Tallinna Tehnikaülikool

Prof. J.Tepandi loengusarja

"TARKVARA KVALITEET JA STANDARDID"

loengukonspekt

Konspekteerisid ja sisestasid arvutisse (1993):
A.Tamboom (LYDS71)ja M.Kandelin (LYDS7
)
Täiendas (1994):
Mait Tõnisson (LYDS71)

Tallinn 1994

Sisukord

Error! Bookmark not defined.

Kursuse hindamine

TÖÖD ja PUNKTID (max):

1. iseseisev töö	20 punkti
2. iseseisev töö	20 punkti
analüüs	10 punkti
ettekanne	10 punkti
veaotsing	10 punkti
eksam	5x10=50 punkti
	+ ebastandardsed punktid

---

arvestatava hinde saamiseks peab kursuse hinde moodustavate punktide hulgas peab olema minimaalselt:

25 punkti eksamilt
25 punkti mujalt

PUNKTIDE VASTAVUS HINDELE:

50p...hinne 1
:
90p...hinne 5

Ettekanne ja selle hindamine

TEEMAKS VÕIB OLLA:

- iseseisev töö

ja • lisateema loengule
või • veaotsing
või • iseseisva töö hindamine on teineteist välistavad teemad.

ESITUS:

Ettekanne esitatakse semestri jooksul kas loengul või praktikumil.
Kuulajaskonna suurus peab olema vähemalt 5 tudengit pluss õppejõud.
Ettekande pikkus on 5..15 minutit, millele järgneb arutelu ca 10 minutit.
Ühel loengul saab esineda 2..3 inimest.

ETTEKANDE HINDAMINE:

Baashinne ettekande eest on maksimaalselt 5.
Hinnatakse sisu (Max 3 punkti) ja vormi (max 2 punkti).
SISU puhul vaadatakse:

- probleemi tegelikkust,kas on tegemist laiatarbepaketiga? (1;0.5;0)
- süsteemsete meetodite kasutamine testimisel (1;0.5;0)
- oma töö maht (1 nädal=1punkt)

VORMI juures peetakse silmas:

- esituse selgust, struktueeritust (max 1punkt)
- ettekandele kuunud aega, abivahendite kasutamist, kuuldavust (max 1punkt)

LISAPUNKTID:

Lisapunkte ettekande eest võib saada järgmiselt:

- esitus oktoobris, või varem ⇒ punktid korrutatakse 3-ga
- esitus peale oktoobrikuud ⇒ punktid korrutatakse 2-ga

Kaastudengi töö hindamine

Töö hinnang koosneb järgmistest osadest:

1. Hindaja andmed

- Ees- ja perekonnanimi.
- Õpperühma nimetus.
- Number professor Jaak Tepandi registreerimislehel.

2. Hinnatava projekti andmed

- Autori nimi, number ja rühm.
- Projekti nimi ja lühike sisukirjeldus. (Soovitav oleks kogu töö lisamine või eksamil esitamine).
- Projekti staatus (Üleandmise kuupäev).
- Läbivaatuse sisendid (Tekst, Dokumendid,..).

3.Töö analüüs (töö põhiosa)

- Kas testolukord loodud
- Kas ülesanded lahendatud.
- Kas kõik osad olemas.
- Hinnang vormistusele.
- Hinnang töö mahule.

4. Soovitus tööhinde punktisumma kohta

- 1. Töö (Max. 20 p.).
- 2. Töö (Max. 20 p.).

Arvestatakse et töö on antud ära õigeaegselt, s.t. punkte antakse 100%.

Iseseisvad tööd

Kursuses loetava materjali põhjal tehtavad kaks iseseisvat tööd moodustavad ühtse terviku.

TÖÖ EESMÄRK:

Kogemus testimisest

I. TÖÖ KOOSNEB:

- sisukord,
- sissejuhatus,
- tiitelleht,
- plaan,
- projekt,
- ülesanne

II. TÖÖ KOOSNEB

- probleemid,
- vead,
- kokkuvõte,
- kirjandus,
- lisad

TÖÖDE HINDAMISEL ARVESTATAKSE JÄRGMISTE NÕUETEGA:

Probleem: Tegelikus. Selgus. Testolukord loodud selgelt, realistlik.
Tarkvara: Mittetriviaalne
Meetod: Milline koos põhjendusega
Testimine: Arvutil, põhjalik
Kokkuvõte: Lähtub probleemist, realistlik
Taust: Võimalikud küsimused tööst vastatud
Struktuur: Osad olemas, arusaadav
Välimus: Korrektne, prinditud ja köidetud

Tööd dokumenteeritakse vastavalt *ANSI/IEE Std 829-1983 Standard of software test documentation*'ile

ANSI/IEE Std 829-1983 Standard for Software Test Documentation

* 1.TEST PLAN	-- testimise plaan
* 2.TEST-DESING SPECIFICATION	-- testimise detailprojekt. Kuidas? Millised testid?
3.TEST-CASE SPECIFICATION	-- testide spetsifikatsioon, iga testi kohta eraldi.
4.TEST-PROCEDURE SPECIFICATION	-- testimise protseduuri spetsifikatsioon
* 5.TEST-ITEM TRANSMITTAL REPORT	-- testitavate objektide üleandmise aruanne
6.TEST LOG	-- testimise käik (kuidas läks)
* 7.TEST-INCIDENT REPORT	-- testprobleemide aruanne
* 8.TEST-SUMMARY REPORT	-- testimise kokkuvõte

Vaatluse alla võtame tärniga tähistatud dokumendid.
I iseseisev töö koosneb dokumentidest: 1, 2 ja 5 ning II iseseisev töö dokumentidest 7 ja 8.

Test plan

Kui olukord keeruline ja pabereid palju, siis on standardid kasulikud. Need annavad lähte ja pidepunkte tarkvara testimise kohta.

1. Test-plan identifier

On igal dokumendil. See peab kajastama olukorda lühidalt. Selle järgi on asja hea ära tunda. Ta on dokumendi märgend. Näiteks: RMTP-TEST-PLAN-TEST-1,TEST-2 jne.

2. Introduction

Sissejuhatuses selgitatakse probleemi ümbrust. Kes algatas, kelle jaoks, mida testitakse.

3. Test items

Ülevaade testitavatest objektidest. Mida testitakse. Viited dokumentidele, kust võetakse testimise andmed.

4. Features to be tested

Omadused, mida testitakse. Sisend, väljund.

5. Features not to be tested

Omadused mida ei testita. On vahel omadusi mida pole vaja testida.

6. Approach

Testimise, katsetuse meetod. Milliseid ja missuguse meetodiga testime.

7. Item pass/fail criteria

Testi läbimise kriteerium. Kriteeriumid mille puhul testimine on läbitud ja toode vastu võetud.

8. Suspension criteria and resumption requirements

Testimise katkestamise ja lõpetamise tingimused. Märgitakse ära olukorrad mille puhul pole mõtet testi jätkata.

9. Test deliverables

Testimise tulemused. Näidatakse ära, materjalid mis antakse üle testimise lõppedes.

10. Testing tasks

Testimise ülesanded. Mida vaja teha, et testimise juurde asuda. (Ettevalmistused, kooskõlastused, koolitus...)

11. Enviromental needs

Nõuded testimise ümbrusele

12. Responsibilities

Testimise vastutused. Näidatakse ära kes millise töö osa eest vastutab. Arendajad on vastutavad vigade eest. Tellija olukorra eest.

13. Staffing and training needs

Inimeste ja koolituste vajadus. On ehk vaja mõnda inimest kes on asja juures. Vastutus.

14. Schedule

Ajakava. Mis millal toimuma hakkab ja millal valmis saab.

15. Risks and contingencies

Riskid ja ootamatused. See puudutab testimist ennast. Mis juhtub siis, kui test mingil põhjusel nurjub. Antakse hinnang projekti edukuse kohta.

16. Approvals

Osapoolte allkirjad ja kooskõlastused.

Test-Design Spetsification

1. Test-disign-spetsification identifier

See on eraldaja, mis on antud dokumendi nimetuseks. Hea, kui on süsteemne nimetus.

2. Features to be tested.

Põhjalikult kirja pandud, et mida testitakse. (Kogu dokument peab olema enam-vähem iseselgitav. Kommentaarid ja viidad) Omadused ja alaomadused.

3. ***Approach refinements***

Meetodite täpsustus. Millised testid, milliste meetoditega tehakse.

4. Test identification.

Pannakse kirja testi idee. Sisend ja väljund ning oodatud tulemus. Ühesõnaga loetakse testid ükshaaval üles.

5. Features pass/fail criteria

Ükshaaval kirja pandud läbimise võimalused. Et kogu objekt oleks vastu võetud, peaksid olema rahuldatud selles punktis kirjeldatud omadused.

Test-Item Transmittial Support

Siia pannakse kirja testdokumentide üleandmisega seotud asjad.

1. Transmittal-report identifier

Üleandmise aeg.

2. Transmitted items

Üleantava objekti kirjeldus.

3. Location

Asukoht

4. Status

Lähteolukord, millises olukorras on objekt.

5. Approvals

Allkirjad.

Test-Case Specification

Testi ettevalmistamine

1. Test-case-specification identifier

Identifikaator.

2. Test items

Milliseid objekte testitakse. Mida vaja testida.

3. Input specifications

Testiks vaja minevaid sisendandmeid

4. Output specifications

Oodatavad väljundandmed

5. Enviromental needs

Vajadused ümbrusele. Tarkvara, riistvara, andmed jne.

6. Special procedural requirements

Eriprotseduuride nõuded

7. Intercase dependencies

Testide vahelised seosed. Mis millises järjekorras teha.

Test-Incident Report
Testjuhtumite kirjeldus

1. Test-incident-report identifier

2. Summary

Märgitakse ära, et millise probleemiga oli tegemist.

3. Incident description

Juhtumi kirjeldus
INPUTS -- sisend
EXPECTED RESULTS -- oodatud tulemus
ACTUAL RESULTS -- tegelik tulemus
ANOMALIES -- kõrvalekalded
DATE and TIME
PROCEDURE STEP
*ENVIROM*ENT -- millises kohas ümbruses tekkis
*ATTEMPTS TO REPEA*T -- millised katsed võeti ette selle vea kordamiseks
TESTERS -- kes olid testijad
OBSERVERS -- kes olid tunnistajad

4. Impact

Hinnatav vea mõju ja raskus

Test-Summary Report

1. Test-summary report identifier

2. Summary

Hindaja number, nimi ja rühm

3. Variances

Kõrvalekaldumised, mida leiti

4. Comprehensiveness assessment

Põhjalikuse hinnang. Kui head olid testi tulemused. Kas leidub veel vigu.

5. Summary of results

Leitud vigade ülevaade. Millised testid tehti.

6. Evaluation

Hinnang projektile

7. Summary of activities

Tegevuste kokkuvõte. Mida sai tehtud. Millisid tööd tehtud ja kui palju aega kulutatud.

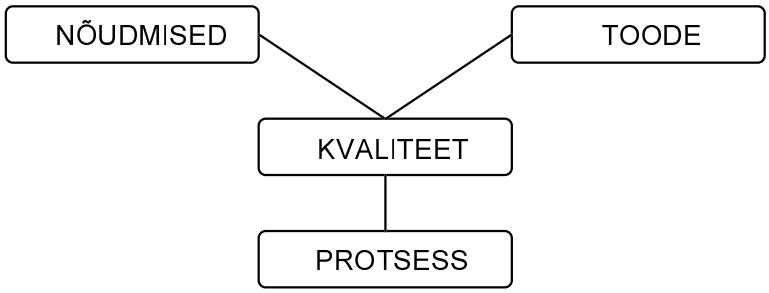
8. Approvals

Ülalkirjeldatud dokumentidel on erinev kaal. Kõige tähtsamaks osutub tavaliselt viimane dokument. See määrab ära tavaliselt tehtud tööde kvaliteedi ja millest omakorda sõltuvad tööde eest saadavad rahad.

Sissejuhatus ja kursuse põhimõisted

Sissejuhatus.

On enesest mõistetav, et tootja hoolitseb oma toote kvaliteedi eest, kuna siis on seda kergem müüa. Tarkvara ja ka mõningate muude tooteliikde puhul tekib aga mõneti ootamatu küsimus: kui kvaliteetne peab toode olema? Ilmneb, et liiga kõrge kvaliteet läheb ka liiga kalliks ja pole seega otstarbekas. Seetõttu tuleb kvaliteeti vaadelda sõltuvalt toote kasutuseesmärgist.



Ideed ja põhimõisted.

