspidi lähtuma TPÜ tarkvaraarendusprojektide elluviimisel. Kolmanda osa (Lisa. 2 - Lisa. 6) tööst moodustavad põhimõtteid täiendavad normdokumendid ja vormide blanketid.

2 INFOTEHNOLOOGIA STANDARD ISO 12207 - "Tarkvara elutsükli protsessid"

Rahvusvaheline standard ISO 12207 koostati ISO/IEC[2] tehnilise ühendkomitee (JTC 1 - "*Infotehnoloogia*") alamkomitee (SC 7 - "*Tarkvaratehnika*") poolt ning avaldati 1995. aastal. Eesti standardina kinnitas Standardiamet selle 1998. aastal. Standardi eesmärgiks on anda selge ja struktureeritud ülevaade kogu tarkvara elutsüklist ning kirjeldada kõikide protsesside tegevused: "Käesolev standard kehtestab tarkvara elutsükli protsesside tarbeks üldise, täpselt määratletud terminoloogiaga raamstruktuuri, millele saab viidata tarkvara valdkonnas. See struktuur sisaldab protsesse, tegevusi ja töid, mida tuleb rakendada tarkvara sisaldava süsteemi, iseseisva tarkvaratoote või tarkvarateenuse hankimisel ning tarkvaratoote tarnimisel, väljatöötamisel, ekspluateerimisel ja hooldamisel." [ISO-Tep]

Kuna standard on suhteliselt üldine ja suureplaaniline dokument, siis ei anna see ette dokumentatsiooni blankette, konkreetset dokumentide sisu ja muud taolist - selle peab välja töötama tarkvaraarendaja ise. Samuti ei ole toodud konkreetseid ettepanekuid tarkvara elutsükli mudeli või arendusmeetodi kohta: "Standard ei kirjuta ette konkreetset elutsükli mudelit ega tarkvara arendusmeetodit. Standardit järgivate huvipoolte vastutusele jääb elutsükli mudeli valimine tarkvaraprojekti tarbeks ning standardi protsesside, tegevuste ja tööde kajastamine selles mudelis. Huvipoolte vastutusele jääb ka tarkvara arendusmeetodite valimine ja rakendamine ning konkreetse tarkvaraprojekti jaoks kohaste tegevuste ja tööde sooritamine." [ISO-Tep]

Standard jaotab tarkvaraarendusega seotud tegevused erineva astme protsessideks:

- viis primaarprotsessi (hankimisprotsess, tarneprotsess, arendusprotsess, ekspluatatsiooniprotsess, hooldusprotsess)
- kaheksa abiprotsessi (dokumenteerimisprotsess, konfiguratsiooni halduse protsess, kvaliteedi tagamise protsess, verifitseerimisprotsess, valideerimisprotsess, ühisläbivaatuse protsess, auditeerimisprotsess, probleemide lahendamise protsess)
- neli organisatsioonilist protsessi (haldusprotsess, infrastruktuuri protsess, täiustusprotsess, koolitusprotsess)

Korduvalt toonitatakse asjaolu, et kõiki toodud protsesse pole kohustuslik rakendada, kui see pole mõistlik ja otstarbekas. Organisatsiooni nõudmistest ja vajadustest lähtuvalt tehakse

2 ISO (*the International Organization for Standardization - Rahvusvaheline Standardiorganisatsioon*) ja IEC (*the International Electrotechnical Commission - Rahvusvaheline Elektrotehnikakomisjon*) moodustavad spetsialiseeritud ülemaailmse standardimise süsteemi, mille rahvuslikud liikmesorganisatsioonid osalevad rahvusvaheliste standardite väljatöötamises. [ISO-Tep]

analüüs ning vastavalt sellele kohandatakse kehtivat standardit - osa nõudmistest jäetakse välja, osa täiendatakse jne.

<table>
<tr><th colspan="2">ELUTSÜKLI
PRIMAARPROTSESSID</th><th>ELUTSÜKLI
ABIPROTSESSID</th></tr>
<tr><td colspan="2" rowspan="2">1. Hankimine</td><td>1. Dokumenteerimine</td></tr>
<tr><td>2. Konfiguratsiooni haldus</td></tr>
<tr><td colspan="2" rowspan="2">2. Tarnimine</td><td>3. Kvaliteedi tagamine</td></tr>
<tr><td>4. Verifitseerimine</td></tr>
<tr><td rowspan="4">3. Arendus</td><td rowspan="2">4. Ekspluatatsioon</td><td>5. Valideerimine</td></tr>
<tr><td>6. Ühisläbivaatus</td></tr>
<tr><td rowspan="2">5. Hooldus</td><td>7. Auditeerimine</td></tr>
<tr><td>8. Probleemide lahendamine</td></tr>
<tr><td colspan="3">ELUTSÜKLI ORGANISATSIOONILISED PROTSESSID</td></tr>
<tr><td colspan="2">1. Haldus</td><td>2. Infrastruktuur</td></tr>
<tr><td colspan="2">3. Täiustamine</td><td>4. Koolitus</td></tr>
</table>

Tabel 1. Infotehnoloogia standardi ISO 12207 "Tarkvara elutsükli protsessid"struktuur [ISO-Tep]

2.1 Primaarprotsessid

Standardi järgi on primaarprotsessid need, mis teevad üldse võimalikuks tarkvaraarenduse. Nagu protsesside nimetustest võib järeldada, on tegemist tellija ja tarnija pädevusse kuuluvate toimingutega. Primaarprotsessid on kõige tähtsamad ja siin ei ole võimalik ühtegi protsessidest välja jätta - tarkvaraarendus peab kõik need läbima. Kõige tähtsamad on kahtlemata hankimis- ja arendusprotsess. Mõne aja möödudes omandab suurema tähtsuse hooldusprotsess, mille olulisim osa on tarkvarasse muudatuste tegemise protsessi kirjeldus.

2.1.1 Hankimisprotsess

Kogu tarkvaraarendusprotsess saab alguse süsteemi, tarkvaratoote või tarkvarateenuse hankimise vajaduse määratlemisest. [ISO-Tep] TPÜ-s peaks nimetatud toimingu läbi viima

struktuuriüksus(ed), kellel on ilmnenud vajadus uue tarkvararakenduse järele. Suur probleem, mis on TPÜ-s juba mitmeid kordi ilmnenud, on see, et paljude infotehnoloogia-alaselt ebapädevate inimeste seas levib arusaam nagu IT lahendus muudaks kõike paremuse poole. Arvatakse, et kui tarkvararakendus saab valmis, siis ongi probleemid lahenenud. Reaalsus näeb paraku pisut teistmoodi välja. Hästi toimivat süsteemi ei looda ainult ühe tarkvararakenduse abil. Kogu toimemehhanism peab juba eelnevalt eksisteerima. Loodav tarkvara aitab seda vaid automatiseerida s. t. süsteemiga seotud tegevusi hõlbustada. Seega on IT osakonnal kujunenud välja kindel soov, et tellija peab endale (ja ka ülikooli juhtkonnale) väga konkreetselt selgeks tegema, miks üht või teist rakendust üldse vaja on. Teiseks tähtsaks sammuks on koostada ja kinnitada normdokumendid, mis aitaksid reguleerida süsteemi toimimist ning motiveeriksid töötajaid (või ka üliõpilasi) rakendust kasutama.

Standard kirjeldab hankija esmaseid tegevusi järgmiselt: "5.1.1.1 Hankija alustab hankimisprotsessi süsteemi, tarkvaratoote või tarkvarateenuse hankimise, väljatöötamise või täiustamise kontseptsiooni või vajaduse kirjeldamisega. 5.1.1.2 Hankija peab püüdma määratleda ja analüüsida nõudeid süsteemile. Süsteeminõuded peaksid sisaldama tööalaseid, organisatsioonilisi ja kasutajakeskseid ning [...] muid kriitilisusnõudeid [...]." [ISO-Tep]. Standardis märgitakse, et hankija võib jätta tarkvaranõuete määratlemise ka tarnija hooleks, kuid selles osas on TPÜ-s juba mitmeid negatiivseid kogemusi: tellija nõustub projekti algfaasis IT osakonna poolt väljapakutud lahenduste ja ideedega, kuid hiljem leiab, et valminud tarkvara ei vasta (kasutaja) nõuetele ning kõik tuleb ümber teha. Niisiis on hilisemate arusaamatuste tekkimise vältimiseks parem, kui esialgsed nõuded ja üldise süsteemi toimemehhanismi formuleerib projekti algataja ise.

Standard toob välja viis erinevat hankevarianti: valmistarkvaratoote ostmine, tarkvara tootmine organisatsiooni enese poolt (käesoleval juhul niisiis IT osakonna poolt), tarkvarateenuse hankimine lepingu kaudu, olemasoleva tarkvaratoote täiustamine ning viimaseks kolme esimese variandi omavahel kombineerimine. Standard ütleb et ka hankevariandi otsuse peaks langetama hankija, kuid siinkohal võib kogemustele tuginedes väita, et pigem peaks olema otsustajaks IT osakonna vastutavad isikud ning ülikooli juhtkond, kes teavad paremini ülikooli ressursse ja võimalusi ning oskavad langetada ka pädeva otsuse.

Pakkumiskutse koostamine, lepingu sõlmimine ning tarnija seire on vajalikud ainult juhul kui tarkvara ostetakse väljastpoolt ülikooli. Kuna käesoleva töö peatähelepanu on koondatud tarkvara arendusele TPÜ siseselt, siis need teemad jäävad hetkel lähemalt puudutamata.

Lepingut peaks antud juhul asendama tellija poolt koostatud ning hiljem IT osakonna poolt vastavalt täiendatud projektiplaan, milles on määratletud tarkvaratoote funktsionaalsused,

kasutajate nõudmised, süsteemi toimimiseks vajalikud mehhanismid jmt. Kui antud dokument on kooskõlastatud TPÜ juhtkonna, IT osakonna ning tellija poolt, siis seda võibki lugeda lepingu sõlmimiseks - tellija on määratlenud enda soovid ning tarnija aktsepteerib need. Juhtkond omakorda tunnustab loodava süsteemi vajalikkust ning varustab üksused tööks vajalike ressurssidega. Vältimaks hilisemaid arusaamatusi, peaks kooskõlastamine toimuma kirjalikult (projekti allkirjastamise näol) ja mitte suusõnaliselt.

2.1.2 Tarnimisprotsess

Tarnimisprotsess kirjeldab täpsemalt tarnija tegevused ja kohustused. Tarnija esmaseks ülesandeks on vaadata läbi tellija poolt esitatud projekt, viia sisse vajalikud parandused ning otsustada projekti edasine käekäik. Standardis on need tegevused nimetatud kui: algatamine, vastuse koostamine, lepingu sõlmimine. TPÜ-s on projekti esmaseks läbivaatajaks IT osakond, kes viib sisse vajalikud parandused ja muudatused. Seejärel tehakse otsus projekti edasise käekäigu osas. Otsustamisel on kahtlemata tähtis roll TPÜ juhtkonnal, kelle pädevuses on eraldada projekti realiseerimiseks vajalikud ressursid. Esialgse otsuse võib teha IT osakond, kes annab juhtkonnale ülevaate vajaminevatest ja olemasolevatest ressurssidest. Lõplikku otsust ja eelarvet ei saa kindlasti teha esialgse projektiplaani alusel. Seega on esmalt vajalik juhtkonna otsus selle kohta, kas projektiga üldse tegelema hakatakse. Pärast positiivse otsuse langetamist algab plaanimise protsess. Standardi järgi hõlmab see endas hankenõuete läbivaatamist, tarkvara elutsükli mudeli valimist, tarkvaraprojekti halduse nõuete kehtestamist jmt. Viimased kaks peaks olema ühekordselt teostatud ning dokumenteeritud tegevused ning neid peaks saama kohandada kõikidele järgmistele tarkvaraprojektidele. Seega jääb plaanimisel tarnija ülesandeks hankenõuete läbivaatamine ja vajadusel täiendamine.

Plaanimisele järgnevad arendus-, ekspluatatsiooni- ja hooldusprotsessid on eraldiseisvad protsessid, kuid kuuluvad tarnimisprotsessi alla. Neid kirjeldatakse lähemalt järgnevates punktides. Tarneprotsess lõppeb tarkvara läbivaatuse, hindamise ja üleandmisega ning projekti lõpuleviimisega.

