

Ohjelmistotuotanto

Ohjelmistojen testaus 1

1

Laadunvarmistustekniikoita

- Testaus (testing)
 - ohjelman suorittamista tarkoituksena löytää virheitä
- Tarkastukset, katselmukset (inspections, reviews)
 - asiantuntijoiden suorittamia dokumentin (voi olla myös koodi) läpikäyntejä tarkoituksena löytää virheitä
- Osittain samoja virheitä - osittain erilaisia

© Harri Laine, Jukka Paakki 2

Testaus – periaatteita

- Testiajo on onnistunut, jos se on paljastanut ohjelmistosta aikaisemmin havaitsemattoman virheen
- Hyvä testitapaus on sellainen, joka suurella todennäköisyydellä paljastaa aiemmin havaitsemattoman virheen
- Hyvä testaus on tuhoavaa toimintaa, toisin kuin esimerkiksi luova ohjelmointi

© Harri Laine, Jukka Paakki 3

Testaus – termejä

- virhe (error, mistake, bug):
 - ohjelmiston poikkeaminen sen spesifikaatiosta (määrittelystä), virhe esimerkiksi ohjelmakoodissa
- vika (fault)
 - virheellisen ohjelmakohdan suoritus - vika esiintyy siis vain, jos virheellinen ohjelmakohta suoritetaan
- häiriö (failure)
 - viasta aiheutuva ulkoisesti havaittava poikkeama ohjelmiston spesifikaatiosta
 - kaikki virheet ja viat eivät välttämättä johda häiriöön

© Harri Laine, Jukka Paakki 4

Testaus – havaintoja

- Havainnoja:
 - ammattiohjelmoija tuottaa ~1,2 virhettä / 200 riviä ohjelmakoodia
 - uudessa 200000 rivin ohjelmatuotteessa on ~1200 virhettä
 - pitkäaankin käytössä olleissa ohjelmistoissa on noin 1 virhe / 1000 riviä koodia
 - yhden virheen poistamiseen kuluu noin 12 henkilötyötuntia
- Testaus on erittäin tehokas tapa varmistua ohjelman toimivuudesta
- Testaus voi osoittaa ohjelmiston virheellisuuden mutta ei sen oikeellisuutta (Dijkstra)

© Harri Laine, Jukka Paakki 5

Testaus – havainnoja

- Tyypillinen virhejakauma
 - Beizer: Software Testing Techniques, 2nd ed, van Nostrand Reinhold, 1990
 - ohjelmakoodin lauseissa 25%
 - ohjelmakoodin tietorakenteissa 22%
 - spesifikaation toteutustavassa 16%
 - moduulien/komponenttien integroinnissa 9%
 - huolimattomuusvirheitä 9%
 - ...
 - spesifikaatioissa 8%

© Harri Laine, Jukka Paakki 6

Testaus

- Testitapaus:
 - tietty ohjelmiston syötearvojen joukko
- 1. Rakenteellinen eli sisäinen eli lasilaatikkotestaus (structural / glass-box / white-box testing)
 - perustuu ohjelmakoodin mukaiseen läpikäyntiin, eli testattava komponentti on 'läpinäkyvä laatikko', joka paljastaa sisäisen rakenteensa
 - testitapaukset valitaan siten, että ne aiheuttavat ohjelmakoodin mahdollisten suoritusreittien läpikäyntiä
 - ohjelmasilmukoista johtuen kaikkien mahdollisten suoritusreittien läpikäynti ei ole yleensä mahdollista järkevässä ajassa

© Harri Laine, Jukka Paakki

7

Testaus

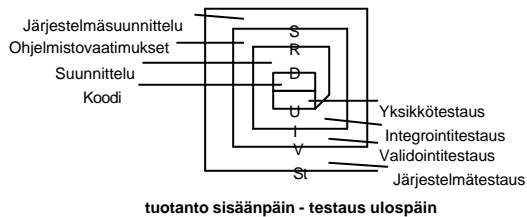
2. Toiminnallinen eli funktionaalinen eli ulkoinen eli mustalaatikkotestaus (functional / black-box testing)
 - perustuu ohjelmiston ulkoisesti havaittavan toiminnan tarkasteluun
 - testattava komponentti on 'musta laatikko', jonka sisään ei nähdä
 - testitapaus -> komponentti -> tulokset (ok?)
 - suuresta syöteavaruudesta johtuen (esim. R: real) kaikkien mahdollisten syötteiden testaus ei ole mahdollista järkevässä ajassa