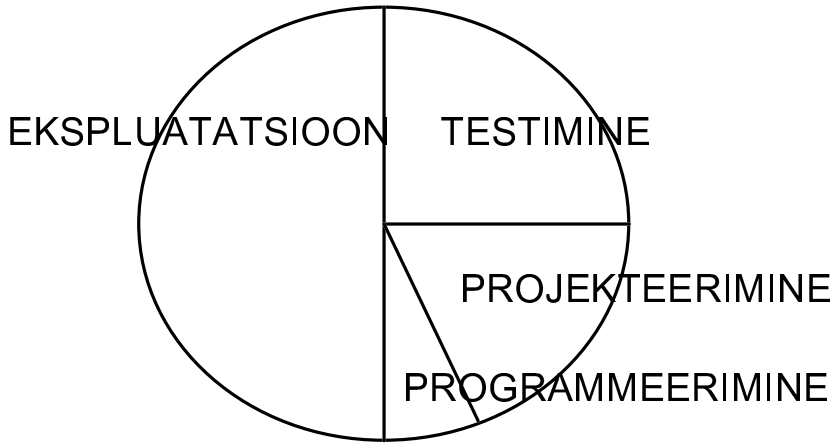
KVALITEET
- vastavus nõudmistele.
- et saavutada kvaliteetset toodet peab ka tootmisprotsess olema kvaliteetne
- suhe nõudmiste, toote ja protsessi vahel
- totaalne mõiste, kuna kvaliteet käib iga asja kohta

STANDARDID -- on kolm normolekut:
- konsensus
- tunnustatus kehami poolt
- eesmärkide saavutamine (kulutuste minimaliseerimine, komponentide vahetatavus jne)

TARKVARA
- programmid
- protseduurid
- dokumentatsioon
- andmed

PROTSESS
- tarkvara arendus, näiteks kosemudel

Enamasti võib tarkvara arendusprotsessi maksumuse jagada umbes nii nagu joonisel kujutatud:



I osa. Tarkvara kontroll: meetodid ja korraldus

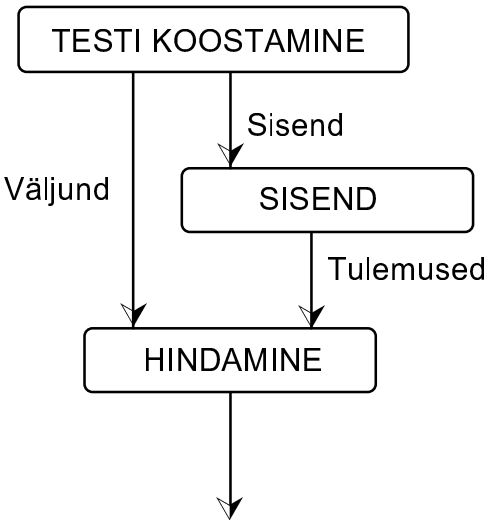
1.Testimine

1.1. Testimise põhimõtted

Testimine on programmi täitmine eesmärgiga leida vigu. Test on komplekt sisendandmeid ja vastav komplekt oodatuid väljundandmeid. Sisendandmed antakse programmile ette ja saadakse väljundandmed. Oodatud väljundandmete ja tegelike väljundandmete võrdlusest saadaksegi testi tulemused. Programm on vigane kui programmi tegelik väljund ei vastanud oodatud väljundile. Kui leiti viga (vigu), siis on see olnud edukas test. Ühegi vea mitteavastamise puhul pole test õnnestunud. Hea on test, mis avastab võimalikult palju vigu. Testimine ei suuda tõestada, et programmis vigu pole. Ta võib vaid näidata, et programmis on vigu.

Esimesed testid viivad läbi arendajad. Kuid sellest tavaliselt ei piisa. Parimaid tulemusi võib oodata süsteemi tulevastelt kasutajatelt, kes ei ole arendusega seotud.

Testi toimumise graaf võiks olla järgmine:



programmist.

Testimise projekteerimisel tekib terve rida küsimusi:

- kuidas hinnata väljundit?
- millal testimine lõpetada
- kuidas valida sisendeid?
- kes on testijaiks
- millal, kas ja kui, siis kuidas läbi viia kordustestimist

On välja mõeldud terve rida tarkvara testimiseks ja arendamiseks ette nähtud meetodeid. Näiteks musta ja valge kasti meetodid, kus esimesel juhul ei süüvita programmi teksti, vaid testitakse tema funktsionaalsust; teisel juhul vastupidi uuritakse tähelepanelikult koodi ja püütakse teha teste programmitekstist lähtuvalt.

Silmas tuleb pidada ka seda, et eeldatavad väljundid saadakse ülesande püstitusest, mitte

Mõisteid:
Silumine -- leitud vea kõrvaldamine.

Verifitseerimine -- püüab näidata, et järgmise etapi produkt vastab eelneva etapi määratlusele.
Valideerimine -- püüab näidata, et oleme üleüldse teinud seda mida vaja.
Sertifitseerimine -- kolmanda osapoole tegevus, mis püüab näidata, et antud toode vastab vajalikele standarditele ja normdokumentidele.

1.2. Programmipõhine testimine.
1.2.1 Lauseadekvaatsuse kriteerium.

Testimise tulemusena peab programmis iga lause vähemalt üks kord töötama. Programmi testimine teksti põhjal ei ole tavaliselt piisav.
NÄIDE:
Programm: Tähtaeg (teha funktsioon, mis väljastab järgmise kuu esimese tööpäeva praeguse kuupäeva põhjal). Function DueDate (anyDate) -- arvuta järgmise kuu esimene tööpäev.

˝ PROGRAMM:

```
Dim Result;
if  Not IsNull(AnyDate) then
        Result=DateSerial (Year(AnyDate),chonth(AnyDate)+1,1)
        if  Weekday(Result)=1  then DueDate=Result+1  "Pühapäev"
        elseif WeekDay(Result)=7 then DueDate=Result+3 "Laupäev"
        else  DueDate=result
        endif
else Result=null
end if
end if
end function
```

Programm: Keskmine vanus. (Leida andmete koguarv, isikute arv, kes on keskharidusega ja vanus 25-50.

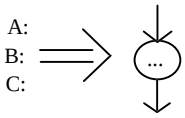
PROGRAMM:

```
yldno:=no:=summa:=0;
read nimi,haridus,vanus;
do  while  nimed not" "
        ylddno:=yldno+1;
if  (haridus="kesk" and vanus>=20 or  vanus<=50)
        then no:=no+1; summa:=summa+vanus; read nimi,haridus,vanus;
end while;
print yldno,no,summa/no]
```

Lauseadekvaaatsuse kriteeriumi täidetavus -- alati pole võimalik neid kriteeriume täita, näiteks pole lause täitmist võimalik vaadata, sest programmi loogika järgi pole võimalik tingimust täita.
Alati pole võimalik automaatset algoritmi teha, et otsustada, kaas lauseadekvaatsuse kriteerium on täidetav või mitte.
KOKKUVÕTE:
IDEE: Kõik programmi osad on töötanud.
EELTINGIMUSED: Programmi teksti analüüs
EELISED: Programm on süstemaatiliselt katsetatud. Vähe teste. Selge idee.
PUUDUSED: Ei anna andmete teste. Ei leia puuduvaid harusid. Teksti pole alati olemas.
TULEMUSED: Testikomplekt mis katab programmi
SUHE TEISTESSE: Iseseisvalt mitte kasutada.
HINNANG: Kui aega vähe, siis kasuta.
VAHENDID: On olemas mis aitavad kaasa.

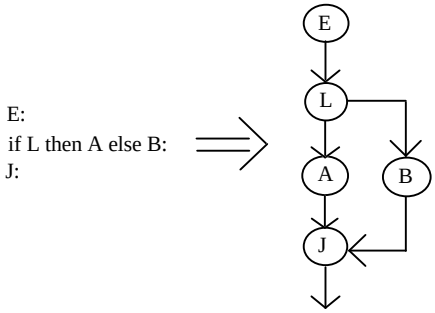
1.2.2. Haruadekvaatsuse kriteerium

Kõik harud peaksid programmis olema läbitud. Programmi kaared graafil peaksid olema läbitud.
Näiteid programmi graafi kohta ja erinevate kriteeriumite võrdlus:
Näide 1.



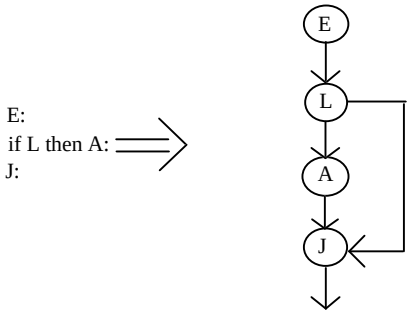
Lauseadekvaatsus = Haruadekvaatsus

Näide 2.



Lauseadekvaatsus = Haruadekvaatsus

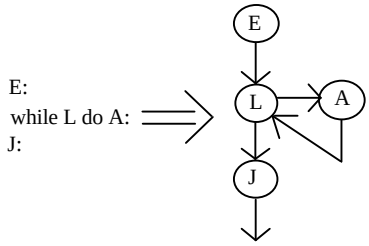
Näide 3.



Lauseadekvaatsus ≤ Haruadekvaatsus

Lauseadekvaatsuse puhul läbitakse kõik laused, kuid harud jäetakse läbimata. Haruadekvaatsus läbib ka kõik harud. Seega on ta täielikum.

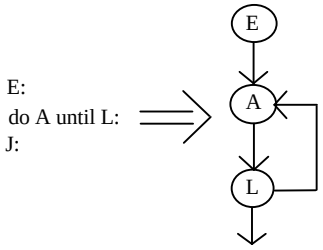
Näide 4.



Normaalselt Lauseadekvaatsus = Haruadekvaatsus

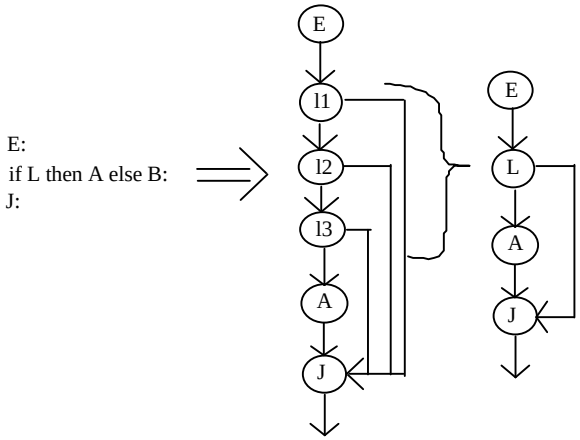
Lõpmatu tsükli puhul Lauseadekvaatsus < Haruadekvaatsus

Näide 5:



Lauseadekvaadsus ≤Haruadekvaatsus

˝ Näide 6:



Lauseadekvaatsus ≤ Haruadekvaatsus

Sõltuvalt translaatorist võivad elementaartingimused täitmata jääda. Näiteks "C"-s. Vahel võidakse ka elementaartingimustes (l1, l2, l3) midagi teha.
MÄRKUSI:

-- Testimiseks pole graafi vaja
-- Tulemused ülesandest
-- Arvestada elementaartingimusi

Lause- ja haruadekvaatsuse testid viiakse läbi programmi teksti põhjal, mis vahel võib päris kasulikuks osutuda. Paljudel juhtudel lause- ja haruadekvaatsus langevad kokku. Kuid keskmiselt on lause-adekvaatsus nõrgem haruadekvaatsusest. Mõningatel juhtudel piisab lauseadekvaatsusest, sest karmimad testid on liialt töömahukad.

TESTIMINE PROGRAMMI TEKSTI PÕHJAL

Lõpetamise otsustamine ⇐ programi tekst
Testide sisendid ⇐ --------"----------
Testide väljundid ⇐ ülesanne
⇓
Testimine

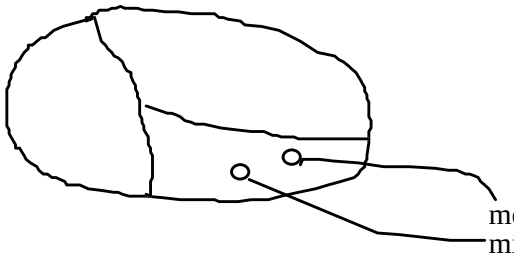
Testimine ei võimalda selgitada, et vigu pole, testimine selgitab, et vigu on, sellepärast võib testimine kesta lõpmatuseni. Testimise võib lõpetada siis kui mingi adekvaatsuskriteerium on rahuldatud.

1.3. Funktsionaalne testimine

Lõpetamise otsustamine ⇐ ülesanne
Testide sisendid ⇐ -----"----
Testide väljundid ⇐ ülesande püstitusest
⇓
Testimine

EKVALENTSIKLASSIDE ANALÜÜS

Eelduseks on et igas piirkonnas on kõik punktid testimise koha pealt võrdsed:



Oskame nii sisendid kui väljundid jaotada ekvivalentsi klassideks, nii et klassi sees olevad komponendid oleksid sarnased. Kui üks leiab vea, siis ka teised ja vastupidi.

Funktsionaalse testimise võib eteppeiks jagada järgmiselt:

1. eristada ekvivalentsiklassid
2. leida igas klassis piiridel olevad olukorrad-andmed
3. leida seosed olukordade andmete vahel
4. koostada testolukorrad-testid
5. vormistada testimise plaan

KOKKUUVÕTE:

EELDUSED:	spetsifikatsiooni olemasolu
EELISED:	kasutajale lähedasem
PUUDUSED:	kui ekvivalentsiklassid on sõltuvuses, läheb palju aega
HINNANG:	hea

2. Staatilised meetodid.

Erineb eelmises peatükis kirjeldatud meetoditest selle poolest, et programmi ei käivitata vaid uuritakse dokumentatsiooni. Meetodi positiivseks küljeks on see, et testida saab juba enne programmi valmimist projekteerimise ja spetsifitseerimise ajal. Just neil etappidel tehtud vead osutuvad hiljem väga valusalt kätte maksvaiks. On vajalik, kuna kõiki olukordi ei õnnestu modelleerida; Näiteks katstroofid või mingid muud piirsituautsioonid.