2.1.3 Arendusprotsess

Arendusprotsess sisaldab reaalse rakenduse väljatöötaja ülesannete loetelu: "See protsess hõlmab tarkvaratootega seotud nõuete analüüsi, projekteerimise, programmeerimise, integreerimise, testimise ning installeerimise ja vastuvõtmise tegevusi." [ISO-Tep]

TPÜ-s eeldab arendusprotsessi jõudmine projekti heakskiitmist juhtkonna poolt. Esimese sammuna luuakse meeskond, mis koosneb nii tellija- kui ka tarnijapoolsetest esindajatest. Vastavalt oma pädevustele teostatakse ühise meeskonnana süsteeminõuete analüüs, koostatakse tarkvara nõuete spetsifikatsioon ning teostatakse muud vajalikud tegevused. Pärast

täpsemat analüüsi selgub ka tarkvaratoote kogumaksumus. Kuigi standardis on loetletud suur hulk kõikvõimalikke spetsifikatsioone, dokumente jmt., mille loomist soovitatakse, siis üldises plaanis muutuvad järjest populaarsemaks tarkvaraarendusmetoodikad, mis eeldavad väiksema hulga teoreetilise dokumentatsiooni koostamist ning suurema tähelepanu koondamist kasutajate vajadustele ja nende realiseerimisele. Sellest aspektist lähtuvalt on mitmed standardis loetletud etapid välja jäetud ja eeskujuks võetud juba mõne konkreetse arendusmetoodika ideoloogia.

Tarkvara arhitektuuri projekteerimine ja detailprojekteerimine on osaliselt kattuvad tegevused, viimase puhul on tulemus detailsem ja põhjalikum. Projekteerimine hõlmab endas andmebaaside üldskeemi koostamist, kasutajaliideste- ja testimisplaani väljatöötamist jmt. Oluliseks saab ka tarkvaraarenduse meetodi valimine, millele vastavalt määratletakse edasine tegevus. Järgneb rakenduse programmeerimine ja testimine. Standard ei kirjelda väga üksikasjalikult programmeerimise osa - arvatavasti on põhjus selles, et antud etapp määratletakse pigem mõne metoodikaga. Rõhutatud on tarkvara, andmebaaside ja testide dokumenteerimist, mis on kahtlemata väga oluline. Testimine hõlmab lisaks koodi ja funktsionaalsuse mehaanilisele testimisele ka kontrollimist, kas tarkvara vastab projektiplaanis esitatud nõuetele ja eeldatud tulemustele.

Arendusprotsessi käigus valmib lõplik tarkvarapakett, mis rakenduse koodi kõrval sisaldab ka kõiki juhendeid: kasutajajuhendit, juhendeid installeerimiseks, deinstalleerimiseks, varundamiseks, halduseks jne.

2.1.4 Ekspluatatsiooniprotsess

Ekspluatatsiooniprotsess kirjeldab ekspluateerija ehk kasutaja tegevused: katseekspluatatsioon, süsteemi ekspluatatsioon ja kasutaja toetamine [ISO-Tep]. Katseekspluatatsioon tähendab sisuliselt kasutajapoolset testimist eesmärgiga kontrollida rakenduse vastavust kirjeldatud nõuetele. Süsteemi ekspluatatsioon toimub selleks ettenähtud keskkonnas ja vastavalt kasutajajuhendile. Standardi järgi peaks kasutajatuge pakkuma ekspluateerija ehk siis tellija. TPÜ-s peaks kasutajatoe organiseerimine toimuma kokkuleppel hankija ja tarnija vahel. Üldjuhul peaksid mõlemad pooled olema valmis kasutajatuge pakkuma eeldusel, et rakenduse kasutamist puudutavatele küsimustele vastab tellija ning spetsiifilisematele IT alastele küsimustele vastab rakenduse looja (ehk IT osakond).

2.1.5 Hooldusprotsess

Hooldusprotsess muutub aktuaalseks juhul kui olemasolevat ja toimivat tarkvara on vaja muuta või täiustada. Esimese sammuna analüüsitakse, milline on muudatuse toime organisatsiooni tegevusele ja senisele süsteemile. Standard näeb ette muudatuse analüüsi

kolmest aspektist lähtuvalt:

- Muutuse tüüp - kas muudatuse käigus parandatakse süsteemi sisulist viga või on tegemist mõne funktsionaalsuse juurdelisamisega
- Muutuse ulatus - kui suured on muudatustega kaasnevad kulud, kui kaua võtab muudatuste tegemine aega
- Muutuse kriitilisus - kuidas mõjutab muutus olemasolevat süsteemi, selle ohutust ning turvalisust

TPÜ-s peaks antud juhul kehtima järgmine nõue: kui muudatus on väga suure ulatusega, siis tuleb algatada uus projekt - koostada projektiplaan, taotleda juhtkonnalt nõusolek, eraldada eelarvest raha jne. Kui tegemist on väiksemat sorti tööga, siis pole projekti algatamine vajalik ja täiendused tehakse paralleelselt muude ülesannetega. Siinkohal on tähtis ka märkida, et tuleb eristada kaht tüüpi muudatusi: süsteemi vea parandus või funktsionaalsuse lisamine. Kui süsteemi töös ilmneb kriitiline viga, siis kindlasti ei saa teostada pikalevenivat analüüsi ja arutleda muutuse tegemise vajalikkuse üle. Taoline muudatus tuleb teha võimalikult kiiresti. Sellele järgneb kogu süsteemi testimine ja parandatud versiooni töösseandmine. Funktsionaalsuste lisamine seevastu peab läbima eelpoolnimetatud analüüsi, kinnitamise jm. etapid. Nii veaparandused kui ka muudatused funktsionaalsustes peavad kajastuma tarkvara dokumentatsioonis.

Tegevused, mis puudutavad tarkvara kasutuselt mahavõtmist pole TPÜ-s veel kuigivõrd aktuaalsed, kuna hetkel luuakse alles tarkvararakenduste esimesi versioone. Seda etappi on mõistlik hakata täpsemalt formuleerima siis, kui ilmneb taoline vajadus. Standardis märgitud kasutajate informeerimine on üks olulisemaid tegevusi. Samas võiks tuua lisakommentaariks, et TPÜ-s on ette tulnud juhtumeid, kus IT osakond jätkuvalt kindlustab tellitud tarkvara töötamise serveris, kuid keegi seda enam ei kasuta. Niisiis peaks (vastupidiselt standardis esitatud nõudele) ka kasutajatel endil lasuma kohustus IT osakonnale teada anda, kui üks või teine rakendus enam kasutusel ei ole - mittevajaliku tarkvara saaks maha võtta ja säästa niigi nappe ressursse.

2.2 Abiprotsessid

Tarkvara elutsükli abiprotsesside hulka kuuluvad dokumenteerimisprotsess, konfiguratsiooni halduse protsess, kvaliteedi tagamise protsess, verifitseerimisprotsess, valideerimisprotsess, ühisläbivaatuse protsess, auditeerimisprotsess ja probleemilahendusprotsess. Olulisimad on dokumenteerimisprotsess ning kvaliteedi tagamise protsess.

2.2.1 Dokumenteerimisprotsess

Dokumenteerimisprotsess on abiprotsessidest olulisim. Nagu ütleb standard, on dokumenteerimine elutsükliprotsessi või -tegevusega loodud informatsiooni jäädvustamine. Dokumentatsiooni ei vaja mitte üksnes programmeerijad, vaid ka üksuste juhid, tarkvara kasutajad jt.

Esimese dokumendina tuleks koostada nimekiri dokumentatsiooni osadest, mis on olulised loodava tarkvara seisukohalt. Nimekirjas peaks olema märgitud dokumendi nimetus, selle otstarve ja sihtgrupp. Koostamisel tuleb kindlasti arvestada tarkvara mahuga ning seda silmas pidades jälgida, et nõutavate dokumentide arv ei oleks üleliia suur. Loomulikult pole ka puudulik dokumentatsioon hea. Standard lubab kasutada ka automaatseid dokumenteerimisvahendeid. Siinkohal on hea märkida, et koodi ja andmebaaside dokumentatsiooni luuakse TPÜ-s juba automatiseeritult. Ülejäänud dokumentatsiooni osad luuakse käsitsi. Kogu dokumentatsioon säilitatakse digitaalsel kujul ühtse paketina koos tarkvaraga.

2.2.2 Konfiguratsioonihaldus

Konfiguratsioonihalduse käigus luuakse süsteem muudatuste ohjeks, tarkvara redaktsioonide piiritlemiseks jne. TPÜ-s tähendab see eelkõige tarkvara versioonide nummerdamise süsteemi väljatöötamist. Üksikasjalikuma raamistiku koostamine ei tundu momendil vajalik olevat, kuna see oleks taoliste väiksemat sorti tarkvaraprojektide puhul ilmne liialdus.

2.2.3 Kvaliteedi tagamise protsess. Verifitseerimis- ja valideerimisprotsessid. Ühisläbivaatuse- ja auditeerimisprotsess

Kõik nimetatud protsessid täidavad ühel või teisel moel sama eesmärki - kontrollida ja tagada, et tarkvaratooted töötaksid veatult ning vastaksid nende kohta esitatud nõuetele. Antud protsessides võivad osaleda nii organisatsiooni liikmed (kvaliteedi tagamise-, ühisläbivaatuse protsess) kui ka väljastpoolt tulevad sõltumatud eksperdid (auditeerimis-, verifitseerimis-, valideerimisprotsess). Ühisläbivaatuse protsessis peaksid osalema just nimelt tarkvara kasutajad, kes oskavad kõige paremini avastada probleeme ja küsitavusi rakenduses. Kõik tarkvara juures leitud vead tuleb sisestada probleemide lahendamise protsessi.

Ülalnimetatud protsesse TPÜ tarkvaraprojektide puhul siiamaani praktiliselt rakendatud ei ole. See võib olla ka üheks põhjuseks, miks tarkvararakendused on tihti puudulikud ning paljud vead ilmnevad alles kasutamise käigus.

2.2.4 Probleemide lahendamise protsess

Probleemide lahendamise protsessi ülesandeks on - nagu nimigi ütleb - kõikvõimalike probleemide analüüsimine ja lahendamine. Seda protsessi võib rakendada mistahes teise

protsessi raames, kui peaks ilmnema mõni küsitavus või probleem. Oluline on dokumenteerida probleem, selle analüüs ja lahenduskäik.

2.3 Organisatsioonilised protsessid

Organisatsioonilisi protsesse on neli: haldusprotsess, infrastruktuuriprotsess, täiustusprotsess ja koolitusprotsess. Kuna tegemist ei ole niivõrd tarkvaraarendust otseselt puudutavate protsessidega vaid juba halduslike ja personaliküsimustega, siis käesolevas töös jäävad need lähemalt käsitlemata.

2.4 Kohandamisprotsess

Kohandamisprotsess on esitatud standardi lisana ja teeb ettepanekud standardi kohandamiseks tarkvaraprojektile. Protsessi käigus realiseeritakse projekti keskkonna tuvastus, sisendite taotlemine, protsesside, tegevuste ja tööde valimine ning kohandamisotsuste ja -kaalutluste dokumenteerimine. Üks osa - projekti keskkonna ning protsesside-tegevuste-tööde tuvastus on teatud määral realiseeritud käesolevas töös. Ülejäänud tegevused tuleks lisada konkreetsetes situatsioonides vajaduse tekkimisel.

3 XP (*Extreme Programming*)

XP meetodist räägitakse kui kergekaalulisest, "väledast" (agiilsest), inimese- ja suhtluskesksest ning kasutaja vajaduste rahuldamisele orienteeritud arendusprotsessist. XP looja Kent Beck võttis nimetatud meetodi esmakordselt kasutusele 1996. aastal. Niisiis on tegemist suhteliselt uue, kuid praeguseks palju populaarsust võitnud tarkvaraarendusmeetodiga. Populaarsuse üheks põhjuseks peetakse eelkõige kliendikesksust. Projekti algstaadiumist alates suheldakse tihedalt tulevaste kasutajatega ning muudetakse ja täiustatakse tarkvara vastavalt nende soovidele. Kuna kliendil enamasti puudub väga konkreetne visioon uue tarkvara funktsionaalsuste ja väljanägemise osas, siis on see heaks võimaluseks järk-järgult välja kujundada kasutajale meelepärane tarkvara. XP on kasutusel peamiselt väikestes arendusmeeskondades, kuna see nõuab dünaamilisust, kiiret ümberpaigutumist jne.

Kui nn. "raske" arendusmeetodi rakendamine tähendab enamasti suure hulga nõuete ja reeglite järgimist ning mahuka dokumentatsiooni loomist, siis XP puhul on nõuete hulk viidud miinimumini. Alles on jäänud vaid olulisimad. XP eesmärgiks on luua keskkond, kus arendajad tunnevad end vabalt olemaks produktiivsed ja loomingulised, kuid säilitada siiski teatud tasemel organiseeritus ja fokuseeritus [XP].