© Harri Laine, Jukka Paakki

8

Testaus

Spiraalikuvaa ohjelmistotuotantoprosessista



© Harri Laine, Jukka Paakki

9

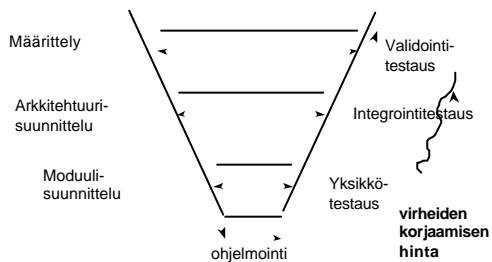
Testaus - vaiheet

- Yksikkötestaus (unit test):
 - testataan erikseen ohjelmiston jokainen komponentti (moduuli, aliohjelma, luokka, ...)
 - tyypillisesti white-box
- Integrointitestaus (integration test)
 - testataan komponenttien yhteistoiminta
- Validointitestaus (validation test)
 - testataan ohjelmiston ja vaatimusmäärittelyn vastaavuus
- Järjestelmätestaus (system test)
 - testataan järjestelmä kokonaisuudessaan (mukana laitteistot, ympäristö) tyypillisesti black-box

© Harri Laine, Jukka Paakki

10

Testausstrategia: V-malli



© Harri Laine, Jukka Paakki

11

Testausstrategia: V-malli

- V-malli kytkee yhteen ohjelmiston rakentamisvaiheen ja testausvaiheen
- Testaussuunnitelma laaditaan kutakin testausvaihetta vastaavassa rakentamisvaiheessa
- Kussakin testausvaiheessa ohjelmiston toimintaa verrataan vastaavan rakentamisvaiheen tuottamaan spesifikaatioon

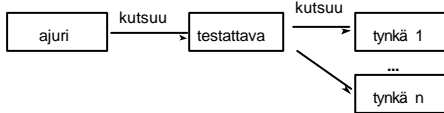
© Harri Laine, Jukka Paakki

12

Testaus – vaiheet

■ Yksikkötestaus

- toimiiko moduuli irrallisena kokonaisuutena?
- toimivatko paikalliset tietorakenteet ja sisäiset suoritusreitit?
- voidaan tarvita erityisiä 'tynkiä' (stub) ja 'ajureita' (driver) korvaamaan testattavan kutsumia ja sitä kutsuvia moduuleja



© Harri Laine, Jukka Paakki

13

Testaus – vaiheet

■ Integroitintestaus

- toimiiko moduulijoukko integroituna kokonaisuutena (kun yksittäiset moduulit on ensin testattu erikseen)?
- toimiiko tiedonvälitys moduulien rajapinnolla?
- toimiiko moduulien kytkentä?

■ Tapa integroida vaikuttaa testaukseen

- big bang - kaikki valmiiksi, sitten kasaan ja testataan
- poluittain (threads) - toteutetaan ja testataan suorituspolku kerrallaan
- hierarkian suhteen top down (tarvitaan tynkiä) tai bottom up (tarvitaan ajureita)
- useimmiten bottom up tehokkain

© Harri Laine, Jukka Paakki

14

Testaus – vaiheet

■ Validointitestaus / järjestelmätestaus (usein yhdistetty)

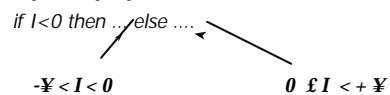
- toimiiko kokonaisjärjestelmä niin kuin tuotantoprosessin alussa on määritetty?
- millainen on ulkoisesti havaittava toiminta, suorituskyky, varmuus, siirrettävyys, virheensietokyky, ylläpidettävyys...?
- toimivatko ohjelmisto ja laitteisto yhdessä?
- testilajeja:
 - **alfa-testaus**
 - asiakkaan suorittama tuottajan tiloissa ja laitteilla
 - **beta-testaus**
 - asiakkaan suorittama omassa todellisessa käyttöympäristössään
 - **hyväksymistestaus**, jonka osana voi olla **käytettävyystestaus**

© Harri Laine, Jukka Paakki

15

1. Toiminnallinen testaus (black-box)

- testaus perustuu useimmiten syötemuuttujien arvoalueen (domain) osittamiseen => arvoalueetestaus (domain testing)
- parametrin tyyppi (jos tiedossa) kertoo arvoalueen
var I: integer; I kokonaisluku (välillä -¥ < I < + ¥)
- ohjelma synnyttää arvoalueita



© Harri Laine, Jukka Paakki

16

Toiminnallinen testaus (black-box)

■ Arvoalueestauksen periaate:

- valitaan yksi testitapaus arvoalueen sisältä
- valitaan arvoalueen rajat
- valitaan yksi testitapaus välittömästi arvoalueen rajojen ulkopuolelta
- esim: spesifikaatiossa arvoalue: $0 \leq I \leq 100$, testiaineisto:
 - I=50 (sisällä, voisi olla yhtä hyvin vaikkapa 78)
 - I=0 (alarajalla)
 - I=100 (ylärajalla)
 - I= -1 (alarajan ulkopuolella)
 - I=101 (ylärajan ulkopuolella)

© Harri Laine, Jukka Paakki

17

Toiminnallinen testaus (black-box)