2.1. Kontrollimine arvutita.

On levinud 3 erinevat suunda: a) Töö analüüs autori poolt
b) Läbivaatus \ audiitorkontroll
c) Programmeerija hindamine

Vaatame neid lähemalt.

a) Seda teeb iga programmeerija, kuid see võib olla ebapiisav. Siin võib olla mitmeid põhjuseid: näiteks programmeerija ei ole küllalt enesekriitiline. Autor tunneb oma programmi küll kõige paremini, aga ta jälgib oma loogikat ja ei taju neid (varjatud) võimalusi, mis programmis olemas on. On kõige odavam meetod.

b) Selle jaoks on olemas spetsiaalne standard. *ANSI/IEEstd 1028-1988 IEE Standard for Softvare Reviews and Audits.*

AUDIT on sõltumatu osapool, kellel on mingi tunnistus või tunnustus. Tegevus on rohkem väline:

- hinnatakse tarkvara produkti ja protsessi
- hinnatakse vastavust juhendeile, standardeile..
- põhineb dokumenteil, mis annavad objekti kohta infot

Siia kuulub (struktuurne) arutelu (ing. k. *walkthrough*). Struktuurne arutelu annab parema koostöö grupis, ükski teine meetod ei anna hinnanguliselt niivõrd suurt vigade vähenemise hulka. Struktueerimata programme ei paranda ükski arutelu.
IDEE: Mingi grupp vaatab produkti üle ja arutab selle läbi.
MIKS SEE ON HEA? Arutelu on kõige parem viis vigade avastamiseks. Paranevad kontaktid grupis, vigade arv väheneb (väidetakse, et kuni 10 ja enam korda), suureneb produktiivsus ja kvaliteet, ühe osavõtja lahkumisel saab teda lihtsalt asendada ilma, et mida väga halba juhtuks.
TAKISTUSED:

- grupi liikmed võivad olla erinevatest osakondadest, siis ei saa asja korralikult käima.
- grupi liikmed võivad olla erinevad: kõrge IQ-ga, kannatamatud, konservatiivsed, vähe huvitatud "reaalsest maailmast", eelistavad eraldatust jne.
- kellegile ei meeldi, kui teda kritiseeritakse

FORMAALSUS: Nii palju ettekandeid (ettekanne, listingud) kui vaja , nii vähe kui võimalik. Rusikareegel, et ettekande pikkus 30..60 min.
EELTINGIMUSED: Kõigil grupi liikmetel peab olema teatav ettekujutus sellest, mida neilt oodatakse. Peab olema koostööõhkkond. Materjalid peaksid olema ette valmistatud ja kätte jagatud ja osavõtjad peaksid nendega tutvunud olema. Rusikareegel siin: Igaühel on üks positiivne ja üks negatiivne kommentaar.

Osavõtjad ja nende rollid:
Grupi liikmed võiksid olla järgmised:

- Esitaja (see ei pruugi olla produkti autor)
- Kordinaator (juhataja), keegi kes jälgib, et asi oleks asjalik
- Sekretär (paneb kirja ettepanekud, ideed ja muu tähtsa teksti, et see kaduma ei läheks)
- Liikmed
- Vaatlejate ja konsultantidena juurutamise spetsialist, standardite spetsialist, kasutaja esindaja (eriti alguses ja lõpus)

Inimesi ei pruugi nii palju ollagi, rollid võivad olla ühendatud.
Grupi koosseisu ei ole soovitav võtta otsest ülemust. See võib panna rahva oma töötasude peale mõtlema.

Mõnikord võib arutelu käigus tekkida mäng. Näiteks sõnaühend "Ja, aga ... " viitab mängule. Tuleks aru saada, et käib mäng ja see õigel ajal ära lõpetada. Mängud rikuvad normaalset arutelu. Mäng s.o. tegevuste jada, millel on oma varjatud siht, kuid mis väljaspoolt paistab teistsugune.

ARUTELU TULEMUS:

a) leitud ja parandatud vead ning parem süsteem.
b) allakirjutatud protokoll.

JUHTKOND JA ARUTELU:

- Soodustada juhtkonna osavõttu arutelust, kui pole just triviaalseid ülesandeid.
- Jälgida, et osavõtjad on ette valmistunud, soovivad saada tulemust.
- Usaldada gruppi, lubada "tühja juttu" mõnikord see laadib pingeid maha
- Jälgida ajalimiiti.
- Osavõtjad kirjutavad kokkuvõtetele alla.
- Jälgida formaalseid nõudeid.
- Veenduda, et teatakse standardeid.
- Vältida arutelul viibimist, kui pole otsene grupi liige.

c) Programmerija hindamine.
IDEE: Anda programmeerijaile arusaamine oma tugevaist ja nõrkeist külgedest. On anonüümne, ei mõjuta grupi liikmete palka ja käekäiku.

MEETODI SIHT: Anda hinnang programmeerija kvalifikatsioonile.
KIRJELDUS: On vaja gruppi, kuhu kuuluks 6-12 inimest. Igaüks neist valib enda tehtud töödest oma hinnangul parima ja halvima programmi (ei ütle kumb on halvim ja kumb parim) ning hindab teiste poolt kirjutatud kahte programmi. Peab olema tagatud hinnatavate anonüümsus.
Tuleks vastata küsimustele: Arusaadavus? (ka projekt) . Projektile vastav? Muudetav? Oleks ise rahul ?
TINGIMUSED: Grupi programmide olemasolu. Vajadus selle protsessi järele. (kuna ise see protsess on aeganõudev). Vaja initsiaatoreid.
TULEMUSED: Iga programmeerija saab hinnangu oma töödele.

2.2. Küsimustikud.

Küsimustikud on küsimuste komplektid, mis on tehtud eri keelte eripära arvestades aga on ka üldisi küsimustikke. Neid võib kasutada autor, läbivaataja, analüüsija... Sisaldavad paljusi momente, millele muidu ei oskagi nagu tähelepanu osutada. Samas võtab nende küsimustike järgimine tohutult aega (NASA küsimustikes on näiteks üle 1 000 küsimuse) Siin käsitletakse küsimusi, mida programmide kohta küsida. Mõned küsimused on üldisemad, mõned kitsamad, mõnede keelte puhul langeb mõni küsimus sootuks ära.
PROGRAMMI ANALÜÜSIMISEKS ESITATAVAD KÜSIMUSED:

Andmete kirjeldamine

- Kas muutujad on ilmutatult kirjeldatud? (sõltub keelest)
- Kas vaikimisi atribuudid OK?
- Kas inidekseeritud õieti? (massiivid)
- Kas muutujad on sarnaste või segadusseajavate nimedega. Näiteks VOLT ja VOLTS, I1 ja O0

Pöördumine andmete poole

- Kas pöördumisel väärtus olemas?
- Kas indeks väljaspool piire?
- Kas indeksile omistatakse täisarv?
- Kas viidamuutujale vastav mäluväli on määratud?
- Kas ühismälu on õieti?
- Adresseerimine sõna piiril
- Kas ühised andmestruktuurid on kirjeldatud alamprogrammides ühte moodi?

Arvutuste vead

- Kas tehakse arvutusi lubamatute tüüpidega?
- Kas kasutatakse eri tüüpi andmeid koos?
- Kas juhtub üle/ala(tulemus kaob väiksuse tõttu ära)täitumist?
- Kas toimub jagamist nulliga?
- Kas muutujad on väljaspool sisulisi piire ?N: Tõenäosus>1
- Kas ebatäpsuste kuhjumisel võib tekkida oluline viga?
- Kas täisarvuline aritmeetika on õige? (kas sulud õieti)
- Kas tehete prioriteedid on õiged?

Võrdluste vead

- Kas võrreldakse sobimatuid? N: teksti ja numbrit
- Kas spetsifikatsiooni terminid on programmis õieti ? N: 'suurim' , 'mitte väiksem kui'
- Kas võrdlustehete prioriteedid on õieti antud?
- Kas tulemus sõltub kompilaatorist? IF((x<>0)AND(y/x>z))
- Kas võrreldakse reaalarvu mingi võrdusega? reaalarv=...

Juhtimise vead

- Kas igal BEGIN'il on END
- Kas programm või moodul lõpetab töö? (Kuidas seda kindlaks teha?)
- Kas tsükkel või kordus lõpetab töö?
- Kui tsükli tingimus on kohe vale, mis siis?
- Kui FOR alumine raja>ülemine, kas siis on OK?
- Kas on kõrvalekaldumisi? (suured/väikesed korduste arvud)

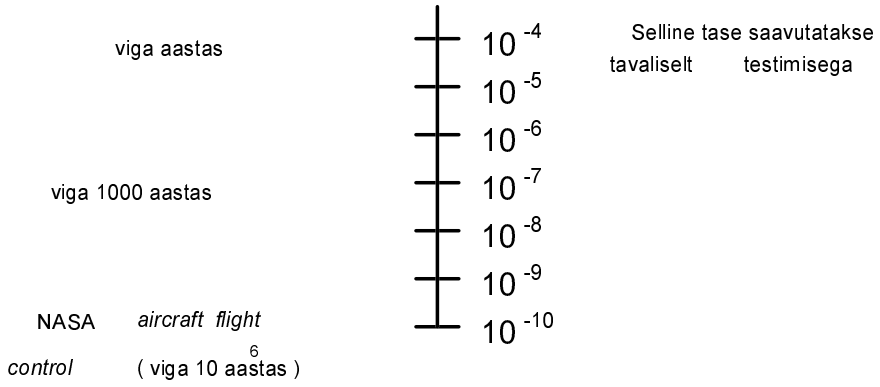
Alamprogramm
- Kas väljakutsel muutujate arv/atribuudid sobivad?
- Kas mõõtühikud on samad (kraadi C° / RAD)
- Kas parameetrid sisuliselt samas järjekorras?
- Mitu sisendpunkti on - kas kõigile neile antakse väärtusi?
- Kas alamprogramm muudab sisendparameetreid?
- Kas kantakse üle konstante?
- Kas globaalsete muutujate atribuudid on igas moodulis samad?

Sisend ja väljund
- Kas faili atribuudid on õiged?
- Kas avamine/sulgemine on õige?
- Kas kõik vajalikud failid avatud/suletud?
- Kas pöördumine ühtib kirjeldustega?
- Kas sisend/väljund puhvrite pikkused on õiged?
- Kas faili lõputingimus OK?
- Kas veaolukorrad/katkestused OK?
- Kas tekstides sisulisi/grammatilisi vigu?

Varia küsimused
- Kas translaatori hoiatusi ja teateid on uuritud?
- Kas vigaseid sisendeid proovitud?
- Kas kõik funktsioonid olemas?
- Kas veel testida? Kui leiti viga, võib veel leida, kui ei leitud, siis halvasti testitud.

2.3. Tõestamine.

Miks on vaja programmi tõestamist? Uuringud on näidanud, et peale testimist annab programm keskmiselt 10^{-4} viga tunnis st. 1 viga aastas. On olemas vahendeid, mis suurendavad veakindlust suurusjärgu võrra st. 1 viga 10.aasta jooksul. Aga näiteks lennujuhtimises on nõutav 10^{-10} viga tunnis s.t. miljoni aasta kohta üks viga.



• tuum/ümbrus - see kindlustab, et "halvad asjad ei tule" , aga ei kindlusta, et "head asjad toimuvad"
• veapuu analüüs - st. et alustatakse mingist suurest veast, mida tahetakse vältida ja siis hakatakse vaatama selle vea eltingimusi ja selle eeltingimuse eeltingimusi. Saab puu.
• dubleerimine - kui pannakse paralleelselt käima mitu programmi ja võrreldakse tulemusi, enam levinud vastused loetakse õigeiks.
Vea tõenäosust saab vähendada ka programmi tõestamise abil?

SIHT: Näidata, et programm vastab spetsifikatsioonile
MEETOD: Arvutus (tuletusreeglid, aksioomid, WFF)
WFF- süntaks, mis määrab keele
VERIFITSEERIMINE: Peab olemas olema väited programmi sisendi (pre-) ja väljundi (post) kohta + vahepealsed
EESMÄRK: Näidata, et pre->post ja programm lõpetab töö.
EELISED: See on range protseduur, ainuke viis silmuste (WHILE ja üldse tsüklite) jaoks. Ainuke viis 10^{-10} saavutamiseks (?)
PUUDUSED: Töömahukas
See ei välista spetsifikatsiooni vigu. Sel juhul tõestatakse vale spets.
Ei välista tõestuse vigu.
Suurte süsteemide jaoks ei sobi.
Iteratsiooni invariandid - raske
TINGIMUSED: Spetsifikatsioon, pre- ja posttingimused.
TULEMUS: Programm tõestatakse spetsifikatsiooni suhtes.
SUHE: Saab kasutada koos analüüsiga.
HINNANG: ? Vähekriitiliste puhul pole otstarbekas, kriitiliste puhul pole selge, kas ta on saavutatav.

Abstraktsioonitasemed/probleemid tõestamisel:

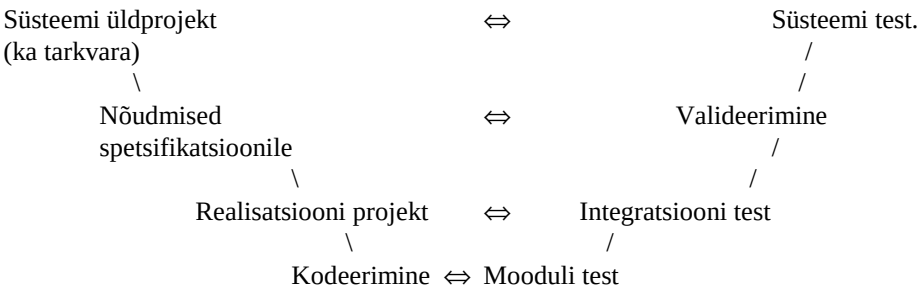
TASE	VAHENDID	MÄRKUSI	RASKUSASTE
Ülemise taseme spetsifikatsioon	Läbivaatus	Ei ole formaalne testimine, tõestamist ei saa teha	Peamisi probleeme
Realisatsiooni projekt	Projekti tõestus	Sisulist kontrolli ei saa veel teha. Saab kontrollida , et poleks vasturääkivusi.	Sõltub formalismist
Programmi kood	Programmi tõestamine	Väga mahukas	Raske
Tõestusmeetodid OK	Loogika		Keskmine
Objektkood	Programmi või kompilaatori tõestamine		Raske
Täitmine	Mikroprogrammi/riistvara tõestamine	Vajadus töökorralduse järele. Kallis. Tarvis ka toetavat tarkvara.	