Planeerimine	**Kodeerimine**
Kasutajalood.	Tellija pidev kohalolek.
Redaktsioonide planeerimine.	Kood vastab kokkulepitud standardile.
Redaktsioonid lühikese aja tagant.	Testid kirjutatakse enne kodeerimist.
Projekt on jaotatud iteratsioonideks.	Paarisprogrammeerimine.
Planeerimine iteratsiooni alguses.	Pidev integreerimine.
Inimeste ümberpaigutamine.	Kood on kollektiivne omand.
Koosolek iga päeva alguses.	Koodi optimeerimine.
Parandused XP-metoodikasse.	Ületunnitöö vältimine.
Disain	**Testimine**
Lihtne disain.	Kogu kood on kaetud testidega.
Süsteemi metafoor.	Testimine enne redaktsiooni.
Ainult nõutud funktsionaalsus.	Vea leidmisel kirjutatakse test.
Rekodeerimine.	Pidev kasutajapoolne testimine.

Tabel 2. XP-metoodika reeglid ja tavad [XP]

Järgnevalt on kirjeldatud XP rakendamise peamised põhimõtted ja lähtepunktid, mille leiab metoodikat tutvustaval veebilehel (http://www.extremeprogramming.org/rules.html). Paralleelselt ülevaatega XP-st on toodud kogemused TPÜ tarkvaraprojektidest.

3.1 Kasutajalood

XP arendusmetoodika puhul on kasutajalugude osatähtsus äärmiselt suur. Kasutajalood joonistab või kirjutab rakenduse tulevane kasutaja. Üks lugu koosneb ühest ekraanipildist või 1-3 lausest ning kirjeldab soovitavalt ainult üht funktsionaalsust. Kogutud kasutajalood asendavad tarkvarale esitatavate (teoreetiliste) nõuete mahukat dokumentatsiooni. Üks suur erinevus kasutajalugude ja dokumentatsiooni vahel on see, et esimese puhul on vaatluse all kasutaja (ehk konkreetse indiviidi) vajadused, teise puhul aga üleüldised nõudmised, mis ei pruugi üldse reaalsust kajastada.

TPÜ-s on kasutajalugude suureks probleemiks asjaolu, et tihti puudub kasutajatel selge visioon tarkvarast ning teadmised infotehnoloogilistest võimalustest. Seega võib juhtuda, et kasutaja näeb hulga vaeva lugude kirjutamisel, kuid hiljem osutuvad need arendusel kasutuskõlbmatuks. Võib ette tulla olukordi, kus kasutajal tekkib kinnisidee mõne funktsionaalsuse vajalikkuse osas, aga see ei ole tehniliselt realiseeritav või pole lihtsalt otstarbekas seda teostada, kuna analoogne rakendus on juba olemas ja see on võimalik sisse osta. Sel juhul on väga keeruline tellijale selgitada põhjusi, miks üht või teist funktsionaalsust tellitava tarkvara raames ei realiseerita. Üleülikooliliste projektide puhul tekkib ka küsimus, milliste kriteeriumide alusel valitakse kasutajalugude kirjutajad ning kes kehtestab kriteeriumid ja teeb valikud. Võimalus, et kasutajalugusid saavad kirjutada kõik soovijad, võib tekitada kaks äärmuslikku olukorda. Ühelt poolt võib kasutajalugusid koguneda liiga palju ning liiga erinevaid, võimaldamaks luua adekvaatset üldpilti loodavast tarkvarast. Samahästi võib tekkida olukord, kus kasutajalugusid ei tule peaaegu üldse, kuna impersonaalselt antud ülesanne ei puuduta üldjuhul mitte kedagi isiklikult ning seda lihtsalt ignoreeritakse. Parem lahendus on niisiis välja valida konkreetsed isikud ning teha kasutajalugude kirjutamine neile kohustuslikuks. Alternatiivina võiks TPÜ jaoks pakkuda välja lahendust, kus esialgne projekt ja "üldplaan" kirjeldatakse projekti algataja poolt ning edasise analüüsi käigus toimuvad intervjuud kasutajatega. Intervjuudel võib selgitada ka kasutajalugude kirjutamise ideoloogiat ning soovi korral võivad kasutajad neid luua. Lisaks kasutajalugudele võib koostada süstematiseeritud funktsioonide loetelu, mis peaks andma selge ülevaate loodava tarkvara eesmärkidest ja ka mahust.

3.2 Redaktsioonid

Redaktsioonide puhul peab XP tähtsaks tähtaegade ning funktsionaalsuste (kasutajalugude) realiseerimise järjekorra kindlaks määramist. Silmas tuleb pidada tarkvaraarenduse nelja muutujat (funktsionaalsus, ressursid, aeg, kvaliteet), mille vahel balansseerides tuleb saavutada optimaalseim ning kõik osapooli rahuldav tulemus. Nagu teada, saab nendest neljast kindlaks määrata vaid kolme ja üks jääb nö. "vabaks", mille arvelt on võimalik projektis korrektiive teha.

TPÜ tarkvaraprojektide puhul on siiamaani olnud kindlaks määratud vaid lõpp-tähtaeg. Tarkvara arendusele kuluv aeg on iteratsioonideks jaotamata ning alameesmärgid defineerimata. Tõenäoliselt ongi enamuse probleemide põhjuseks asjaolu, et puuduvad vahe-etapid, mil peaks toimuma tarkvara (kasutajapoolne) ülevaatamine, testimine ja tagasiside andmine. On esinenud mitmeid juhtumeid, kus tarkvara saab valmis vahetult enne tähtaja saabumist ning korralikke teste ei jõutagi läbi viia - vead hakkavad ilmnema alles rakenduse reaalsel kasutamisel.

XP-metoodika puhul rõhutatakse vajadust anda redaktsioone välja lühikeste vahedega ning väiksemate muudatustega. Ressursside vähesuse tõttu pole see TPÜ-s paraku võimalik, sest see eeldaks eraldi töötaja palkamist, kelle ülesandeks oleks tarkvara pidev installeerimine töökeskkonda. Siiani on sellega tegelenud TPÜ süsteemiadministraator, kellel päevast-päeva ilmuvate redaktsioonide installeerimiseks tõenäoliselt aega ei jätkuks. Teiselt poolt on praeguseni loodud rakendused suhteliselt vähese funktsionaalsusega ning nende tükeldamine veelgi väiksemateks osadeks ei pruugi enam mõistlik olla.

3.3 Iteratsioonid

XP-metoodika jaotab tarkvaraarenduse protsessi 1-3 nädalasteks etappideks ehk iteratsioonideks. Iga iteratsiooni eel antakse kõigile ülesanded, mis tuleb iteratsiooni lõpuks valmis saada. Kõik ülesanded kooskõlastatakse ka tellijaga. Iteratsiooni lõpus vaadatakse üle, kas kõik püstitatud eesmärgid on täidetud ning saadakse kasutajalt tagasisidet. Siis püstitatakse uued ülesanded järgmise iteratsiooni jaoks ning kogu protsess kordub. Kuna tähtajad on väga lühikesed, siis on lihtsam konkreetseid ülesandeid anda ning see omakorda lihtsustab hiljem nende kontrollimist. Ühtlasi hoiab lühike tähtaeg programmeerijad motiveeritud ja pidevas töös.

Nagu juba eelpool öeldud - ei ole TPÜ tarkvaraprojektide puhul iteratiivset arendustegevust siiani rakendatud. Produktiivsema töö ja kvaliteetsema tulemuse saavutamiseks peaks aga edaspidi sellele järjest enam tähelepanu pöörama.

3.4 Programmeerijate ümberpaigutamine

Programmeerijate ümberpaigutamise eesmärgiks XP-metoodikas on saavutada olukord, kus iga programmeerija on kogu süsteemi tööga kursis ning kood on sõna otseses mõttes ühisomand - igaüks võib ja suudab mistahes süsteemi osa parandada, muuta või täiendada, ilma et koodist arusaamisele soovitud kohas väga pikalt aega kuluks.

Kuna TPÜ-s on momendil tööl vaid kaks programmeerijat ning mõlemad on võetud tööle erinevate projektide realiseerimiseks, siis taoline ümberpaigutamise võimalus siin puudub. Ühe abinõuna - tagamaks oskusteabe säilimist - rakendatakse tarkvara koodile ja dokumentatsioonile kõrgete nõuete esitamist. Kui kood on arusaadavalt kommenteeritud ning põhjalikult dokumenteeritud, siis peaks väljastpoolt tuleval uuel programmeerijal olema suhteliselt lihtne koodist aru saada.

3.5 XP-metoodika parandamine

XP-metoodikat puudutav nõuanne ütleb: "Fix XP when it breaks" ("P*aranda XP-d kui see ei toimi"*) [XP]. On üldteada fakt, et kõik projektid - olgu nad tarkvaraga seotud või mitte - on erinevad ning ei ole võimalik anda täpset ja ühest nõuannete paketti, mis garanteeriks sajaprotsendilise edu. Niisiis on vaja kehtestada reeglid organisatsiooni ühtse toimimise tagamiseks, kuid samal ajal tuleb jälgida, et reeglid ei muutuks koormavaks ja ei hakkaks projekti teostumist pidurdama. Kui on ilmselgelt näha, et üks või teine nõudmine vaatamata pingutustele ei toimi, siis pole mõtet selle täitmist vägisi nõuda.

Seetõttu on ka käesolev töö esitatud pigem üldiste põhimõtetena kui üksikasjalike reeglitena. Ajapikku kujundavad osakonna töötajad ise need head tavad, mida järgitakse. Samas on siiski kasulik evida teatud raamistikku, mis annaks kätte tegevuste suunised ja aitaks ühtlustada tarkvaraprojektide elutsükli kulgemist.

3.6 Süsteemi metafoorid ja prototüübid

Metafoor on lühike lugu, mis kirjeldab nii programmeerijatele, kliendile kui ka juhtkonnale, kuidas süsteem töötab. Rakenduse metafoor peab olema lühike, lihtne, kõigile (ka täiesti võõrale) üheselt arusaadav mõnesõnaline selgitus rakendusest. Metafoori kasutamine on mõeldud selleks, et hoida meeskonda nö. "ühel lainel", mis puudutab klassidele, meetoditele jm. nimetuste andmist. See on üks, mis lihtsustab nii kohe kui ka hiljem koodist arusaamist. [Piho2003]

Prototüüpide loomise põhiliseks eesmärgiks on aidata välja selgitada nii tehnilisi kui disaini probleemkohti, mis reaalse tarkvara kasutamisel tekkida võivad.

Nagu juba mitmeid kordi mainitud, on tarkvara analüüsi etapp TPÜ-s olnud siiamaani

praktiliselt olematu. Niisiis ei saa siinkohal (kogemustest lähtuvalt) kommenteerida ei metafooride ega ka prototüüpide loomise vajalikkust või kasulikkust TPÜ tarkvaraprojektide seisukohast.

3.7 Minimaalne funktsionaalsus

Funktsionaalsuse osas näeb XP ette, et luuakse ainult need funktsionaalsused, mis on kasutajate poolt nõutud. Niikaua kuni uusi nõudmisi ei tekki, ei ole programmeerijatel lubatud ise midagi välja mõelda ja juurde lisada. Põhjenduseks taolisele nõudmisele tuuakse väide, et üldjuhul ainult 10% programmeerijate initsiatiivil lisatud funktsionaalsustest leiab hiljem ka rakendust - seega 90% on raisatud töövaev ning aeg. Niisiis peaks ühe iteratsiooni raames programmeerijate peatähelepanu olema pööratud pigem koodi kvaliteedi kui kogu tarkvara parandamisele (antud kontekstis on tarkvara parandamise all silmas peetud programmeerija omaalgatuslikku funktsionaalsuste lisamist, mida klient pole nõudnud, kuid mis programmeerijale näib olevat vajalik või kasulik).

TPÜ-s on märgata tendentsi, et programmeerijad mõtlevad iseseisvalt kõikvõimalikke funktsionaalsusi välja ja lisavad neid arvates, et hiljem leiavad need kasutust. Suuremat tähelepanu pööratakse tarkvara välisele poolele ja vähem koodi kvaliteedile. Üheks põhjuseks, miks programmeerijad saavad suhteliselt vabalt tegutseda, on tõsiasi, et klient ei osale aktiivselt arendusprotsessis ning ei oma eriti selget nägemust soovitud rakendusest. Kuna ka tarkvara eelanalüüs on siiamaani olnud väga minimaalne, siis langebki kogu vastutus funktsionaalsuse loomise eest programmeerijatele. Siinkohal ei saa taas märkimata jätta esialgse projektiplaani ja nõudmiste koostamise vajadust tellija poolt ning sellele järgnevat põhjalikku eelanalüüsi, mille teostavad IT osakonna analüütikud.

3.8 Tihe koostöö kliendiga

Kui suur osa XP põhimõtetest on pigem soovituslikku laadi, siis kliendiga tiheda koostöö tegemine on nõue - ja väga oluline. Tellija ei ole ainult kõrvaltvaataja, vaid täieõiguslik tarkvaraarendusmeeskonna liige, kes on samaväärselt vastutav loodava tarkvara kvaliteedi eest. Tellija esmaseks ülesandeks on kirjutada kasutajalood ja esitada nõudmised projekteeritava süsteemi kohta, vastata programmeerijate küsimustele loodava tarkvara osas ning hiljem testida valmivaid komponente ja anda nende kohta tagasisidet.

Kuna TPÜ on organisatsioonina küllalt suur ja mitmekülgne, siis on IT osakonnal praktiliselt võimatu omada ülevaadet kõigest ülikoolis toimuvast ning seega on ükskõik millise tarkvara loomisel äärmiselt oluline, et projektis osaleks ka tellija. Tellija peaks olema kõige paremini informeeritud loodavat rakendust puudutavate vajaduste ja nõudmiste osas. Samamoodi peaks

tellija ülesandeks olema rakenduse kasutamist reguleerivate eeskirjade väljatöötamine ja kehtestamine.

3.9 Koodi standardiseerimine

XP pakub välja, et programmeerijad peaks järgima ühtset koodi standardit. See tagaks koodi ühtsuse ja võimaluse kogu arendusmeeskonnal mistahes koodi osast kergesti aru saada.
Idee on kahtlemata väga hea ja soovitatav, kuid reaalsuses paraku üpris keeruliselt teostatav. Nimelt on kogemustega programmeerijal üldjuhul välja kujunenud nö. "oma stiil", millest teda lahti ütlema panna on suhteliselt keeruline. Seega oleks mõistlik leida kompromiss kahe alternatiivi vahel: luua ühtne kodeerimise stiil oma organisatsiooni siseselt programmeerijate endi kaasabil. Ühtsed reeglid peaks esmajärjekorras kehtima muutujanimedele, funktsioonidele jmt. aga ka failinimedele ja failide paigutusele ühe programmipaketi raames.

3.10 Paarisprogrammeerimine

XP-s kirjeldatud paarisprogrammeerimine tähendab seda, et kaks inimest istuvad ühe arvuti taga ja kirjutavad koos ühte ja sama koodi. Üks programmeerija kirjutab koodi ja pöörab suuremat tähelepanu hetkel loodava süsteemiosa kvaliteedile. Teine istub kõrval ja jälgib, et semantilisi vigu ei tehtaks ning ühtlasi mõtleb strateegiliselt kogu süsteemile suures plaanis.
Ühelt poolt kiidetakse sedalaadi programmeerimist väga - väidetavalt hõlbustab see süvenemist, kiirendab arendusprotsessi ja vähendab tehtavaid vigu. Kriitikute sõnul aga on see vaid harukordne juhus, kui nimetatud eelised tõepoolest esile tulevad. Üldjuhul muudab see arendustöö siiski keerukamaks.
TPÜ-s oleks sellise meetodi kasutuselevõtmine üpris vähetõenäoline. Taas on suurimaks probleemiks kaadri vähesus. Oleks ebareaalne ja kindlasti ka aeganõudvam, kui kaks arendajat töötaksid kahekesi mõlema projektiga. Teiselt poolt tundub, et paarisprogrammeerimist ei lubaks hetkel rakendada ka programmeerijate isiksuseomadused.

3.11 Tihe integreerimine

XP-metoodika puhul töötab mitu programmeerijate paari ühe suurema süsteemi kallal. Iga paari ülesandeks on kodeerida väikseid süsteemi osi ning need siis suurde süsteemi integreerida. Seetõttu on väga tähtis, et integreerimine toimuks pidevalt. Integreerimise käigus on võimalik kindlaks teha süsteemi ebakohti ning need siis koheselt parandada. Integreerimiseks on olemas eraldi server ning ühel ajahetkel saab integreerimisega tegeleda üks programmeerijate paar. Integreerimine peab toimuma kohe, kui uus süsteemi komponent on valminud ning testitud. See tagab olukorra, kus iga järgmine integratsioon toimub uusima redaktsiooni peale.

Rääkides TPÜ-st jõuame taas ressursside puudusest tulenevate probleemideni, mis takistavad taolise metoodika rakendamist. Praegusel momendil pole taoline tihe integreerimine - eesmärgiga rakendusi reaalses keskkonnas testida - veel nii aktuaalne, kuid ühtse infosüsteemi loomisel peaks hakkama sellele suuremat tähelepanu pöörama, et tagada uute tarkvarakomponentide ühtimine olemasoleva süsteemiga.

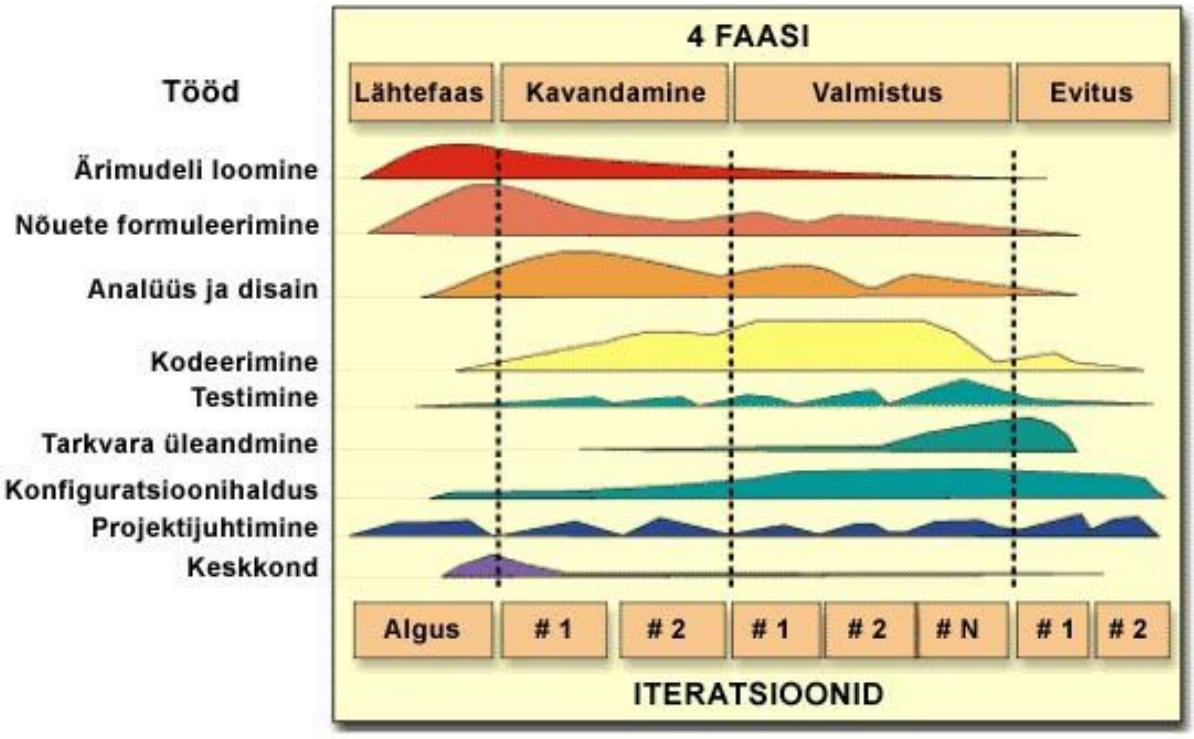
3.12 Testimine

XP-metoodikas on kasutusel testimisel tuginev arendustehnika (*Test Driven Development*). Kogu meeskond tegeleb tarkvara testimisega igal ajahetkel projekti esimestest päevadest alates. Programmeerijate esmaseks ülesandeks ongi vastavalt kasutajalugudele kirjutada testid ning alles seejärel hakata looma koodi, mis vastaks testides toodud nõuetele. Funktsionaalsed testid luuakse koostöös tellijaga, suurema osa nendest testidest viib läbi tellija. Nii on tagatud, et tarkvara töötab just nii nagu kasutaja seda soovib. Piirsituatsioonid jmt. tuleb testida siiski professionaalsemate inimeste poolt.

TPÜ-s ei ole testimisel põhineva arendusmetoodika rakendamine hetkel ilmselt mõeldav. Suurimaks takistuseks on vastava valmisoleku puudumine programmeerijatel. Võimalikku lahendust olukorrale näeb IT osakond arendusmeeskonna täiendamises kogemustega testija võrra.

4 RUP (*Rational Unified Process*)

RUP on objektorienteeritud arendussüsteem, mis pakub juhtnööre, malle ja näiteid tarkvaraarenduse kõigi aspektide ja etappide tarvis. RUP kujutab endast komplekssет tarkvaraarenduse vahendit, mis ühendab arenduse protseduurilised aspektid (defineeritud etapid, võtted ja meetodid) ning arenduse muud komponendid (dokumendid, mudelid, juhendid, koodid jne.) ühtsesse raamistikku. [E-teatmik] Eesmärgiks on kindlustada kvaliteetse tarkvara tootmine lõpp-kasutajale kindlalt määratud aja ja eelarve raames. RUP eelistab suuremahulise tekstidokumentatsiooni koostamisele mudelite loomist, kasutades selleks UML-keelt (*Unified Modeling Language*)[3]. RUP-i iseloomustatakse kui kasutuslugudest juhitud ja arhitektuurikeskset meetodit, mis kasutab iteratiivset elutsükli mudelit.



Joonis 1. Ülevaade RUP-arendusest [Pollice2003]

Kui üldiselt levib teadmine, et RUP-i kasutatakse vaid suurte tarkvaraprojektide juures, siis põhimõtteliselt on seda metoodikat võimalik kohandada ka väikestele projektidele. RUP hõlmab endas lihtsalt teatud häid tavasid (*best practices*), mis on kasutusel igasuguse suurusega tarkvaraprojektide puhul. Kuus head tava, mida RUP-i puhul nimetatakse on: iteratiivne arendus, nõuete haldus, komponendipõhine arhitektuur, tarkvara visuaalne

3 UML (*Unified Modeling Language*) - keel tarkvarakeskse süsteemi artefaktide visualiseerimiseks, spetsifitseerimiseks, väljatöötamiseks ja dokumenteerimiseks. UML-i töötasid välja firma Rational Software juhtivad süsteemispetsialistid.

modelleerimine, tarkvara verifitseerimine, muutuste kontroll. [Gornik2003]

RUP kehtestab arenduse neli faasi, kusjuures iga faas koosneb teatud arvust eraldiseisvatest iteratsioonidest, mille lõpptulemus peab rahuldama kindlaksmääratud kriteeriume, enne kui saab üle minna järgmisse faasi. Kõige esimeses - lähtefaasis - defineeritakse projekti ulatus ja selle ärimudel, kavandusfaasis luuakse enamus kasutajalugudest, analüüsitakse detailsemalt projekti vajadusi ja defineeritakse selle arhitektuuriline alus, valmistusfaasis luuakse rakenduse kujundus ja lähtekood ning viimases - evitusfaasis - antakse rakendus üle kasutajatele. [Gornik2003]

4.1 Lähtefaas (*Inception Phase*)

Lähtefaasi käigus luuakse arenduse lähteideed ning täpsustatakse neid tasemeni, mis võimaldaks alustada kavandifaasi. Võtmesõnaks on visioon. Määratletakse, mis süsteemiga on tegu, kes seda kasutama hakkavad, miks seda kasutatakse, millised on süsteemi põhitunnused, millised on põhilised piirangud jne. Töötatakse välja ja esitatakse sihtsüsteemi talitlusjuhtumi (*business case*) kirjeldus ja süsteemi prototüüp, määratletakse projekti maht. Suures osas on ära nimetatud kasutuslood ning osaliselt on need ka detailsemalt lahtikirjutatud. Lähtefaasi põhieesmärgiks on saavutada huvirühmade esialgne nõusolek uue süsteemi ülesannete ja tema loomiseks kulutatavate ressursside suhtes. [Petuhhov2004]

Lähtefaasi käigus viiakse läbi neli põhitegevust [Pollice2003]:

Projekti ulatuse määramine - määratletakse üleüldine kontekst ning kõige tähtsamad nõudmised.

Juhtumi analüüs - defineeritakse riskide maandamise strateegia, määratletakse projekti maksumus, ajakava ja tasuvus.