Riistavara puhul 10^{-10}?.

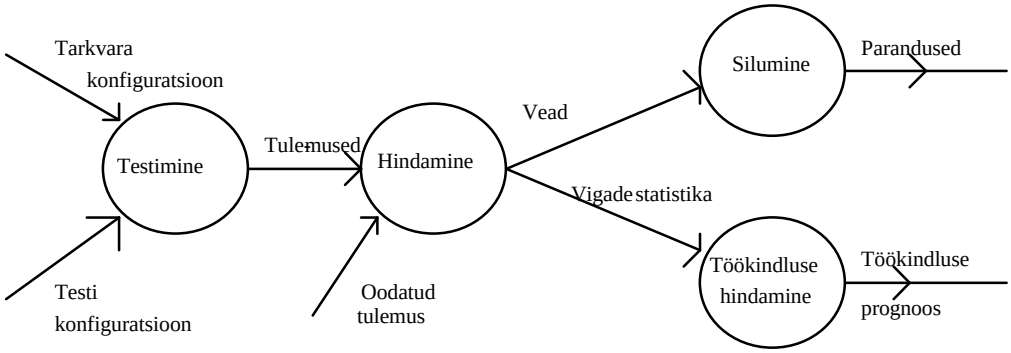
- Kõrge töökindlusega seadmed.
- Dubleerimine / hääletamine
- Lihtsad funktsioonid

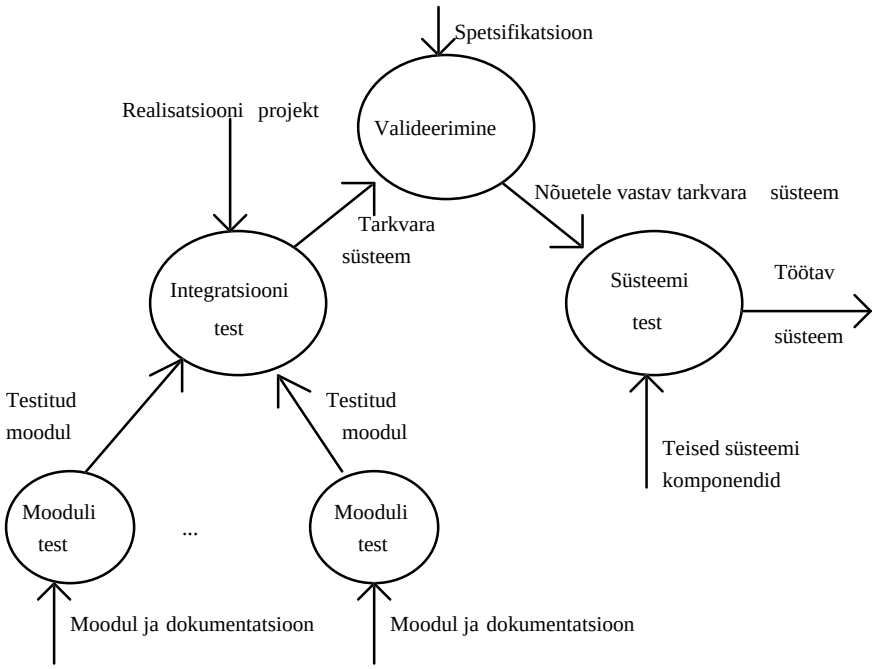
3. Kontrolli korraldus.

TESTIMISE STRATEEGIA.:



TESTIMISE KORRALDUS:

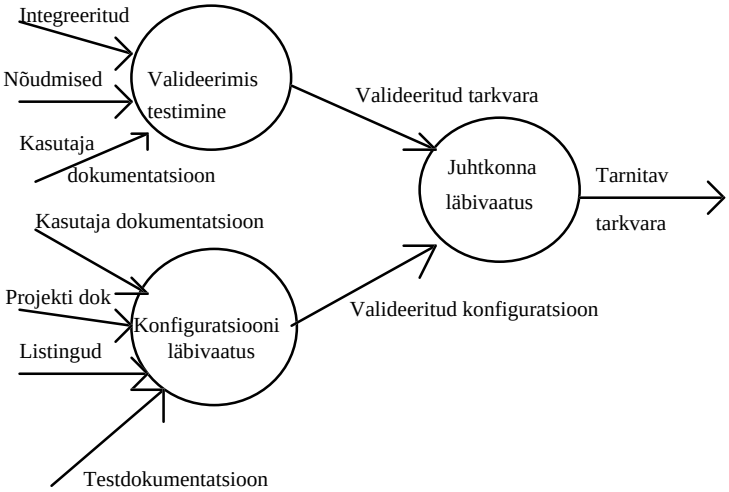




Vaatame testimisi lähemalt:
* Ühiku/mooduli test.
Läbivaatus
Piirjuhud
Vea oletus Need on eelnevalt vaadatud
Ekvivalentsiklassid
Programmi põhjal

* Integratsiooni test.
'Kõik kokku ja test' - pannakse programm moodulitest kokku ja vaadatakse, kas hakkab tööle või mitte. Tavaliselt ei hakka. Seda eriti ei soovitata.
'Alt üles' - Osa mooduleid asendatakse draiveritega, mis teevad osa tööd ära.
IDEE: • moodusta klastrid
• kirjuta draiverid
• testi
• eemalda draiverid (klaster s.o. moodulite kogum)
EELIS: Korraga saavad töötada paljud inimesed - igaüks saab testida oma juppi.
MIINUS: Süsteemi üldine töötamine võib jääda selgusetuks.
'Ülalt alla' - Põhimoodul-lühised-test-asenda-regressioonitest (lühis so. asendav programm).
EELISED: Saab järk-järgult lisada uusi mooduleid. Midagi on juba algusest saadik olemas, parem side on kasutajaga. Kohe alguses saab mingi üldpildi süsteemist. Eriti kasulik arendusjuhtidele, kes vähemalt aimavad kuhu nad on teel.
'Võileib' - kombineeritud 'ülalt alla' testimine. Võetakse mingi vahenivoo ja testitakse selle tasemini (tase tuleks valida selline, kus hakkab selguma juba sisulisi tegevusi). See on kõige soovitatum.

* Valideerimine.
KONFIGURATSIOONI LÄBIVAATUS:
ALFA testimine: kasutaja poolt/andmetega arendaja keskkonnas
BEETA testimine: kasutaja poolt/andmetega kasutaja keskkonnas



Süsteemi testimine:
Recovery (taasalustamine, taastamine)
Simuleeritakse võimalikke veaolukordi: voolukatkestust, mõne seadme riket, jne.Uuritakse kui raske on süsteemi uuesti tööle panna.

Security (Kaitse)

Kas kaitsesüsteem on tugev? Kas kaitsesüsteem on korrektne? Võimalik, et palgatakse näiteks üks häkker-tüüpi mees ja lastakse ta süsteemi kallale.

Stress

Katsetakse olukordi, mis spetsifikatsioonis pole ette nähtud (ülepingutatud olukorrad). Süsteem peab ära hoidma suuremad õnnetused, küll aga on lubatud mõningate osade või kasutajate blokeerimine.

Performance (jõudlus)

Loomuliku maksimumolukorra testimine

Süsteemi ei saa testida, kui pole esitatud nõudmisi. On olemas ka niivõrd erilisi olukordi, mida on kulukas või võimatu testida: maavärin vms.

Võimalik on teha:

- töökindluse mudelid
- simuleerimine
- vigade arvu prognoos
- tõestamine, läbivaatus

TESTIMISE FAASID:

1. Mooduli test
2. Integratsiooni test
3. Valideerimine
4. Süsteemi test

II TARKVARA KVALITEET JA STANDARDID.

Tarkvara kvaliteedi parandamisest.

• Kvaliteeti ei saa "sisse testida". Mingi testimine ega kontroll ei muuda toodet paremaks.
• Parendamise vahendid:

a) Tarkvara arenduse meetodid.
-üldised: struktuurne spetsifitseerimine, disain
-spetsiaalsed: veakindel programmeerimine, n-versioon , taastatavad punktid...
b) Riistvara vahendid (otseselt ei tõsta, rohkem soodustavad)
-võimsad keskkonnad
-tagavara koopiad
-UPS
c) Tarkvara vahendid
CASE, head keeled, dokumenteerimisvahendid...
d) Läbivaatused,verifitseerimine, testimine, valideerimine ....
e) Organisatoorsed (aruandlus)
f) Tarkvaraspetsiifiline kvaliteedi juhtimine
g) Standardid

4. Kvaliteet ja standardid (ülevaade)

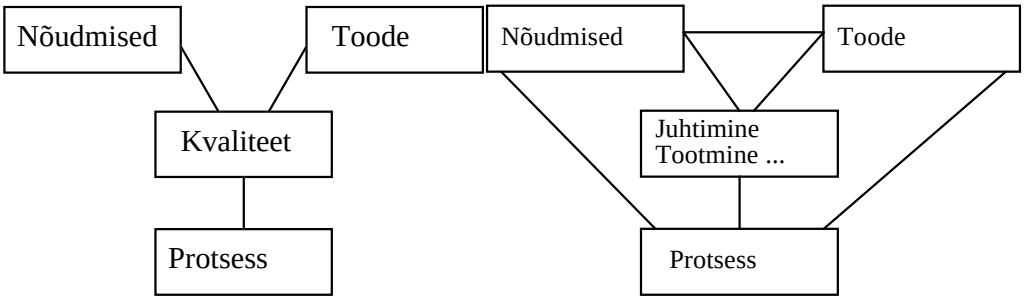
4.1. Kvaliteedi juhtimine

DEFINITSIOON: ISO 9000 järgi on kvaliteet toote või teenuse omaduste kogum, mis võimaldavad sõnastatud või eeldatud vajaduste täitmist.

Seega võib öelda, et kvaliteet on vastavus nõudmistele. Kuna nõudmisi on erinevaid võib ka tootel olla mitu kvaliteeti või kvalteeti mittes mõttes.

Nõudmiste allikaiks võiks olla järgmised:

NORMID	Üldised, sõltuvad erialst
STANDARDID	Sõltuvad eriaalast
SPETSIFIKATSIOON	Sõltub ülesandest



KVALITEEDIJUHTIMISE PÕHIMÕTTED:

Kõigis ettevõtteis on olemas kvaliteedijuhtimine. Mis asi on aga kvaliteedijuhtimine? Sisuliselt juhtimine ongi kvaliteedijuhtimine.

Kvaliteet s.o. moodus organisatsiooni juhtimiseks (Armand V. Feigenbaum)

Kvaliteedi mõiste on olnud erinev kahes koolkonnas või vaatenurgas: USA ja Euroopa ning Jaapani arusaamad kvaliteedist on olnud mõnevõrra teistsugused.

USA, Euroopa - kvaliteet on tulemusorienteeritud (*result oriented*), oluline pole mitte tootmisprotsessi kvaliteetsus vaid lõpptulemuse oma. Paraku, eriti tarkvara puhul, see ei taha hästi töötada.

Jaapan - Kvaliteet on protsessorienteeritud (*Process oriented*), oluliseks peetakse, et kogu tootmine oleks kvaliteetne, ehk kõik mis tehtud olgu tehtud hästi.

Veel on erinevad olnud ka arengumeetodid: kui USA -s ja Euroopas tehakse läbimurdvaid, kardinaalselt uusi otsustusi ja samme (innovatsioon), siis Jaapan on osanud areneda pidevalt, ilma suurte hüpete või suunamuutusteta.

Eelpooltoodust tulenevalt on USA ja Euroopa kvalitedilähenemine orienteeritud partiide kontrollile, s.t. statistilistele meetoditele. Jaapanis on kvaliteet seevastu rohkem iga töötaja või iga hetke küsimus.

Philip B. Crosby on jaapanlaste mõtteviisi paigutanud kolme teesi:

- Tee seda õieti kohe esimesel korral
- Ei ühtegi vigast toodet
- Kvaliteet on tasuta

Philip B. Crosby kvaliteedijuhtimise koostisosad:

1. Juhtkonna toetus
2. Kvaliteedigrupp
3. Kvaliteedi mõõtmine
4. Kvaliteedi maksumuse hinnang
5. Töötajate kvaliteediteadlikkus, selle õpetamine
6. Nulldefekti komitee
7. Nulldefekti päev
8. Selgelt formuleeritud sihid näiteks 30, 60, 90 päeva peale
9. Vea põhjuste kõrvaldamine
10. Parimate tunnustamine
11. Kõike seda tuleb teha korduvalt

See Crosby süsteemn kipub euroameerikaliku ja jaapanliku suhtumise vahepeale.

Armand V. Feigbaum on sõnastanud järgmised kvaliteeedijuhtimise tegevused:

1. Püstita kvaliteedistandardid
2. Hinda vastavust standardeile
3. Tegutse kui standardeid rikutakse
4. Arenda standardeid edasi

Freigebaumi süsteem on rohkem ameerikalik lähenemine.

Kvaliteedi mõõt koosneb:

1. Juhtimine 10%
   Firma poliitika ja strateegia 8%
   Inimeste juhtimine 18%/2
   Kuidas hangitakse, kasutatakse ja käivitatakse vahendeid 9%
2. Potsessid 14%
3. Kasutaja rahulolu (hinnang) 20%
   Firma töötajate rahulolu 18%/2
   Ühiskondlik mõju 6%
4. Äri tulemused 15%

4.2. Standardid.

So. konsensuse alusel koostatud ja tunnustatud kehami poolt kinnitatud normdokumentatsioon, mis on suunatud standardimiseesmärkide saavutamiseks.

Keham - so. organisatsioon, mis midagi teeb

Konsensus - standard loob üldjuhul eelise teatud organisatsioonile. Kehtestamiseks on vaja, et ka teised seda aksepteeriksid. Standardi kehtestamine on seega kokkulepe ja kompromiss. Standard soovitatakse muuta kõigile kättesaadavaks ja mingil määral kohustuslikuks

Tunnustatud keham - mingi organisatsioon, kes selle standardi kehtestamise läbi viib.

EELISED/EESMÄRGID:

Kulutuste minimiseerimine (allhanked).
Komponentide vahetatavus.
Komponentide koostöökulu minimiseerimine (jääb ära igasuguste liideste tegemine).
Protseduuride lihtsustamine / parendamine (saab teistelt ettevõtteilt kindlalt töötavaid protseduure üle kanda).
Kvaliteedi juhtimine.

Standardid võimaldavad tekitada ja parandada tootjate omavahelist konkurentsi.

PUUDUSED:

Lisakulu,-aeg,-bürokraatia
Takistab innovatsiooni? (sissetallatud radadelt on raske kõrvale kalduda)

KOHUSTUSLIK?

Ei, kuid seda võib muuta seaduste abil (omanik võib ka selle muuta kohustulikuks)

MAHT? MILLAL?

Nii palju / vara kui vajalik
Nii vähe / hilja kui võimalik.

Standardite liigitus:

- Ametlikud standardid (vastu võetud tunnustatud kehami poolt).
- Tööstus e. de facto standardid (ei pruugi olla ametlikult kinnitatud, kuid on üldlevinud).
- Tehnilised raamstandardid (standardite kogumikud, juhendid nende kasutamiseks)
- Riiklikud soovitused (tulenevad kliendi poolt)
- Firmasisesed standardid.

Standardimiskehameid:

- ISO *International Organisation for St.-ion* (1947)
- IEC *the Inernational Electrotechnical Comission* (1906)
- ISO / IEC JTC1
- Üle 170 tehnilise komitee (TC)
- *Sub-Committees* (SC) ⇒ *Working Group* (WG)
- ISO / TC 176
- ESMA *European Computer Manufactures Association*
- X/Open

Näide: ISO / IEC JTC1/SC7/WG3 Nº 686 *Qualiti Sub-Characteristics*.

- ANSI

FIRMASISESED: Lihtsad aru saada ja jälgida

Piisavad, lünkadeta
Ühesed
Ellu viidud.

Võib öelda, et kui standard pole ette valmistatud elluviimiseks, siis pole mõtet tema peale aega raisata.

STANDARDID: EESTI '92 LÕPP

• Esmatähtsad standardiseerimise objektid:

1) Märgistikud ja nende kodeerimine: ISO 8859-1, *de facto* "*IBM Code Page 850*", "*WINDOWS Code Pages*" põhjal.
2) Sõrmistikud: ISO 3243 põhjal
3) Esitusvormid (kellaaeg, kuupäev, raha, tähestiku sorteerimine): "*IBM National Language Support*" põhjal

EDASI: Riigi haldussüsteemi avatud raamistik (*frameware for open systems*)
• (eeldati, et) Standardite loomise protseduur (näeb välja nii):

1. Ekspertide töögrupid ⇒ Eesti standardiprojektid
2. Eesti firmade hinnangud projektile
3. Korrigeerimine
4. Registreerimine: Eesti standardiamet või/ja rahvusvahelised organisatsioonid.

EESTI '94:

Korraldajad:

Informaatikafond (on olemas vastav lepe Eesti Standardiametiga)
Eesti Tööstuse Keskliit
Eesti Standardiamet

Eesti on juba ka ISO/IEC liige.

Infotehnoloogiliste riigihangete kord.

Praegu teostab seda Eesti Informaatikafond. tulevikus peaksid sellega tegelema hakkama TTÜ ja TÜ.

On olemas sellekohane valitsuse määrus nr 75 (7.03.94). Enamikus arenenud riikeis on selline kord välja töötatud ja varsti on Eestis tulemas ka üldine riigihangete kord.

MILLEKS?

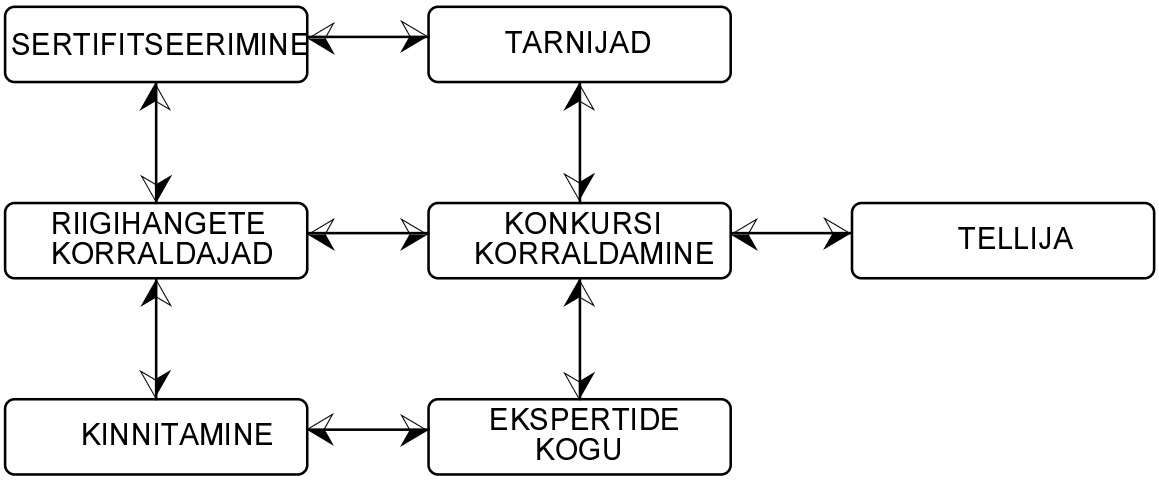
Riigihangete kord peab kindlustama:

• kvaliteetsed hanked (funktsionaalsed, töökindlad, üle riigi ühilduvad...)
• korruptsiooni vältimise (see pole küll lõpuni võimalik)
• raha otstarbeka kulutamise

KASULIK TEADA:

• riigiasutusele - kas ostetav kaup käib riigihangete alla
• infotehnika alal töötajale - kuidas pakkumine käib

Riigihangete kord:



4.3. Kvaliteedistandardid, sertifitseerimine.

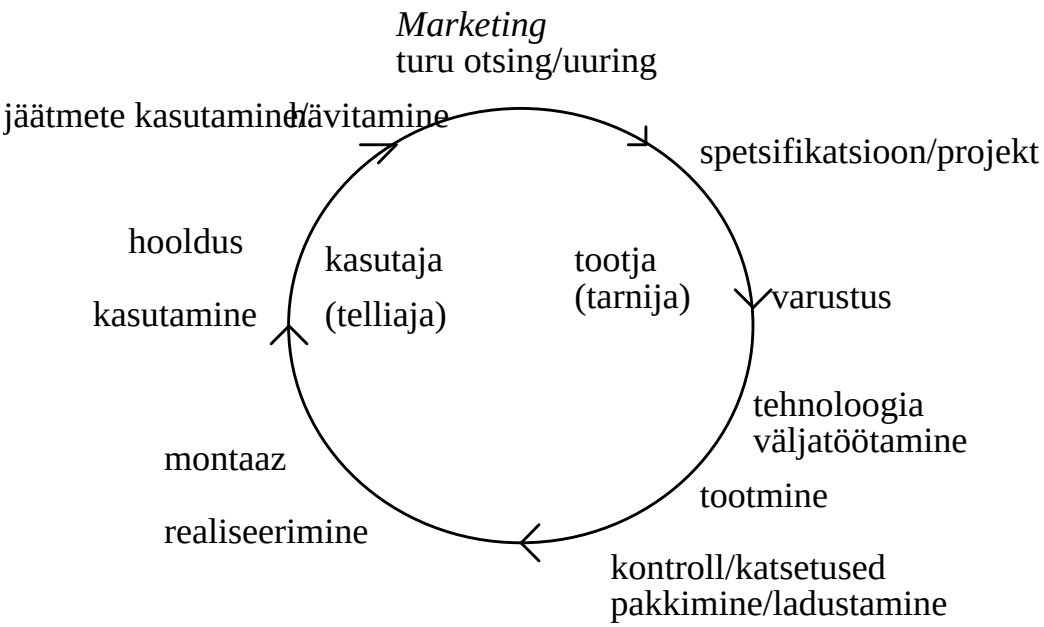
ÜLEVAADE:
ISO 8402 - *Quality vocabulary* (kvaliteedisõnastik)
ISO 9000 - Ülevaade järgnevast 4-st standardist, kasutusjuhised, millal neid kasutada, see on justkui sisukord.
ISO 9001 - Standard kvaliteedi juhtimise jaoks süsteemidele, mis hõlmavad disaini või projekteerimise etappi (tarkvaratootmisel kui elutsükkel).
ISO 9002 - Hõlmab süsteeme, milles põhiline tegevus on tootmine ja installeerimine.

ISO 9003 - Masstootmise standard, lõpliku katsetamise ja inspektsiooni standardid.
ISO 9004 - Üldiselt kvaliteedijuhtimise süsteemidest.
EN29001=BS5750 PART1=ISO9001; Briti standard, mis on aluseks ISO-le.
ISO/DIS 9000-3.- *Guidlines for the Application of ISO 9000 to the Development, Supplay and Maintenance of Software* - tarkvara kvaliteedirakendus.
EN 45 012 - Kehtestab nõuded sertifitseerivatele firmadele.

ISO 10011 - Standard, mis räägib sertifitseerijatest ja sertifitseerimisprotsessist.
'94 ISO 9000 UUEDNDUSED

- neli toote üldkategooriat:
  - riistvara (mitte ainult arvutid, vaid ka tööpingid jms)
  - tarkvara (arvuteile, ...)
  - töötlusmaterjalid
  - teenused
- rohkem efektiivsusele suunatud kvaliteedijuhtimine (rõhk rohkem ettevõttesisesel kvaliteedil)

Kvaliteedi ring:



SERTIFITSEERIMINE:
* Vastavussertifitseerimine - kolmanda osapoole tegevus, mis annab veendumuse, et nõutavalt identifitseeritud protsess või teenus vastab standardile või normdokumendile. (Näiteks: ISO 9000 seeria). Kolmas osapool so. sõltumatu, firma ise ei saa teha sertifitseerimist. Sertifitseerimine ja sertifikaadi ülalhoidmine on kallis lõbu.

KULG:	Informatsiooniline kokkusaamine (kui vaja)	10h
	Avaldus sertifitseerimiseks	
	Eel-audit (kui vaja) (kvaliteedi juhtimine)	20h
	Planeerimine, kokkusaamine	10h
	Audit	80h
	Järel-audit (kui vaja) ⇒ sertifikaat	
	Järgnevad auditid 2x aastas.	

NÄIDE SERTFITSEERIMISE JA SERTIFIKAADI HOIDMISE KALLIDUSEST:
(Soome 1991, 200..300 töötajaga ettevõte)

Sertfitseerimine (120 h)	56.400
Avaldusmaks	3.500
KOKKU:	60.000 FIM (ühekordselt)

Aastamaks	4.700
2 auditit aastas	7.520
KOKKU:	15.000 FIM (iga aasta)

5. Avatud süsteemid.

IDEE: Ühildatavus, vahetatavus, integreeritus.
- süsteemi sees
- süsteemide vahel
- süsteemi ja väliskeskkonna vahel.

MEETOD: (Raam)standardid
Organisatsioon

Kui ühildatavust pole tekib SÕLTUVUS TARNIJAST:
- teenindus (monopoolsed hinnad)
- dumping: - põhikomponent odav ja muud kallid
  - I versiooni odav, järgmised kallid
  - konkurentide väljasurumine

MIS? Põhineb IT-standarditel
SIHID - sõltumatus tarnijast

Ses konspektis püütakse anda :
- üldpilt standardeist
- hangete probleemid
- "valged laigud" õpetamises

AJALUGU: UNIX AT&T *Bell laboratories source code*
POSIX (*Portable Open Systems Interface for Computer Systems*)

JURIIDILINE: Liikmesmaad juhinduvad avalikes IT hangetes
- Euroopa standarditest
- Rahvusvahelistest standarditest, mis on tunnustatud vastaval maal.

STANDARDITE SISU:
- Riistvara
- Kommunikatsioonid
- Tarkvara
- Liidesed
- Süsteemiarendus
- Kvaliteet
- ...