Esialgse süsteemi arhitektuuri loomine - arendustehnoloogiate valik, prototüübi loomine jm. projektiga seonduva täpne läbimõtlemine ja paikapanemine, mis aitab vältida suurte vigade ja probleemide tekkimist hilisemates projekti faasides, kus nende likvideerimise maksumus kujuneb märgatavalt suuremaks.

Projekti keskkonna ettevalmistamine - iga projekti jaoks on vaja määratleda ka füüsilised ressursid.

Lähtefaasi kestvus sõltub otseselt projekti mahust, väiksemate projektide puhul võib lähtefaasi läbida koguni mõne päevaga.

Nagu eelnevalt mitmel pool mainitud, on TPÜ tarkvaraprojektide eelanalüüs ja täpsem piiritlemine jäänud siiamaani üpris kesiseks. Kuna see on aga projekti õnnestumise seisukohalt esmatähtis, siis peaks edaspidi sellele rohkem tähelepanu pöörama. Eelkõige on oluline aktiivne suhtlus tellijaga ja võimalikult erinevate kasutajagruppidega. Viimase puhul on

eesmärgiks see, et loodav tarkvara arvestaks võimalikult paljude ja erinevate soovide ning nõudmistega. Momendil on põhirõhk asetatud kõrgema taseme analüüsile, mis puudutab ülikooli infosüsteemi, andmebaase, registreid jmt. Kuid praegusest põhjalikumalt tuleks hakata tegelema ka projekteeritavate tarkvararakenduste kavandamisega.

4.2 Kavandusfaas (*Elaboration Phase*)

Kavandamise faasi eesmärgiks on analüüsida püstitatud ülesannet ja luua arhitektuuriline põhi, mida saaks kasutada uue süsteemi vundamendina. Kavandusfaasi käigus arendatakse välja projektiplaan ja realiseeritakse suure riskiga seotud osad, kirjeldatakse suurem osa nõudmistest ja hinnatakse üldist ajagraafikut ning ressursse. [Petuhhov2004] Selles faasis luuakse ka enamus kasutuslugude stsenaariumitest ja kasutajaliidese prototüüpidest (*Use-Case Storyboards, UI Prototypes*), identifitseeritakse kõik kasutajad (*actor*). Määratletakse ka mittefunktsionaalsed nõudmised - nõudmised, mis ei ole otseselt seotud ühegi kasutuslooga. Faasi lõpus on kogu süsteem põhjalikult läbi mõeldud, nii et on võimalik juba konkretiseerida projekti maht, eelarve ja tähtajad.

Kavandamise faasi kolm põhitegevust [Pollice2003]:

Arhitektuuri defineerimine, valideerimine ja piiritlemine - võetakse kokku arhitektuuriga seotud küsimused ja lahendused, analüüsitakse võimalusi ja nende kasutatavust loodavas süsteemis.

Visiooni viimistlemine - kui lähtefaasi käigus luuakse esialgne visioon, siis kavandamise faasis muutub see selgemaks ja detailsemaks.

Iteratsioonide plaani loomine valmistusfaasiks - iteratsiooni planeerimisel järjestatakse nõudmised prioriteetide alusel ja tööd alustatakse prioriteetsematest süsteemi osadest. Järjestamise kriteeriumiteks on riskid (tehniline keerukus jm.) ja komponendi osatähtsus süsteemis. Prioriteetide seadmisel arvestatakse ka kliendi seisukohti - funktsioonid, mis on kliendile tähtsamad, luuakse varem. [Petuhhov2004]

Kavandamise faasis pööratakse tähelepanu juba ka testimisele - püütakse kirjeldada, mida hakatakse testima ja kuidas. Kasutatavust võib testida mõne juba realiseeritud stsenaariumi najal. Mõningatel juhtudel on koostatud ka tarkvara eelkasutusjuhend (*preliminary user manual*) [Gornik2003].

4.3 Valmistusfaas (*Construction Phase*)

Valmistuse käigus toimub tarkvararakenduse kodeerimine ning integreerimine olemasolevasse süsteemi. Arhitektuursest arendusalusest saab töövalmis süsteem. Samuti sooritatakse testid kinnitamaks toote kvaliteeti. Valmistusfaasi käigus luuakse konkreetne tarkvara toode, mida

on võimalik juba tellijale üle anda. Nagu tööde loetelust näha on tegemist kõige rohkem ressurssi nõudva faasiga. Üldjuhul koosneb see faas kõige suuremast arvust iteratsioonidest ja iga iteratsioon lisab juba nähtava funktsionaalsuse. Valmistusfaasi lõpuks peab toode sisaldama juba kogu planeeritud funktsionaalsust. Vastavalt tarkvaratoote täiendustele on muudatused vaja teha ka dokumentatsioonis. Valmistusfaasi lõppedes antakse välja tarkvarapaketi nn. beeta-versioon.

Valmistusfaasi kolm põhitegevust [Pollice2003]:

Ressursihaldus, protsessi kontroll - ülesannete jagamine meeskonnaliikmete vahel, tähtaegadest kinnipidamise jälgimine jne.

Süsteemi komponentide arendus ja testimine - funktsionaalsuste loomine kasutuslugudele ja süsteemi üldistele nõuetele vastavalt.

Iga iteratsiooni töötulemuste hindamine - iteratsiooni lõppedes kontrollitakse, kas valminud süsteemi komponendid vastavad planeerimisel esitatud nõuetele.

Kuna tarkvara tootmise puhul toonitatakse üha enam iteratiivse arenduse tähtsust, siis on ka TPÜ-s plaanis seda juurutama hakata. Eelkõige peaks see leidma rakendust valmistusfaasis, mil toimub tarkvara kodeerimine ja testimine.

4.4 Evitusfaas (*Transition Phase*)

Evitusfaasis toimub tarkvararakenduse lõplik üleandmine kasutajale. Kui kasutajal avaneb võimalus hakata rakendust reaalses situatsioonis kasutama, siis üldjuhul ilmneb veel rida vigu ja probleeme, mida on vaja parandada. Ühtlasi võib kasutajate tagasisidest ilmneda vajadus mõne lisafunktsionaalsuse järele. Pärast muutuste tegemist antakse välja uus redaktsioon. Siinkohal tasuks vahemärkusena tõdeda, et RUP-i puhul on redaktsioon hoopis teise tähendusega kui XP puhul. Redaktsioon ei ole ainult uus funktsionaalsus, vaid see hõlmab palju enamat. Uue redaktsiooni välja andmiseks tuleb ette valmistada ka dokumentatsioon, kasutajajuhendid, testida rakendust lõpp-kasutaja keskkonnas, viimistleda pisiasju tellija tagasiside põhjal jne. - ühesõnaga koostada kogu tarkvarapakett ja viia see täismahus lõpp-kasutajani.

Evitusfaasi neli põhitegevust [Pollice2003]:

Kasutajatoe organiseerimine - ressursside tagamine pädeva kasutajatoe andmiseks, tugiisikute koolitamine, kasutajajuhendi täpsustamine ja viimistlemine jne.

Toote testimine reaalses keskkonnas - üldjuhul on arendusserver ja kliendi server erineva konfiguratsiooniga ning seetõttu on enne tarkvara üleandmist vaja seda testida ka reaalses keskkonnas.

Väikeste muudatuste tegemine tarkvaras, tuginedes kasutajate tagasisidele - soovitatav on

planeerida üks või mitu beeta-testimise perioodi, mil piiratud hulk kasutajaid testib tarkvara. Testitulemuste põhjal saab kasutaja veel teha ettepanekuid väiksemate muudatuste või paranduste tegemiseks.

Tarkvara üleandmine lõpp-kasutajale - tarkvara lõpliku versiooni, mis sisaldab ka lõplikult viimistletud dokumentatsiooni, kasutajajuhendeid jms., üleandmine.

5 TARKVARA ARENDUSPROTSESSI ÜLDPÕHIMÕTTED TPÜ-s

Järgnevalt on välja toodud olulisimad tarkvaraarenduse etapid TPÜ-s ning nendega seonduvad põhimõtted, millega kõik osapooled (nii tellija, kasutaja kui ka tarnija) ühistest huvidest lähtuvalt arvestama peaksid.

5.1 Projekti algatamine

Et uue tarkvara loomise projekti oleks üldse võimalik algatada, peab eksisteerima vajadus üht või teist tegevust ülikoolis automatiseerida. Siiani on vastavasisulisi otsuseid teinud ülikooli juhtkond, kes on ülesande realiseerimise delegeerinud mõnele struktuuriüksusele. Edaspidi oleks mõistlik kaaluda analoogselt Tartu Ülikoolile luua nn. "kasutajate komisjon", kelle ülesandeks oleks teha ülikooli juhtkonnale tarkvaraprojektide käivitamise ettepanekuid. Kuna on ilmselge, et kõiki projekte ei saa algatada ja realiseerida ühekorraga, siis peaks kasutajate komisjoni ülesandeks olema ka erinevate tööde prioriteetsuse üle otsustamine.

Projekti algatamisel osalevad:

- TPÜ juhtkond
- (Kasutajate komisjon)

Projekti algatamine sisaldab järgmiseid tegevusi:

- Probleemi püstitamine
- Juhtkonna (või kasutajate komisjoni) otsus projekti algatamiseks
- Ülesannete delegeerimine vastava(te)le struktuuriüksus(t)ele

Projekti järgmise etapina on vaja formuleerida tarkvarale esitatavad üldised soovid ja nõuded s.t. tuleb koostada projektiplaan.

5.2 Projektiplaani koostamine

Projektiplaan peab sisaldama kogu projekti üldist kirjeldust. Teatud aspektid on vaja formuleerida ka juba täpsemalt - et oleks selgelt arusaadav loodava tarkvara otstarve ja vajalikkus. Esialgu pole niisiis oluline kirjeldada IT alaste lahenduste üksikasju, vaid üldisi - kogu ülikooli asjaajamist jmt. puudutavaid - küsimusi. See tähendab, et tarkvaraprojekti taotlejatel peavad olema välja töötatud reaalsed skeemid ja põhimõtted, mis tagavad tarkvaraga seonduvate tegevuste nõuetekohase toimimise. Mõnel juhul võib süsteemi toimimise tagamine eeldada uute normide ja/või eeskirjade väljatöötamist, mis kinnitatakse ülikooli valitsuses või nõukogus.

Projektiplaani koostamisel osalevad:

- Tellija esindaja
- IT osakonna poolne esindaja

Projektiplaani koostamine sisaldab järgmiseid tegevusi:

- Nõuete formuleerimine
- Projektiplaani koostamine (vt. Lisa. 2. Projektiplaan)

Väljuvad dokumendid:

- Nõuetekohaselt vormistatud projektiplaan

Kui projektiplaan on koostatud ning IT osakonnas kõikide nõuete osas kooskõlastatud, siis liigub dokument edasi ülikooli juhtkonna kätte.

5.3 Nõusoleku taotlemine juhtkonnalt

Juhtkonna ülesandeks selles etapis on teostada projektiplaani läbivaatus. Arvestades organisatsiooni poliitikat ja muid eeskirju ning silmas pidades projekti olulisust ja olemasolevaid ressursse [ISO-Tep], otsustatakse projekti realiseerimine. Otsustamisel mängib tähtsat rolli IT osakonna poolne ressursianalüüs, mis annab täpsema ülevaate kavandatava projekti realiseerimise võimalikkusest, vajaminevatest ressurssidest, orienteeruvast ajakavast jmt. Tarkvararakendust on võimalik ka sisse osta, kui juhtkond (või ka IT osakonna juhataja) ei pea vajalikuks selle tootmist TPÜ IT osakonna poolt.

Juhtkonna lõplik nõusolek projekti käivitamiseks peab olema vormistatud ametliku dokumendina vastavalt ülikooli asjaajamiskorras ettenähtud nõuetele (kas nõukogu või valitsuse otsus või määrus).