EELISED: - Vabadus toodete / tarnijate valikul (saame neid igal hetkel vahetada)
- ülekantavus (*portability*) tehnoloogia arenedes saame uuemad versioonid sisse vahetada).
- vahetatavus / koostöö
- eri suurusega süsteemide koostöö
- ühine arenduskeskkond (nii riist- kui tarkvara)
- ühine andmekeskkond
- ühine vaade kaitstusele

Tänu eelistele tekivad SÄÄSTUD (Rootsis valitsuse hinnanguil kuni 200 miljonit SEK-i aastas):
- laiem tarnijate valik
- odavam üleminek uuele tehnoloogiale
- madalam hind tarkvara modifikatsioonidele ja ülekandmisele
- vähem väljaõpet, siit madalam hind
- paremad andme / dokumentide vahetuse võimalused

PUUDUSED / RASKUSED / PROBLEEMID:
- millal, kas, kuidas rakendada? (kui suletud süsteem rahuldab täielikult)
- vaja uut riist / tarkvara (pole selge kas, ja millal, kuna see pole hüpe vaid protsess)
- otsus: avatud või mitte?
- ebastandardsed funktsioonid standardites

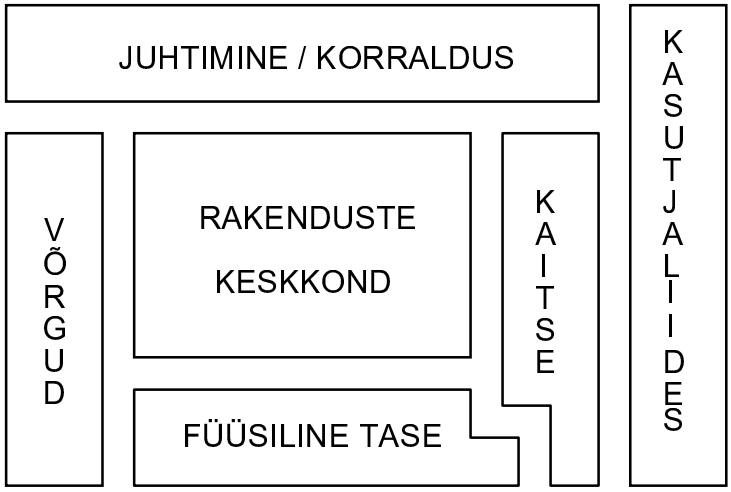
FÜÜSILINE TASE:
- kaabeldus
- võrgutase
- riistvara füüsiline tase

VÕRGUD (KORRALDUS)
- füüsiline tase
- loogiline tase

KAITSE- *secyrity system*
- *protection tehnology*
- *authorization control*

RAKENDUSTE KESKKONNAD
- operatsioonisüsteemid
- süsteemi liidesed
- süsteemi arendus
- programmeerimiskeeled
- andmekäsitlemine (*data handling*)
- *data interchange*
- printerid
- andmekommunikatsioon
- informatsiooni ümbrus
- liidesed kinniste süsteemidega
- rakendusprogrammid

JUHTIMINE / KORRALDUS:
- süsteemi arendus
- süsteemi administreerimine
- testimine / sertifitseerimine
- kvaliteet
- dokumentatsiooD



Üldiselt on põhimõte, et võetakse kõige enamlevinud asjad ja kuulutatakse need avatuks. Eestis peaks olema avatud süsteemide kasutamine lihtsam kui enamikus mujal maailmas, kuna Eestis:

- pole vana firmatark- ja riistvara, sellist "Suurt rauda", mis ka kasu toob on vähe.
- on arvutialane ajalugu lühem
- ressursse oli ja on vähem - seega osteti ja ostetakse standardseid asju

Oluline on Eestis mitte korrata mujal maailmas tehtud vigu.

NÄITEID STANDARDEIST:

* operatsioonisüsteemid ja süsteemiliidesed
  - MS-DOS personaalarvutitele
  - OS, mis rahuldab *x/open portability guide*

UNIX -- - POSIX

* võrgud ja andmekommunikatsioonid
  - Ethernet
  - Token Ring
  - *Client - server tehnology*
  - SOSIP avatud süsteemide
  - UK GOSIP rahvuslikud
  - US GOSIP raamstandardid.
* Arendustarkvara
  - 3 GL (C, GOBOL, ADA, C++, FORTRAN, PASCAL)
  - 4 GL nõudmised graafilistele liidesele, spetsifikatsiooni haldamine, koodi genereerimine)
  - SQL

KASUTAJALIIDES:

- terminalid
- graafika
- klaviatuurid (kindlad klahvikombinatsioonid teevad kindlaid tegevusi (help jms))

INFOKESKKOND

- EDI *Electronic Data Interchange*
- EDI FACT

RIISTVARA

- ekraanid: *de facto* std VT220 (*Digital Equipment*)
- printerid (laserprinterid: *Hewlett-Packard*)
- flopid (3.5" 1.44 Mb, 5.25" 1.2 Mb)
- (*intel inside*)

KAITSTUS

- Kaitstuse hinnang *DoD, Orange Book* (praeguseks juba vananenud), *ITSEC European*
- Kommunikatsioonide kaitstus *DoD, the Red Book*
- Vastutus: Omanik / juhtkond (kes vastutab näiteks läptopi eest, mida kasutavad mitu inimest jne, selle peab kõik paika paneme juhtkond)
- Ohud: info / vahendite hävimine (andmete kustutamine)
  andmete ebaõige muutmine (kellel on õigus andmeid muuta)
  andmete vargus
  andmete lubamatu avalikustamine
  teenuse katkemine
- Kaitse tasemed
  administratiivne
    kaitse poliitika formuleerimine
    vastutuse määramine
    selge süsteemi struktuur ja mõistlikkus (et kaitse ei häiriks normaalset tööd)
  füüsiline
    kaablisüsteemid, lukud, valve
    tulekindlus, elektrisüsteemid,UPS
  loogiline
    juurdepääsu kontroll
    lubatud tegevuste kontroll
    andmekooskõlalisuse kontroll
    tagasihelistamine
    kodeerimine/krüpteerimine
  kaitse õpetamine / teadvustamine

Kaitsele ei ole mõtet kulutada rohkem kui on potentsiaalne kulu kaitsetuse korral.

6. Tarkvara

6.1 Ülevaade arendusstandarditest

MILLEKS TARKVARALE ERALDI STANDARDID:

Tarkvara puhul ei taha hästi töötada üldised kvaliteedijuhtimise reeglid. Tarkvara so. programmid, protseduurid, dokumentatsioon, andmed, mis on vajalikud andmetöötlussüsteemide tööks. Üldiselt on tarkvara võrdlemisi laialivalguv mõiste. Selge pole alati ka vahe tark- ja riistvara vahel.

Tarkvara elemendid: (ANSI/IEEE Std.)

- plaani dokumendid
- spetsifikatsioon, realisatsiooni projekt
- testimise dokumentatsioon
- kasutaja dokumentatsioon
- lähtekood
- tarkvara, mis realiseeritud riistvaras
- aruanded (nt. läbivaatuse kohta)
- andmed (nt. testi tulemused)

STANDARDI OMADUSED:

1. Vahetatvus (pole tarkvaraarenduse puhul eriti oluline. Oluline on et vahetatav on protsess või dokumentatsioon)
2. Kvaliteedi parendamine (tooted, mis on tehtud standardi järgi on kvaliteetsed)
3. Effektiivsus (standardid tõstavad loodetavasti mingi arendusprotsessi effektiivsust, omab mõtet kui tegeletakse paljude projektidega)
4. Mõõtmine (standard võimaldab mingit protsessi mõõta ja teda hinnata)

TARKVARA ARENDUSE STANDARDID

Näiteid (seni) ANSI / IEEE 829-1983 testdokumentatsioon
ANSI / IEEE 1028-1988 läbivaatus / audit
ANSI / IEEE 646, 686
ISO 9000-3 kvaliteedi juhtimine

Liigitusi: seotud kogu elutsükliga / seotud e.t. mõne etapiga (1028/829)
rahvusvahelised / riiklikud (ISO / BS) rahvusvahelised võiksid olla eeskujuks või aluseks rahvuslikele kuid tihti saavad suurriikide standardid aluseks rahvusvahelistele
ametlikud / mitteametlikud näiteks spetsifikatsioonide juures andmevoo- ja olekudiagrammid
ettevõttesisesed / -välised hea oleks kui ettevõttesisesed ühilduksid välistega

Kehameid: ANSI / IEEE, ISO / IEC , BS, MIL, DoD,JSP, IEE,CCITT

TARKVARAARENDUSE STANDARDEID, MIS SEOTUD KOGU ELUTSÜKLIGA

Üldised: ANSI / IEEE 729-1983 terminoloogia
ISO 2382/1 terminoloogia
ANSI / IEEE 1002-1987 standardite ülevaade
SESAM Rootsi avatud süsteemide arendusstandard

Juhtimine / korraldus
IEEE 1058-1987 projektijuhtimine
IEEE 1074-1991 elutsükli arendus
ISO / IEC JTC1/SC7 Nº 962 projektijuhtimine
ISO / IEC JTC1/SC7 Nº 685 arendus / juhtimine

Dokumenteerimine:
IEEE 1063-1987 kasutaja dokumentatsioon
BS 5515-1984 üld-dokumentatsioon
DoD 7935-1983 üld-dokumentatsioon (*Department of Defence*)
IEE tööstuses
ISO 6592-1985 üld-dokumentatsioon

Konfigureerimine
ANSI / IEEE 828-1983 konfiguratsiooni plaan
BS 6488:1984 konfiguratsiooni korraldus projekti versioonid
IEEE 1042-1987 konfiguratsiooni korraldus haldamine, korraldamine jne.

Kvaliteedijuhtimine:
ANSI / IEEE 1028-1988 läbivaatused/auditid
ANSI / IEEE 730-1984 Kvaliteedi juhtimise plaan
ANSI / IEEE 983-1985 Kvaliteedi juhtimise planeerimine
ISO 9000-3 Kvaliteedi juhtimine
MIL, ACAP(NATO)
ISO / IEC JTC1/SC7 Nº646,686 kvaliteedi fakt. ja karakt.

Töökindlus, kaitstus:
IEEE 982.1-1988 töökindlus
IEEE 982.2-1988 juhend eelneva kasutamiseks
IEC TC56 mitukümmend standardit ka tarkvara kohta

ELUTSÜKLI ETAPPIDEGA SEOTUD TARKVARAARENDUSE STANDARDID:

Nõudmised:
ANSI / IEEE 830-1983
BS 6719:1986

Formaalsed spetsifikatsioonid:
ISO 5806:1944 otsustustabelid
CCITT Z.101-104 1984 spetsiaalne keel

Realisatsiooniprojekt:
IEEE 1016-1987
IEEE 990-1986 ADA kui projekteerimise keel
Programmeerimise keeled
vt. avatud süsteemid

Tegelikult on standardeid muidugi palju rohkem ja neid tuleb ka pidevalt juurde.
KASUTAMISVÕIMALUSI:
Kõik tarkvarastandardid on raamstandardid, s.t., et kuna tarkvara on väga lai mõiste on sandardid rohkem ideed ja soovitused.

- Rahvusvahelised standardid on eeskujuks firmasisestele
- Kollektiivne kogemus: olla teadlik mida peetakse maailmas oluliseks
- Eelmisest punktist saame vajaliku kogemuse standardite tellimiseks
- Standardi vormistamine rahvusvahelisel tasemel, et meie standard põhineks mõnel rahvusvahekisel

6.2. ISO 9000-3

Ülevaade:

1) ulatus - mida sisaldab
2) normatiivsed viisid - viited teistele standardeile
3) definitsioonid koos viidetega teistele kvaliteedi ja tarkvara terminoloogia standardeile.

Kvaliteedisüsteemi raamistik (4)

<table>
<tr><td colspan="8">Juhtkonna vastutus (4.1.)</td></tr>
<tr><td colspan="3">Kvaliteedisüsteem (4.2.)</td><td colspan="3">Kvaliteedisüsteemi audit (4.3.)</td><td>Parandavad tegevused (4.4.)</td></tr>
<tr><td colspan="8">Kvaliteedisüsteemi elutsükli tegevused (5)</td></tr>
<tr><td>leping

(5.1.)</td><td>nõudmi-
sed
(5.2.)</td><td>arendus-
plaan
(5.3.)</td><td>kvalit.
plaan
(5.4.)</td><td>projekt &
realisats.
(5.5.)</td><td>test &
valideer.
(5.6.)</td><td>vastuvõtt

(5.7.)</td><td>hooldus

(5.8.)</td></tr>
<tr><td>Kvali-
teedi-
süsteemi
toetavad
tegevused

(6.)</td><td colspan="7">konfiguratsiooni juhtimine / haldamine (6.1.)
dokumentatsiooni haldamine (6.2.)
kvaliteedidokumendid (6.3.)
mõõtmised (6.4)
reeglid, praktikad ja kokkulepped (6.5.)
tööriistad / tehnoloogiad (6.6)
alamtöövõtt (6.7.)
tarkvara allhange (6.8.)
väljaõpe (6.9.)</td></tr>
</table>

4.*Quality system -- framework*

4.1. ***Management responsibility***
4.1.1. ***Supplieris*** **(tarnija)**
4.1.2. ***Purechaiser's*** **(tellija)**
4.1.3. ***Joint revieris*** **ühine koostöö)**

4.1.2. Tellija juhtkonna vastutus

Tarkvara puhul on tellijal suurem vastutus, kuna tarkvara puhul tekib toode mõistliku koostöö tulemusena. Seepärast tellija juhtkond:

- teeb koostööd tootjaga, annab infot
- määrab esindaja, kellel on volitused:

a) defineerida tellija nõudmised (eeldab teadmisi)
b) vastata tootja küsimustele (selleks peab ta teadma milline info on kinnine ja milline mitte)
c) heaks kiita tootja pakkumised (s.t. omab allkirjaõigust)
d) sõlmida leping/lepped
e) kindlustada, et tellija (org.) vaatab lepingu läbi (kokkulepped)
f) defineerida vastuvõtu kriteeriumid ja protseduurid
. . . . jms.

Ilmselt ühest isikust kõige selle tegemiseks ei piisagi aga peab olema üks, kes koguülaltoodud tegevust kordineerib ja kellel on allkirjaõigus.

4.2. *Quality system* (kvaliteedisüsteem)

- dokumenteeritud (kes vastutab milliste protseduuride eest)
- integreeritud elutsüklisse, kvaliteetne on kogu tootmisprotsess, mitte vaid lõppprodukti kontroll
- probleemi ärahoidmine, mitte vaid tagajärgede kõrvaldamine
- sisse viidud, see peab töötama, nurgas sesvatest raamatuist pole kasu
- kvaliteedi käsiraamat (*quality manual*) vms, mille viimane versioon on kõigile kättesaadav
- kvaliteediplaan kui arendusplaani osa s.t., et kõik uuendused tuleks kasutusele võtta kvaliteetsetena, mitte hakata parandama kvaliteeti töö käigus.

4.3. Sissemiste kvaliteedisüsteemide läbivaatused

4.4. Parandavad tegevused

5. Elutsükli etapid

5.1. Lepingute läbivaatus

5.1.1. Üldine

Tootja läbivaatused:

- nõudmised ja lepingu ulatus defineeritud/fikseeritud (hooldusprobleemid, ühilduvus vana tarkvaraga.
- riskid kirjeldatud
- eraomanduse info kaitstud. Et tootja ei anna välja tellija konfidentsiaalset infot ja vastupidi.
- toote erinevused nõudmistest lahendatud. Kui pole otstarbekas või võimalik neid täita. Suulised kokkulepped kipuvad ununema ja seetõttu tuleks kõik töö käigus tehtavad muudatused dokumenteerida.
- tootja on võimeline täitma. Et jätkuksressusrsse: inimesi, arvuteid, ...
- tootja kohustused alltöövõtu suhtes määratud. Tellija ees vastutab alati täitja, mitte alltöövõtja. Isegi kui tellija annab täitjale oma tarkvara, vastutab täitja. (vt ka punktid: 6.7 ja 6.8
- terminoloogia kooskõlastatud. Tuleb teha kindlak, et mõistetakse termineid samamoodi. Näiteks mida tähendab "hooldus 1 aasta", kui on viirus jms

5.1.2. Kvaliteet lepingus

- vastuvõtutingimused peavad olema täpsustatud
- muudatused tellija nõudmistes. Kui midagi osutub ebaotstarbekaks või võimatuks, siis kes tohib muudatusi kinnitada.
- probleemid peale vastuvõtmist. Kuidas lahendada tekkida võivaid probleeme peale seda kui katsetused on läbi tehtud ja süsteem vastu võetud.
- tellija roll peab olema fikseeritud (nõudmised, realisatsoon, vastuvõtt)
- tellija ressursid. Mis siis teha kui jääb puudu tellijapoolsetest ressurssidest, mitte ainult rahast vaid näiteks ka süsteemi katsetamiseks, vms
- standardid / protseduurid
- paljundamine. Mismoodi süsteemi paljundatakse, levitatkse, koopiaid uuendatakse jns

5.2. Nõudmiste kirjeldus.

Silmas tuleb pidada, et nõudmised ei oleks vasturääkivad.

5.2.1. Üldine

5.2.2. Koostöö tellija ja tootja vahel

5.3. Arendusplaneerimine

5.3.1. Üldine

5.3.2. Arendusplaan

5.3.3. Progressikontroll

5.4. Kvaliteedi planeerimine

5.4.1. Üldine

5.4.2. Sisu

5.5. *Design and implementation*

5.6. Testimine ja valideerimine

5.7. Vastuvõtmine (planeerimine, paljundamine, instrueerimine)

6.1. Konfiguratsiooni juhtimine/haldamine

Tarkvaratoode koosneb paljudest komponentidest: funktsioonid, protseduurid, BAT-failid jne. Arenduse käigus tekib paratamatult neist komponentidest pidevalt üha uued versioonid. Mitte ainult uued versioonid vaid ka erinevaiks vajadusiks uued versioonid. Tarkvara versioon moodustub komponentide versioonidest. Ja, et oleks teada missuguse ühiku missugune komponent millise tarkvara millisesse versiooni kuulub ja kus ta füüsiliselt asub jne on tarvis mingisugust haldamissüsteemi.

6.1.1. Üldine

Konfiguratsiooni juhtimine. Süsteem peab:

- ühiselt identifitseerima iga ühiku, millisesse versiooni ta kuulub
- identifitseerima iga ühiku versioon, mis kokku moodustavad tarkvaraversiooni
- identifitseerima iga produkti staatust (nii arendus, üleandmise, kui installatsiooni)
- korraldama ühikute paralleelset muutmist
- kordineerida muutmist eri kohtades, kui kompnendid erinevais paigus
- identifitseerima ja jälgima muudatusi soovitus → ... → väljalause). Kes võib üldse muudatusi sisse viia, kes muudatust testib, kas see on otstarbekas, toob see kaasa kõrvalmõjusi...

6.1.2. Konfiguratsioonijuhtimise plaan

Pisemad ettevõtted püüavad ilma selleta läbi ajada. Kui on suurem firma, kus korraga mitmeid projekte töös, siis läheb seda tarvis küll. ISO ei paku midagi konkreetset välja aga ütleb et hea oleks ku midagi oleks.

6.1.3. Konfiguratsioonijuhtimise plaan

6.1.3.1. Identifitseerimine ja jälgimine

6.1.3.2. Muudatuste korraldus

Et muudatusi võimalikult vähe oleks.

6.2. Dokumendi juhtimine

On ette nähtud bürokraatia selliseks sisseviimiseks, et see tooks kasu, mitte ei segaks ettevõtte tööd.

6.2.1. Üldine: vaja dokumentide haldamise protseduure

- dokumentide valik, milliste dokumentidega tegeletakse.
- protsesside väljatöötamine/kinnitamine
- dokumendi muudatuste protseduurid

6.2.2. Dokumentide tüübid

Kindlaks peavad olema määratud järgmised dokumentide tüübid

- kvaliteedisüsteemi dokumendid
- planeerivad dokumendid
- produkti dokumendid
    - arenduse etapid, sisend
    - arenduse etapid väljund
    - verifitseerimine plaanid/tulemused
    - dokumentatsioon tellijad/kasutajad
    - hooldusdokumentatsioon

6.2.3. Dokumentide kinnitamine ja valjastamine

a) õiged dokumendid, kehtivad versionid, õiges kohas

b) vanad dokumendid eemaldatud

6.2.4. Dokumendid muudatused

- läbivaatused/kinnitamised tavaliselt samade isikute/organite poolt, kes tegid eelmise läbivaatuse.
- neil juurdepääs taustinfole, et nad üldse saaks midagi otsustada kuna enamsti jääb spetsifikatsioonist väheks.
- muudatuste sisu dokumendis või lisas (kui vajalik)
- loetelu kehtivatest dokumentidest (masterlist)
- uued versioonid peale teatud arvu muudatuste

6.6. *Tools and techniques*

6.7. *Purchesing*

6.7.1. *General*

6.7.2. *Assessment of subcontracts*

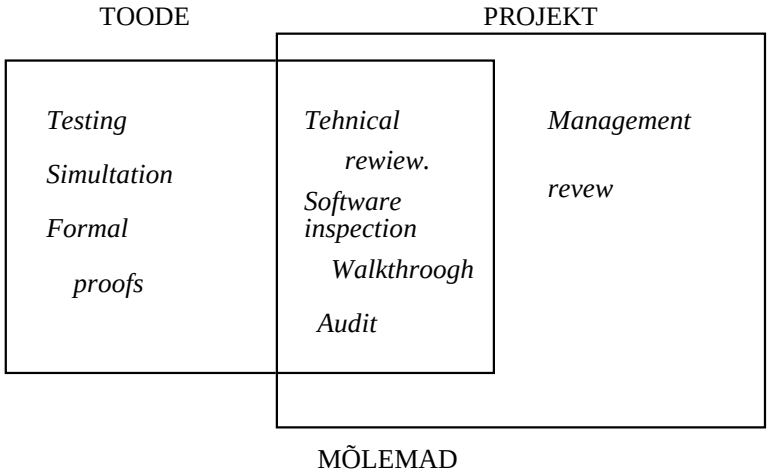
6.7.3. *Validation of purchased product*

6.8. *Included software product.*

Vastutavad mõlemad, kui tellija annab täitjale mingi produkti. Kuigi täitja kontrollib seda, siis tellija vastutab.

Relations of some quality assurance processes t products and projects

ANSI/IEEE Std 1028-1988 (SW Reweves & Audits)



ANSI/IEEE Std 1028-1988 IEEE Standard for software reviews and audits.

AUDIT: • sõltumatu osapool
- hinnatakse tarkvara produkti/protsessi,
- hinnatakse vastavust standardile juhenditele, spetsifikatsioonidele, protseduuridele.
- põhineb dokumentidele, mis annavad objektiivse kriteeriumi:
    - produkti vormi ja sisu kohta
    - protsessi kohta
    - vastavuse mõõtmise kohta

HINDAMINE, ÜLEVAADE *REVIEW* (LÄBIVAATUS, ON MÄRKSA TEHNILISEM):
- hinnatakse tarkvara elemente/projekti, kaasatakse tellijaid, spetsialiste,...
- hinnatakse erinevusi plaanitust
- soovitatakse paraneesi
- formaalne protsess
    - *management* näiteks juhtkonnapoolne läbivaatus
    - *technical* tehniline läbivaatus
    - *SW ispection* tarkvara inspektsioon
    - *walkthrough* arutelu

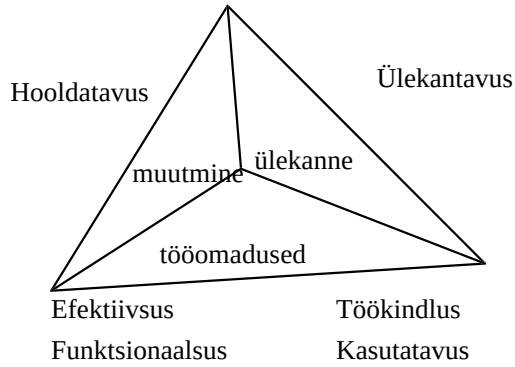
Tarkvara kvaliteedi atribuudid

Et määrata kas toode on kvaliteetne peab meil olema mingi kvaliteedimõõt. Tekib küsimus: mida mõõta? Selleks ongi välja mõeldud tarkvara kvaliteedi atribuudid, et saaks hinnata toote kvaliteetsust: funktsionaalsust, kasutajasõbralikkust jne. ISO 9003 ei esita, ega saagi esitada mingeid konkreetseid nõudmisi. ISO 9003 ütleb vaid, et peavad olema nõuded kvaliteedisüsteemile.

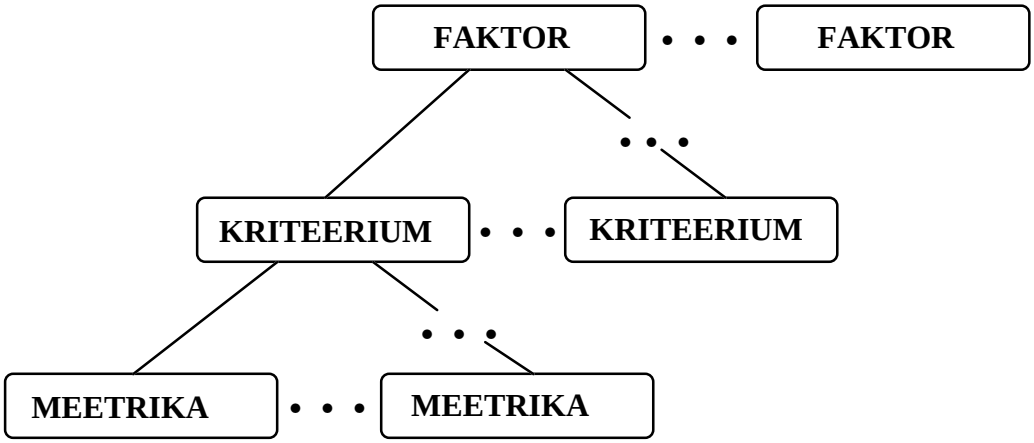
Tegevused ja faktorid

Tarkvaral on kolme liiki omadusi:
- tööomadused
- muutmisomadused
- ülekandmisomadused

Joonisel on vastavalt neile omaduste liikidele välja toodud kuus faktorit, mida võiks mõõta, aga need on liiga üldised. Seetõttu jagatakse need veel omakorda kriteeriumideks ja viimased veel meetrikateks. Meetrika on juba mõõdetav suurus.



ATRIBUUTIDE STRUKTUUR:



ISO/IEC JTC1/SC7 Nr646: Softvare product QUALITY characteristics
ISO/IEC JTC1/SC7/WG3 Nr686 (draft)
Quality subcharacteristis

1. *Functionality* - funktsionaalsus

Subitability - vastavus, kas ja kuivõrd on realiseeritud vajalikud funktsioonid? Näitab otseselt funktsionaalsuse sisu.

Accurateness - täpsus, kas tulemused õiged? Näiteks kas arvutuste täpsus piisav.

Interoperability - koostöövõime teiste süsteemidega, kas töötab näiteks võrgus.

Compliance - vastavus seadustele, standarditele, kas selles paketis teatud asi vastab mingitele normidele. Siin tulevad arvesse nii tarkvaraalased, kui ka muud ühiskonnas tunnustatud reeglid.