Projekti käivitamise otsustamisel osalevad:

- TPÜ juhtkond
- IT osakonna poolne esindaja

Otsustamine sisaldab järgmiseid tegevusi:

- Ressursianalüüs IT osakonna poolt
- Ülevaate andmine juhtkonnale olemasolevatest ja vajaminevatest ressurssidest
- Projektiplaani läbivaatus juhtkonna poolt
- Ressursianalüüsi läbivaatus juhtkonna poolt
- Otsuse langetamine

Sisenevad dokumendid:

- Projektiplaan

Väljuvad dokumendid:

- Otsus tarkvaraprojekti edasise käekäigu kohta

5.4 Projektimeeskonna loomine

Üldjoontes defineerib projektiplaan ka meeskonna liikmed ning nende ülesanded. Pärast juhtkonna positiivse otsuse langetamist luuakse plaanis kirjeldatud projektimeeskond reaalselt. Määratakse ka projektijuht. Oluline on, et projektijuhil oleks piisavad volitused ja pädevus langetada otsuseid ja määrata suures osas projekti kulgu. Meeskonda peavad kuuluma nii tellija- kui ka tarnijapoolsed töötajad. Meeskonna suurus võiks jääda piiridesse 8-10 inimest. IT osakonna poolt peaks meeskonda kuuluma vähemalt üks analüütik ja üks disainer-programmeerija, vajadusel ka osakonna juhataja, kes oleks (teatud küsimustes) otsustaja rollis. Tellija poolt peaks olema esindatud projekti algataja ning 3-4 kasutajat, kes osaleksid reaalselt arendusprotsessis (nagu näiteks XP puhul) - annaksid tagasisidet loodava tarkvara kohta, teostaksid funktsionaalset testimist, viiksid läbi koolitusi jne.

Projektimeeskonna loomisel osalevad:

- Projekti algataja
- IT osakonna poolne esindaja

Meeskonna loomine sisaldab järgmiseid tegevusi:

- Projektijuhi määramine
- Projektimeeskonna loomine
- Ülesannete, kohustuste ja vastutuse jagamine meeskonnaliikmete vahel

Sisenevad dokumendid:

- Otsus tarkvaraprojekti algatamise kohta
- Projektiplaan

Väljuvad dokumendid:

- Meeskonna nimekiri koos iga üksiku liikme ülesannete kirjeldusega

Pärast meeskonna loomist algab tarkvaraprojekti järgmine etapp - analüüs ja disaini väljatöötamine.

5.5 Analüüs

Analüüsi esmaseks ülesandeks on hakata täpsustama kasutajate nõudmisi. Selleks tuleb analüütikul läbi viia intervjuusid kasutajatega ning intervjuud dokumenteerida. Kui rakendust hakkavad kasutama kõik ülikooli üksused, siis on väga oluline, et intervjuudesse oleks kaasatud kõikide osakondade esindajad ja kõik saaks oma soovid edasi anda. Järgmisena on väga tähtis läbi mõelda ja analüüsida kõikvõimalikud piirsituatsioonid, riskid ja nende käsitlemine. Riskide analüüsi puhul tuleb tähelepanu pöörata ka infoturbe küsimustele.

Analüüsi etapis võib info kogunedes hakata koostama tarkvara esialgseid mudeleid ja skeeme.

Nii tekkib tasapisi üldpilt kasutajate soovidest. Esialgu ei oleks mõttekas kasutajalugude kirjutamist nõuda, kuna üldjuhul kasutajatel puudub täpne visioon loodavast rakendusest ja tõenäoliselt ka arusaam, mida kasutuslugu endas sisaldama peaks. Samas on võimalik intervjuude käigus selgitada kasutajalugude olemust ning soovijad võivad proovida nende kirjutamist. Nii võib hakata järk-järgult kujundama ülikoolis harjumust esitada nõudmised tarkvarale ühe- või kahelauseliste lugudena. Mida rohkem rakendusi kasutusele võetakse, seda paremini hakkab tekkima ka ettekujutus, milline üks rakendus peaks välja nägema ja millised on need nõudmised, mida ühele rakendusele üldse esitada saab. Kasutuslugude ja intervjuude põhjal oleks mõistlik koostada spetsifikatsioon, milles on nimetatud kõik nõutud funktsioonid. Taoline loetelu aitab saada parema ülevaate tarkvara mahust ning täpsustada projekti eelarvet. Saadud info peaks lisanduma projektiplaanile.

Kui projektiplaan on lõplikult valmis, siis kirjutatakse see tellija ja tarnija poolt alla ning see saab ametlikuks lepinguks, mis sätestab töö mahu, tähtajad, rakendusele esitatavad nõudmised jm. ning seda täiendada ja muuta võib ainult poolte kokkuleppel.

Analüüsi teostamisel osalevad:

- Analüütik
- Disainer-programmeerija
- Infoturbe spetsialist
- Projektijuht
- Kasutajad

Analüüs sisaldab järgmiseid tegevusi:

- Kasutajate intervjueerimine
- Riskide analüüs
- Projektiplaani täiendamine riskide kirjelduse, kasutajalugude, funktsioonide spetsifikatsiooni ja eelarve osas
- Lepingu sõlmimine tellija ja IT osakonna vahel

Väljuvad dokumendid:

- Kasutajate intervjuud
- Riskide kirjeldused, nende maandamise ettepanekud
- Tarkvara disaini üldskeemid ja -mudelid
- Poolte vahel allkirjastatud projektiplaan

5.6 Tarkvara disain, kasutajaliidese väljatöötamine

Pärast lepingu sõlmimist alustatakse üksikasjalikuma tarkvara disaini väljatöötamisega. Disaini käigus peaks valmima ka andmemudel. Andmemudeli loomisel tuleb arvestada

olemasolevate registrite[4] ja andmepankadega[5] - ning kui vähegi võimalik, siis ainult olemasolevaid kasutada. Vajaduse korral võib asutada uusi nimistuid[6] (vt. alamjaotus 5.7 Nimistu asutamine).

Ka kasutajaliides ning vajadusel graafiline disain ja vastavad prototüübid luuakse selles etapis. Kui tellijalt on saadud kooskõlastus, siis tehakse valmis reaalsed kasutajaliidese vormid, mida on võimalik programmeeritava rakendusega siduma hakata.

Tarkvara disaini ja kasutajaliidese väljatöötamisel osalevad:

- Analüütik
- Disainer-programmeerija
- Graafilise disaini looja (vajadusel)
- Projektijuht
- Kasutajad

Tarkvara disaini ja kasutajaliidese väljatöötamine sisaldab järgmiseid tegevusi:

- Graafilise disaini prototüüpide loomine
- Tarkvara disaini täpsem väljatöötamine, programmeerimine
- Kooskõlastamine kasutajatega
- Graafilise disaini reaalne loomine

Sisenevad dokumendid:

- Kasutajate intervjuud
- Skeemid, mudelid

Väljuvad dokumendid:

- Graafilise disaini ekraanipildid

5.7 Nimistu asutamine

Nagu juba eelpool öeldud, läbitakse nimistu loomise etapp ainult juhul, kui olemasolevate andmekogude andmed ei rahulda kõiki uue tarkvara vajadusi. Uue nimistu asutamine toimub vastavalt kehtivale "TPÜ andmekogude eeskirjale". Nimistu asutamiseks tuleb vormistada vastav õigusakt, kus asutaja määrab kindlaks andmekogu asutamis- ja kasutuseesmärgi, nimetuse, ülesehituse, vastutava isiku, kes hakkab andmeid töötlema, andmekogusse kantavad

Vastavalt TPÜ nõukogus 15.12.2003 kehtestatud määrusega nr 8 - "Tallinna Pedagoogikaülikooli andmekogude eeskiri" - on märgitud mõisted defineeritud järgmiselt:

4 **Register** on ülikooli või asutuse info- ja süsteemitöö korralduse, meetodite ja vahendite kogusumma andmekogu pidamisest.

5 **Andmepank** on mingi teemaga seotud ja selliselt korraldatud andmete kogum, et seda on hõlbus kasutada.

6 **Nimistu** on väikseim korrastatud kogum andmetest, suvalise konkreetse või abstraktse asja, kaasa arvatud nende asjade ühenduste identifitseerimiseks.

andmed, andmekogu pidamise üle järelvalve teostamise korra jm. andmekogu kohta käivad tingimused. Asutamise akti vormistamiseks ja registreerimiseks on vaja täita lisas toodud vormid. Nimistu asutamise otsustab struktuuriüksuse juht.

Andmekogude täiendamisel osalevad:

- Analüütik
- IT osakonna (või mõne muu struktuuriüksuse) juhataja
- Andmekogude registri vastutav töötleja

Nimistu loomine sisaldab järgmiseid tegevusi:

- Nimistu moodustamise ettepaneku tegemine
- Nimistu asutamiseks vajaliku analüüsi teostamine
- Nimistu dokumentatsiooni koostamine
- Nimistu moodustamiseks loa taotlemine
- Nimistu moodustamine

Sisenevad dokumendid:

- Nimistu asutamise vormid (vt. Lisa. 3. Nimistu vormid)

Väljuvad dokumendid:

- Nimistu dokumentatsioon

5.8 Kodeerimine ja testimine

Kodeerimise etapis toimub reaalse rakenduse kirjutamine. See etapp peaks olema jaotatud iteratsioonideks ning iga iteratsiooni lõpuks peaks valmis ja testitud olema teatud osa funktsionaalsustest. Põhimõtteliselt peaks ka koodi ja testide dokumenteerimist alustama kodeerimise esimestest päevadest alates, kuid siiamaani on selle elluviimine osutunud raskesti teostatavaks. Pealegi on koodi dokumentatsiooni loomine automaatne ning kui kood on esimestest päevadest alates nõuetekohaselt kommenteeritud, siis peaks koodi dokumentatsiooni loomine olema võimalik igal ajahetkel, kui selleks tekkib vajadus. Üldine - käsitsi loodav - dokumentatsioon peab sisaldama üldosa, funktsioonide spetsifikatsiooni (mis on koostatud analüüsi käigus ja vajadusel täiendatud), andmebaaside skeeme, kasutajajuhendit, installeerimise, deinstalleerimise ja varundamise juhendeid. Dokumentatsioonile esitatavad nõuded on toodud lisas.

Testimise eesmärgiks on kontrollida tarkvara vastavust tellija nõudmistele, töökindlust ning kasutusmugavust. Üldjuhul viiakse läbi järgmised testid: tekstipõhine- ja funktsionaalsustest, koormustest ja ühilduvustest, kasutajaliidese- ja kasutatavuse test. Üldjuhul peaks testid koostama ja need läbi viima kvalifitseeritud testija. Kasutatavust, kasutajaliidest ja -juhendeid testivad tarkvara lõpp-kasutajad. Kõikide testide tulemused peavad olema dokumenteeritud.

Kodeerimisel ja testimisel osalevad:

- Disainer-programmeerija
- Kodeerija
- Testija
- Projektijuht
- Kasutajad

Kodeerimise ja testimise etapp sisaldab järgmiseid tegevusi:

- Rakenduse koodi kirjutamine
- Rakenduse testimine
- Dokumentatsiooni koostamine

Sisenevad dokumendid:

- Funktsioonide spetsifikatsioon
- Kasutajalood (kui on olemas)
- Andmemudelid, muud skeemid

Väljuvad dokumendid:

- Tarkvara rakenduse dokumentatsioon (vt. Lisa. 4. Tarkvara üldine dokumentatsioon)
- Tarkvara testimise dokumentatsioon

5.9 Juurutamine

Juurutamine algab valmisprodukti integreerimisega olemasolevasse süsteemi. Esimese sammuna viiakse tarkvara arenduskeskkonnast üle töökeskkonda ning testitakse selle töökindlust. Kui süsteem töötab vigadeta, siis on võimalik selle kasutusseandmine. Algab koolituste ja/või infotundide korraldamine. Kui tegemist on väga suurt sihtrühma puudutava rakendusega, siis on kasulik trükkida ja levitada ka rakendust tutvustavat infomaterjali. Mõnda aega peaks kasutajatele kättesaadav olema juurutusjuhina töötav projektimeeskonna liige, kes vastab küsimustele ja abistab probleemide tekkimisel. Juurutusjuhi ülesandeks peaks olema ka tarkvaras ilmnevate vigade programmeerijatele vahendamine. Juurutamise etapis (või ka varem) peaks kinnitatama kõik eeskirjad ja korrad, mis reguleerivad antud rakenduse kasutamist. Vastasel juhul võib motivatsioon antud rakendust kasutada jääda liiga väikseks.

Esmakordsel installeerimisel peaksid nii kasutajad kui ka projekti muud osalised arvestama sellega, et tegemist on tarkvara beeta-versiooniga, kus kasutamise käigus võib veel ilmneda vigu. Beeta-versioon peaks kasutusel olema keskmiselt kaks kuud, mille jooksul toimub rakenduse parandamine ja täiendamine. Kahe kuu möödudes ilmutatakse valmisprodukt ning antakse see lõplikult kasutusse.

Juurutamisel osalevad:

- Projektijuht
- TPÜ juhtkond
- Süsteemi administraator
- Disainer-programmeerija
- Juurutusjuht
- Projektimeeskonna tellijat esindavad liikmed
- Kasutajad

Juurutamine sisaldab järgmiseid tegevusi:

- Rakenduse töökeskkonda integreerimine
- Rakenduse kasutamine (põhieesmärgiga leida vigu)
- Normdokumentide vastuvõtmine
- Vigade ja paranduste dokumentatsiooni koostamine
- Korduma kippuvate küsimuste koostamine

Sisenevad dokumendid:

- Tarkvara tehniline dokumentatsioon
- Kasutajajuhend

Väljuvad dokumendid:

- Leitud vigade ja tehtud paranduste loetelu
- Rakenduse kasutamist reguleerivad normdokumendid

Pärast tarkvara lõplikku kasutuselevõttu loetakse projekt lõpetatuks ning edasisi muutuseid saab teha juba järgmises - muudatuste teostamise - etapis.

5.10 Muudatuste teostamine

Tarkvarasse muudatuste tegemise algatamiseks tuleb vormistada vastav dokument - "Muudatuse taotlus" - ning kooskõlastada see TPÜ juhtkonna (või kasutajate komisjoni) ning IT osakonnaga. Nagu juba eelpool mainitud, tuleb eristada kahte tüüpi muudatusi, mille käsitlemine on erinev: kriitilise vea parandamine või funktsionaalsus(t)e lisamine. Kriitilise vea leidmisel tuleb reageerida võimalikult operatiivselt ning sel juhul ei saa juttu olla pikaleveniva tarkvaraprojekti algatamisest - viga tuleb parandada ja uus redaktsioon installeerida nii kiiresti kui võimalik. Loomulikult ei tohi unustada, et ka vea parandamisel tuleb läbida analüüsi, kodeerimise, testimise jm. etapid. Tarkvara täiendamiseks või funktsionaalsuste lisamiseks tuleb aga algatada uus eraldiseisev projekt, mis sisaldab kõiki eelpool loetletud tarkvaraarenduse etappe: nõusoleku taotlemist juhtkonnalt, analüüsi, disaini, kodeerimist, testimist, juurutamist jne.

Muudatuse teostamisel osalevad:

- Muudatuse algataja
- TPÜ juhtkond (või kasutajate komisjon)
- IT osakonna poolne esindaja
- Analüütik
- Disainer-programmeerija

Muudatuse teostamine sisaldab järgmiseid tegevusi:

- Muudatuse tüübi määramine ja analüüs
- Muudatuse teostamine

Sisenevad dokumendid:

- Muudatuse taotluse vorm (vt. Lisa. 5. Muudatuse taotlus)

Väljuvad dokumendid:

- Parandatud (või täiendatud) redaktsioon
- Täiendatud dokumentatsioon

Eelpool on niisiis kirjeldatud kõik TPÜ tarkvara arendusprotsessiga seotud põhitegevused ja -dokumendid. Muudatuste teostamise protsess algatab sisuliselt uue tarkvaraprojekti ning seega tuleb taas alustada TPÜ juhtkonnalt nõusoleku taotlemisega, et liikuda edasi järgmistesse etappidesse.

KOKKUVÕTE

Käesoleva bakalaureusetöö raames on välja töötatud TPÜ tarkvara arenduse üldpõhimõtted, mille eesmärgiks on muuta tarkvaraarendusprotsess süstematiseeritumaks ja tulemusrikkamaks ning tõsta loodavate rakenduste kvaliteeti. On üsna selge, et soovitud tulemuste saavutamiseks ei piisa ainuüksi tarkvaraarendusega seonduvate tegevuste määratlemisest. Põhimõtete väljatöötamine on alles esimene samm arendustöö parendamise protsessis. Tõenäoliselt osutub märksa vaevarikkamaks koostatud ettepanekute aktsepteerimine ja töösserakendamine kogu arendusmeeskonna (ja ka teiste tarkvaraarendusega kokkupuutuvate töötajate) poolt. Tähtsaimaks siinkohal on teadvustamine, et nõudeid ei juurutata mitte nõuete enese pärast ning et nende esitamine ei ole järjekordne katse suurendada ülikooli bürokraatiat. Eesmärgiks on parandada töökorraldust ning loodavate rakenduste kvaliteeti - seega peaks käesolevate nõuete rakendamine olema kõikide osapoolte huvides.

Kuna koostatud põhimõtete puhul on tegemist alles esialgsete ettepanekutega, siis peaks juurutamise käigus pidevalt jälgima süsteemi efektiivsust ning vastavalt ilmnevatele vajadustele seda muutma ja täiendama. Nii kujuneb järk-järgult välja sobivaim nõuete pakett, mis on teatud hetkel võimalik formuleerida "Tarkvaraprojektide eeskirjaks" ja kinnitada ülikooli juhtkonna poolt. (Analoogse näitena võib tuua "Andmekogude eeskirja", mis on juba TPÜ-s vastu võetud).

Käesoleva bakalaureusetöö edasiarendusena oleks võimalik analüüsida tehtud ettepanekute efektiivsust reaalsete tarkvaraprojektide elluviimisel. Esmalt tasuks jälgida, millised põhimõtted on mõjutanud töö kvaliteeti paremuse suunas ja millised nõuded on olnud pärssivad. Ka tagasiside saamine (ja analüüsimine) projektimeeskonna liikmetelt, kasutajatelt ja juhtkonnalt on väga tähtis, saamaks terviklikku ja objektiivset ülevaadet projektide tulemuslikkusest. Saadud uutele kogemustele ja teadmusele tuginedes oleks seega võimalik teha täiendavaid ettepanekuid ja nõuete edasiarendusi.

KASUTATUD KIRJANDUS

1. [Isajenko2003] **Igor Isajenko,** Infosüsteemi arendamise dokumentatsioon. Juhend, 2003, Tallinna Pedagoogikaülikool, Tallinn.
2. [Normak2003] **Peeter Normak**, Projektijuhtimine. Loengukonspekt, 2003, Tallinna Pedagoogikaülikool, Tallinn.
3. [Petuhhov2004] **Inga Petuhhov,** UP (*Unified process*). Loenguslaidid, 2004 Tallinna Pedagoogikaülikool, Tallinn.
4. [Piho2003] **Gunnar Piho**, "XP-metoodika juurutamisest väikeses eesti tarkvarafirmas", Magistritöö, 2003, Tallinna Pedagoogikaülikool, Tallinn.
5. [E-teatmik] **Heikki Vallaste,** Inglisekeelsete info- ja sidetehnoloogia terminite seletav sõnaraamat, 2000-2004.
   http://www.vallaste.ee
6. [ISO-Tep] EVS-ISO/IEC 12207:1998. **INFOTEHNOLOOGIA Tarkvara elutsükli protsessid.** Eesti standard, 1998.
7. [Gornik2003] **Davor Gornik,** IBM Rational Unified Process: Best Practices for Software Development Teams, 2003.
   http://www3.software.ibm.com/ibmdl/pub/software/rational/web/whitepapers/2003/rup_bestpractices.pdf
8. [Pollice2003] **Gary Pollice,** Using the IBM Rational Unified Process for Small Projects: Expanding Upon eXtreme Programming, 2003.
   http://www3.software.ibm.com/ibmdl/pub/software/rational/web/whitepapers/2003/tp183.pdf
9. [XP] Extreme Programming: A gentle introduction.
   http://www.extremeprogramming.org

NB!: Loetelus esinevad viited Internetti võivad olla oma kehtivuse kaotanud!
Viimati kontrollitud 30. aprillil 2004.

SUMMARY

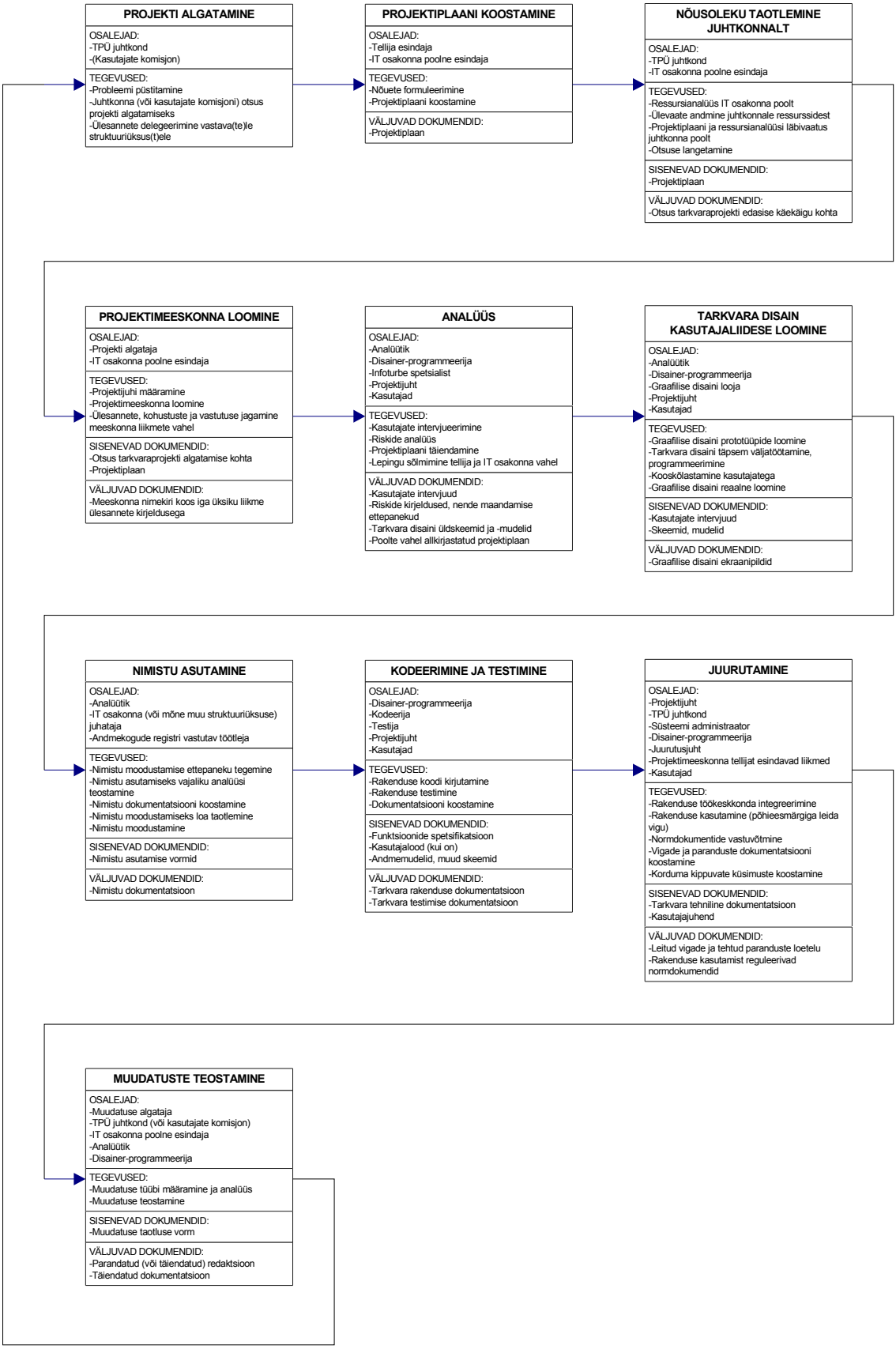
As the use of computers and the corresponding information technology (IT) solutions has increased its foothold in everyday life, the demand for new IT applications has gained significant momentum. Similar developments have occurred at Tallinn Pedagogical University (TPU), where an increasing number of processes are being carried out electronically instead of using the traditional hardcopy. As the number of software development projects and teams is booming, it is essential to implement new guidelines and standards, which regulate the software development process and increase the operating effectiveness of IT department and potential customers. When the software development process at TPU is measured on the SW-CMM (*Capability Maturity Model for Software*) scale it appears, that there is room for significant development. Most of customer's and supplier's activities are obscure, which has caused several misunderstandings as several software projects have either failed partially or as a whole.

The goal of this Bachelor's thesis is to develop general principles for accelerating the performance of software development processes and for increasing the quality of new applications at TPU. In addition to the aforementioned, this thesis outlines the forms of different applications and other documents, which would enable the development of a coherent documentation system for all software packages.

The thesis consists of three parts. The first part (Chapters 2 - 4) covers the description and analysis of standard ISO 12207 - "Information technology - Software life cycle processes" and two software development methodologies - XP (Extreme Programming) and RUP (Rational Unified Process). The author is analyzing the possibilities (and impossibility) of implementing those three in software development processes at TPU based on prior experiences. All three point out, that they must not be followed in detail. The goal of a software producing organization is to determine all necessary requirements and implement only those during the software development process. The second part (Chapter 5) outlines the principles on which all future software projects at TPU should be based. The third part (Appendix) provides a description of the standards and forms to be used during software development at TPU.