Security - katstus, nii programmi kui andmete oma.

2. *Reliability* - töökindlus

Iseloomustab seda kui suur on kindlal platvormil töötades veatõenäosus. Tarkvara puhul ettetulevad vead on projekteerimisvead, realiseerimisvead, ülekandmisvead. Tarkvaral amortisatsiooni pole, tarkvara ei kulu.

Maturity - valmidus, kui tihti vead?

Fault tolerance - veakindlus, kas jätkab tööd tarkvara/ümbruse vigade korral ja kui tuleb siiski töö katkestada, et see toimuks turvaliselt, et süsteem maanduks pehmelt.

Recoverability - taastatavus kui tõsine viga?, kui suur kahju?, kui ruttu töökorda? Et tehtaks varukoopiaid, salvestatataks vaheseise juhuks kui "kaabel kirvega katki tehakse"

3. *Usability* - kasutatavus

Alati pole täpselt teada kes süsteemiga tööle hakkab ja seetõttu on väga raske kriteeriume välja pakkuda.

Understandability - arusaadavus/mõistetatavus loogiline tase. On olemas võrk, hierarhiline ja relatsiooniline konseptsioon: Kõige rohkem on levinud relatsiooniline konseptsioon.

Learnability - õpitavus. Õppimise ajal vajaminevat infot pole töötamise ajal enam tarvis, see muutub siis juba häirivaks.

Operability - kasutamise mugavus / ökonoomsus

4. *Efficiency* - jõudlus

Time behavior - ajajõudlus, aeg kui väga oluline ja sageli erinõudmisi omav ressurss tuuakse eraldi kriteeriumina välja

Resource behavior - ressursijõudlus

5. *Maintainability* - hooldatavus

Analyzability - analüüsitavus, kui raske on leida vigade põhjusi? Sõltub programmi dokumenteeritusest, struktueeritusest, selgusest, loogikast...

Changebility - kui raske on muuta? Sõltub kõige muu hulgas ka sellest kas programm on kompileeritud või interpreteeitud.

Stability - kui tõenäolised muudatused toovad kaasa uusi vigu? Praktikas on kinnitust leidnud põhimõte, et alates mingist kindlast keerukuse astmest toob iga muudatus kaasa vea.

Testability - kui raske on kontrollida muudetud tarkvara?

6. *Portability* - ülekantavus

Adaptability - ülekantavus.kas tarkvara sisaldab vahendeid ülekandmiseks teise, vajalikku keskkonda?

Installability - paigaldatavus, kui raske on paigaldada ettenähtud keskkonda

Compliance - vastavus, sisu sama, aga standard & Co teised, kui funktsionaalses. Vastavus standardeile ja normeile teises keskonnas.

Replaceability - asendatavus ühepoolselt s.t.,et kas meie tarkavara saab kasutada mõne teise tarkvara asemel, mitte vastupidi.

Ülaltoodus võib leida palju vastuolulisi kriteeriume, millede tõttu ei saa mingit ideaalset süsteemi nõuda. Näiteks:

Jõudlus - Funktsionaalsus
Maht - Hooldatavus
Funktsionaalsus - Õpitavus
...

Samas võib leida, et mõni kriteerium parendab teist:

Stabiilsus - veakindlus

Examples of reliability definitions. Do we understand each other?

General (HW,SW) The problability that a system will perform as required for specific period under a given usage.

Software: Set of attributes
Töökindlus iseloomustab kasutaja tööomadusi
Töökindluse kompleksif

- *Trustability*
  - *Matcerity*
    - *Nondeficienty*
    - *Complexity*
    - *Consistency*
  - *Error tolerance* (vastupidavus vigadele)
    - *Integrity*
    - *Robustness*
- *Availability*
  - *Restartability*
  - *Recoverability*

7.Tarkvara meetrikad.

Milleks selline teema?

Tarkvaravalmistamise protsessi on vaja kuidagi mõõta või hinnata, kas või sellepärast, et osata seda tööd rahaliselt väärtustada. Täitja veel oskab oma kogemuste põhjal kuidaga midagi hinnata aga täitja on ses suhtes sootuks rumal. Ka tarkvarafirmasiseselt on see probleem üleval: palju töötajale mõne koodijupi kirjutamise eest makste või kui palju tellija käest mingi toote eest nõuda jms. Sellised probleemid kerkivad ülesse igal tarkvara arenduse sammul.

Tarkvarafirmad esitavad endale umbes sääraseid küsimusi:

- "Kus ma olen?"
- "Kuhu peaksin minema?"
- "Ei saa juhtida, kui ei saa mõõta"(sageli ei saa mõõta, tuleb hinnata)

Tarkvara üldiselt ei saa mõõta.
Meetrikaid kasutatakse tarkvara hindamisel ja arendamisel ning kvaliteedi juhtimisel.
Meetrika s.o.:

- arvutatavad / hinnatavad suurused.
- seotud ülesande, projekti, dokumendi, ettevõttega.
- kasulikud, iseloomustavad olulisi omadusi. Ses suhtes on tarkvara hindamine sarnane kunstiarvustamise või missivalimistega. Ka seal on olemas kindlalt mõõdetavaid suurusi: maali pindala või puusaümbermõõt, mis meile siiski kogu infot kätte ei anna ja samas on palju olulisemaid apekte, mida me kuidagi mõõta ei saa.

NÄIDE: Programmi pikkus (milles mõõdetud? Mis on rida? Kas arvestada kommentaare?)
Milleks kasulik? Milleks mitte? Mis siis saaks kui ainult selle järgi hakataks palka maksma???
MEETRIKA LIIKE:

- hindavad / iseloomustavad programmi (mahtu, keerukust)
- hindavad kvaliteeti
- toetavad arendusprotsessi (prognoosivad töö mahtu, ridade arvu)

KASUTAMINE:

- meetrikate / atribuutide integreerimine
- võrdlemine mingi teadaoleva tasemaga.

Seitsmekümnendate aastate algul hakati välja pakkuma mõõdetavaid meetrikaid. Üks esimesi neist oli:

MCCABE ('76) Programmi keerukuse mõõt.

See põhineb programmi hargnemistel ja seda keerukust saab mõõta neljal moel.
PROGRAMMI KEERUKUS:

= programmi graafi tsüklomaatiline keerukus (cyclomatic complexity) V(G)
= graafi regioonide arv
= E-N+2 (E-kaared, N-tipud)
= P+1 (P-predikaadid)

KASUTAMINE:

- V(G) annab haruadekvaatsete testide arvu
- keerukuse / arendusaja hinnang
- kui projekt valmis, s.t. saab juba ka projekti hinnata (varem kui Halstead')
- saab kasutada arenduses (ühe mooduli V(G) mõõt peaks olema ≤ 10)

Hiljem nähti, et mõõdetavaist meetrikaist üksi kipub nappima ja hakati ka mittemõõdetavaid leiutama.

Halstead ('77) "Tarkvara teadus".

IDEE: Inimese, kes programmi teeb, aju töötab mingeid kindlaid reegleid järgides, mis on paraku küll veel formuleerimata. Neil algoritmeil on omad keerukuse mõõdud. Halstead üritas neid programmist välja leiutada ja mõõta, selleks võttis ta kasutusele järgmised meetrikad:

- meetrikad, mis programmi või täpse projekti põhjal andsid hinnangu
- programmi maht/pikkus
- keerukusele
- keele tasemele

Nende hindamiseks defineeris ta neli suurust:

n_1- erinevaid operaatoreid programmis
n_2- erinevaid operande programmis
N_1- operaatorite esinemiste arv
N_2- operandide esinemiste arv

Operaatorid on: lause lõpp,massiivi indeks, = , IF( ), DO , , , programmi lõpp , =< , >= , GOTO. Operandid on muutujate esinemine. Ja nende järgi üritas ta üht-teist arvutada:

- Programmi pikkus $N=n_1\log_2 n_1+n_2\log_2 n_2$
- Programmi maht $V=N\log_2(n_1+n_2)$

See varieerub sõltuvalt keelest, näiteks :

- mullsortimine FORTRAN-is = 204
- mullsortimine assemleris = 328

JÄRELDUSI: Keskmine keele tase

Inglise keel	2.16 !!!
PL/1	1.53
ALGOL/68	2.12
FORTRAN	1.14
Assembler	0.88

HASTEAD'I MEETRKATE PLUSSID:

- Üks esimesi keerukamaid meetrikaid, mis üldse välja pakuti
- võimaldab objektiivset mõõtmist, kontrollitavad tulemused
- lubab keelte võrdlust, algoritmide mahtu (põhimõtteliselt)
- kooskõla teooria ja praktika vahel

HALSTEAD'I MEETRILKATE MIINUSED

- programmil põhinev, ei saa prognoosida
- liiga "madal" tase, öötab ainult valmis koodi korral
- erinev eri keelte jaoks

NÄIDE:

Programm:

```
      Subroutine Sort(X,N)
      Dimension X(N)
      If (N.LT,2) Return
      Do 20 I = 2,N
         Do 10 J = 1,I
         If (X(I).GE.X(J)) GoTo 10
         Save = X(I)
         X(I) = X(J)
         X(J) = Save
10    Continue
20    Continue
      Return
      End
```

OPERAATORID	ARV	OPRANDID	ARV
Lause lõpp	7	X	6
Massiivi indeks	6	I	5
=	5	J	4
If ()	2	N	2
Do	2	2	2
,	2	Save	1
Programmi lõpp	1		
.LT.	1		
.GE.	1		
GoTo 10	1		
$n_1 = 10$	$N_1 = 28$	$n_2 = 7$	$N_2 = 22$

KVALITEEDI MEETRIKAID (NÄITEID):

Funktsionaalsus→ vastavus

• Spetsifikatsiooni muutmise tase = $\frac{E+M+L}{A}$, kus

E - eemaldatud funktsioonid
M- muudetud funktsioonid
L - lisatud funktsioonid
A - algne funktsioonidede arv

parim väärtus = 0

• Tehnilise ülesande ja spetsifikatsiooni vastavus = $\frac{\text{Funktsioonide arv tehnilises ülesandes}}{\text{Funktsioonide arv spetsifikatsioonis}}$

parim väärtus = 1

• Töökindlus -> valmidus

keskmine tõrgetevaheline aeg = $\frac{\text{Kogu funktsioneerimise aeg}}{\text{Tõrgete arv}}$

parim tulemus = ∞

EKSISTEERIVAID MEETRIKAID:

• ISO/IEC JTC1/SC7/WG3: meetrika-alane standardi prujekt > 110 meetrikat
• HP hooldusalased tarkavara meetrikad: 5 sihti, 31 küsimust, 35 meetrikat.

MEETRIKATE PROBLEEME:

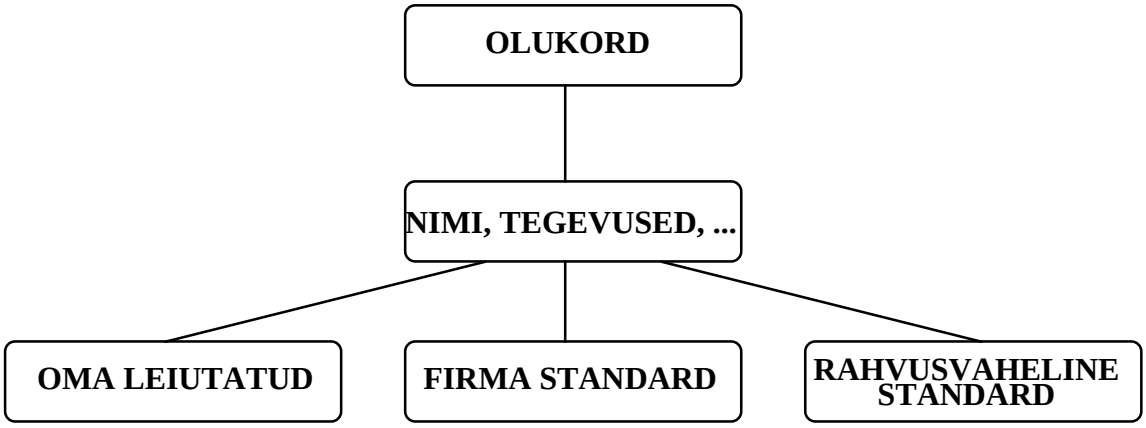
• annavad küll mingi tulemuse, aga on sageli väga raskesti mõõdetavad
• ütleme kliendile, et"Tehnilise ülesande ja spetsifikatsiooni vastavus" = 1.3333.., see arv ei üle talle mitte midagi
•"Mind ei huvita "tehnilise ülesande ja spetsifikatsiooni vastavus", mind huvitab, kas see tarkvara teeb, mida vaja !" → integreerimise probleem

8. Standardite ja kvaliteedijuhtimise meetodite kasutamisest.

Kvaliteedi juhtimise meetodid peavad olema:

• kohandatud ettevõtte vajadustele - universaalsed alati ei sobi.
• lihtsad, piisavad, ühesed
• kooskõlastatud, tutvustatud, sisse viidud nii juhtkonna kui töötajate tasemel.

IDEED:



KUIDAS ALUSTADA?

Kõigepealt peab olema juhatuse otsus, seejärel võiks juhinduda järgmistest punktidest:

- õpi standardeid
- leia oma ettevõtte kõige kriitilisemad kohad (80-20 reegel)
- analüüs - milliseid vigu tekitavad või võivad tekitada
- tekita nende jaoks standardeid
- kui ühe probleeniga korras, võta järgmine ette
- pane kõik süsteemina tööle

---

võib viia lõpuks ISO 9000 sertfikaadi saamiseni