LISAD

Lisa. 1. TPÜ tarkvaraarendusprotsessi tegevusskeem



Lisa. 2. Projektiplaan [Normak2003]

Projekti nimi	
Kuupäev	
Tellija	
Tellija kontaktandmed	
Soovitav tähtaeg	
PÕHIANDMED	
Eesmärk	
Osalevad institutsioonid	
TAUST	
Vajadus	
Momendi olukord	
Valdkonna prioriteetsus	
Sihtrühmad	
KIRJELDUS	
Üldinfo	
Nõutavad funktsionaalsused	
Piirangud	
HALDAMINE	
Täitjate iseloomustus	
Ülesannete jaotus	
Töö- ja aruandluse korraldus	
Osapoolte vastutus	
Riskide käsitlus	
TULEMI KIRJELDUS	
Tulemuste rakendamise ja/või levitamise kavad	
Hinnang majandusliku vm. mõju osas, mis tuleneb projekti eesmärkide saavutamisest	
Kriteeriumid projekti edukuse hindamiseks	
KOOSKÕLASTUS	
Tellija allkiri ja kuupäev	
Tarnija allkiri ja kuupäev	

Projektiplaani lisadena esitatavad või loodavad dokumendid:

- Rakenduse kasutamist reguleeriva eeskirja või määruse tööversioon (vormistatud vastavalt TPÜ-s kehtivatele nõuetele).
- Ajagraafik.
- Eelarve.
- IT osakonna juhataja ressursianalüüs - s.h. projekti orienteeruv maksumus, aeg, vajalikud ressursid jmt. (esitada kirjeldusena).
- TPÜ juhtkonna otsus projekti teostamise kohta.
- Kasutajalood, funktsioonide loetelu.
- Riskide kirjeldus.

Vorm 1. Andmekogu üldandmed

1. Andmekogu nimetus:

Kommentaar: Andmekogu on ülikooli riiklikest õigusaktidest tulenevate ja/või ülikooli põhikirjaliste ülesannete täitmiseks vajalike, ülikooli või asutuse poolt peetav korrastatud andmete kogum ülikooli kui terviku toimimise ja juhtimise vajaduseks, mille pidamisel kasutatakse automatiseeritud andmetöötlust või mida peetakse käsitsi ja korrastatud vormidel ning mis võimaldavad andmetega lihtsat tutvumist või nende mehhaanilist töötlemist. Eristatakse järgnevaid andmekogusid:
1) nimistu, mis on väikseim korrastatud kogum andmetest, suvalise konkreetse või abstraktse asja, kaasa arvatud nende asjade ühenduste identifitseerimiseks;
2) andmepank, mis on mingi teemaga seotud ja selliselt korraldatud andmete kogum, et kasutajad saavad selle poole pöörduda;
3) register, mis on ülikooli või asutuse info- ja süsteemitöö korralduse, meetodite ja vahendite kogusumma andmekogu pidamisest.

2 Andmekogu liik

	Register
	Andmepank
	Struktuuriüksuse nimistu
	Asutuse andmekogu
	Muu

Kommentaar: Valige õige variant. **Register** on ülikooli nõukogu otsusega asutatud, ülikooli töötajatele ja üliõpilaskonnale kasutamiseks määratud andmekogu, mida peetakse üldistes huvides ülikooli kõige olulisemate põhiülesannete täitmiseks. **Andmepank** mõistes on ühe või mitme registri või struktuuriüksuse ülesannete täitmiseks vajalik andmekogu. **Struktuuriüksuse nimistu** on ülikooli struktuuriüksusele põhimääruse või muu õigusaktiga pandud ülesannete täitmiseks või ülikooli struktuuriüksuse töö korraldamise tagamiseks vajalik andmekogu. **Asutuse andmekogu** on asutusele põhikirjaga või muu õigusaktiga pandud ülesannete täitmiseks või töö korraldamise tagamiseks vajalik andmekogu.

3 Andmekogu asutamis- ja kasutuseesmärgid

Vorm 2: Andmekogu vastutav ja voolitatud töötleja

1 Andmekogu nimetus:

2 Andmekogu vastutav töötleja:

Eesnimi	Perekonnanimi	Organisatsiooni nimi	Struktuuriüksus	Ametikoht	Asukoht	Telefon	E-mail

Kommentaar: Andmekogu omanikuks võib olla ülikool või tema asutus. Ülikooli või asutuse **andmekogu vastutav töötleja** on andmekogu omaniku käskkirjaga või korraldusega määratud struktuuriüksuse juht, kes vastutab andmekogu pidamise õiguslikkuse eest, korraldab andmekogu ja tema osade projekteerimiseks ja kasutuselevõtmiseks vajalike tööde läbiviimise ja vastuvõtmise, juhib andmekogu pidamist, teostab andmekogu pidamise üle järelevalvet ning lahendab eeskirjas sätestatud ulatuses ja korras vaidlusi andmekogu pidamisel tekkinud küsimustes.

3 Andmekogu volitatud töötleja:

Eesnimi	Perekonnanimi	Organisatsiooni nimi	Struktuuriüksus	Ametikoht	Asukoht	Telefon	E-mail

Kommentaar: Ülikooli või asutuse **andmekogu volitatud töötleja** on isik, kes peab andmekogu või töötleb andmeid andmekogu vastutava töötleja tellimusel, ametijuhendis ja õigusaktis ettenähtud ulatuses.

Vorm 3.1 Andmekogu ülesehitus ja andmed (antud vorm on ettenähtud struktuuriüksuse nimistu ning asutuse andmekogu jaoks)

1 Andmekogu nimetus

3 Andmekogu tabelite struktuur / Andmekogusse kantavad andmed

Tabeli nimetus	Tabeli veeru nimetus	Kantavate andmete nimetus	Selgitus

Vorm 4 Andmekogu pidamise üle järelevalve teostamise kord;

1 Andmekogu pidamise üle järelevalve teostamise kord

Etapp	Teostaja	Kirjeldus

Vorm 5 Muud andmekogu kohta käivad tingimused.

1 Andmekogu nimetus:

2 Muud andmekogu kohta käivad tingimused.

Nr	Teostaja	Kirjeldus

Lisa. 4. Tarkvara üldine dokumentatsioon

Tarkvara nimetus	
Versiooni number	
SISSEJUHATUS	
Eesmärk	
Ülesanne	
Tellija	
Programmi asukoht	
Kasutajad ja nende õigused	
Vajalikud ressursid	
FUNKTSIOONID	
Funktsioonide spetsifikatsioon	
ANDMEBAAS	
Andmebaasi skeem (NB! Skeemil peavad olema arusaadavalt välja toodud seosehulgad ja nende vastenduse võimsus)	
Side olemasolevate andmekogudega	
RIIST- JA TARKVARALISED NÕUDED	
Andmebaasi server: (Operatsioonisüsteemi nimetus ja versioon (piirangud). Andmebaasi mootor ja versioon.)	
Rakenduste server: (Operatsioonisüsteemi nimetus ja versioon (piirangud). Andmebaasimootori nimetus ja versioon.)	
Veebiserver: (Operatsioonisüsteemi nimetus ja versioon (piirangud). Lisamoodulite loetelu.)	
Kliendi töökoht: (Erinõuded - nt. veebibrauseri nimetus ja versioon.)	
INSTALLEERIMINE / DEINSTALLEERIMINE / VARUNDAMINE	
Installeerimine	
Deinstalleerimine	
Varundamine	

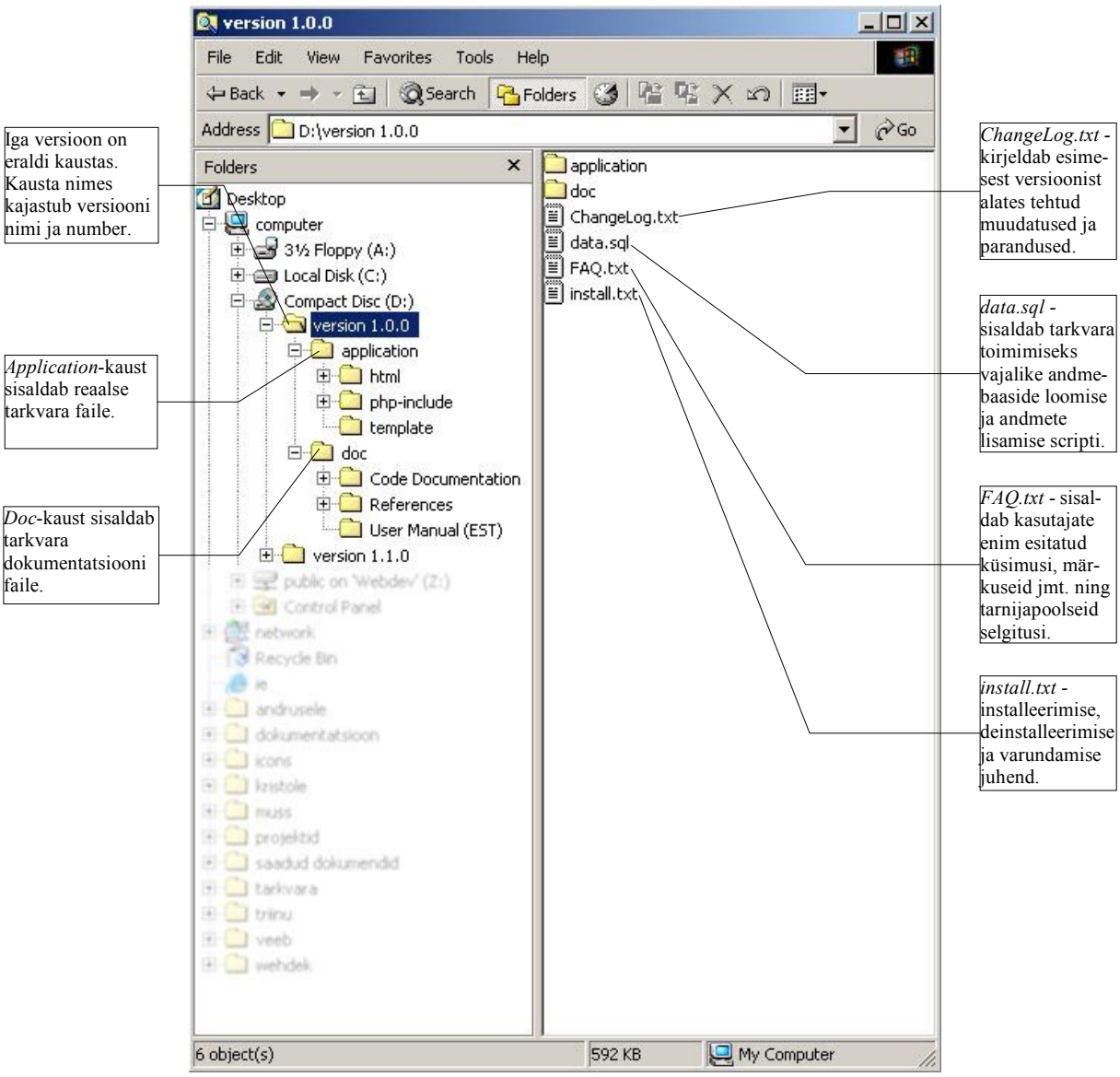
Dokumentatsiooni lisadena esitatavad või loodavad dokumendid:

- Kasutajajuhend (s.h. ikoonide, nuppude, terminite seletus, funktsioonide kirjeldus jms.).
- (Elektrooniliselt koostatud) koodi dokumentatsioon.

Lisa. 5. Muudatuse taotlus

Tarkvara nimetus	
Kuupäev	
Tellija	
Tellija kontaktandmed	
Soovitav tähtaeg	
PROBLEEMI KIRJELDUS	
Vigade kirjeldus	
Uued funktsionaalsused (Uute funktsionaalsuste puhul põhjendada nende vajalikkust)	
Märkused	
KOOSKÕLASTUS	
Tellija allkiri ja kuupäev	
Tarnija allkiri ja kuupäev	

Lisa. 6. Tarkvarapaketi struktuur ja versioonide nummerdamine



Versiooni tähis koosneb kolmekohalisest numbrist:

1 koht - näitab versiooni numbrit (nt. 1.0.0 on esimene redaktsioon ja 2.0.0 on järgmine redaktsioon uute võimalustega)

2 koht - näitab muudatusi andmebaasis (nt. redaktsiooni 1.1.0 puhul on muudatusi tehtud andmebaasides)

3 koht - näitab tarkvara veaparandusi (nt. redaktsioonis 1.0.1 on parandatud tarkvaras esinevaid vigu)