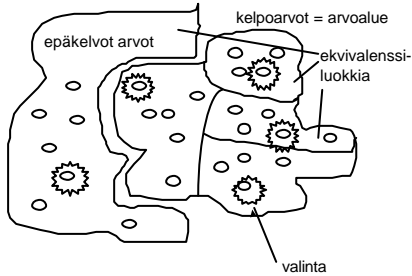
■ Ekvivalenssiluokat

- T.J.Ostrand, M.J.Balcer: The Category-Partition Method for Specifying and Generating Functional Tests, CACM, 31, 6, 1988, pp 676-686
- jokainen arvoalue jaetaan **ekvivalenssiluokkiin**, joille pätee:
 - jokainen tiettyyn ekvivalenssiluokkaan kuuluva arvo käyttäytyy testauksen kannalta 'samalla tavalla'
 - jokainen luokan arvo paljastaa virheen yhtä suurella todennäköisyydellä
 - jos toiminta on virheellistä arvolla α , se on virheellistä muillakin ko. luokan arvoilla
 - jos toiminta on oikein arvolla α , se on oikein myös muilla ko. luokan arvoilla
- testitapauksiksi kustakin luokasta 1 (tai muutama) arvo

© Harri Laine, Jukka Paakki

18

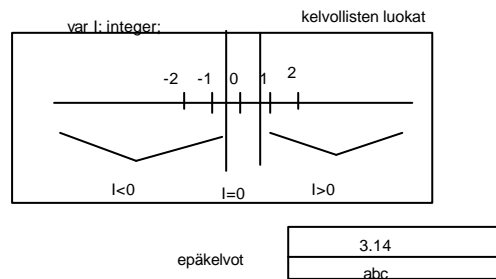
Toiminnallinen testaus (black-box)



© Harri Laine, Jukka Paakki

19

Toiminnallinen testaus (black-box)

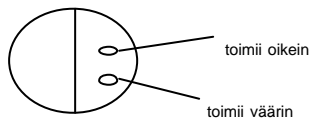


© Harri Laine, Jukka Paakki

20

Toiminnallinen testaus (black-box)

- Testauksen aikana ekvivalenssihypoteesi saattaa osoittautua pätemättömäksi

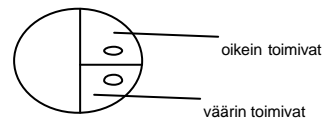


© Harri Laine, Jukka Paakki

21

Toiminnallinen testaus (black-box)

- Luokkarajoja on tällöin muutettava



© Harri Laine, Jukka Paakki

22

Tekijäperustainen testaus (category partitioning)

- Jaa ohjelmisto spesifikaation perusteella erikseen testattaviksi osajärjestelmiksi
- Tunnista jokaisen osajärjestelmän syötemuuttujat
 - testitapaus on syötemuuttujien alustusarvojen joukko
- Määrittele jokaiselle syötemuuttujalle sen tekijät
 - tekijä, kategoria (category) on muuttujan keskeinen ominaisuus
 - esim. järjestettävä taulukko
 - tekijöitä: taulukon koko, alkioiden tyyppi, alkioiden minimiarvo, alkioiden maksimiarvo, minimiarvon sijainti, maksimiarvon sijainti - asiat jotka ovat oleellisia järjestämisen toimimisen kannalta

© Harri Laine, Jukka Paakki

23

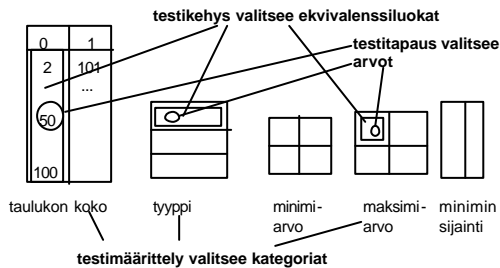
Tekijäperustainen testaus (category partitioning)

- Määrä jokaiselle kategorialle arvojen ekvivalenssiluokat (choice)
 - esim. järjestettävä taulukko / taulukon koko: {koko=0} {koko=1} {2<=koko<=100} {koko>100}
- Tuota testimäärittely jokaiselle osajärjestelmälle
 - kategorialista
 - kunkin kategorian ekvivalenssiluokat
 - näiden perusteella voidaan määrätä **testikehykset** (test frame)
 - testikehys muodostetaan valitsemalla joukko kategorioita
 - tarkoituksena on muodostaa testitapaus valitsemalla ensin yksi ekvivalenssiluokka kuhunkin testikehyksen kategoriaan liittyen ja sitten yksi arvo kustakin valitusta luokasta

© Harri Laine, Jukka Paakki

24

Tekijäperustainen testaus (category partitioning)



© Harri Laine, Jukka Paakki

25

Tekijäperustainen testaus (category partitioning)

- Lisää testimäärittelyihin rajoitukset, jotka koskevat kategorioiden ja ekvivalenssiluokkien mukaanottamista testikehyksiin
 - esim. jotkut ekvivalenssiluokat eivät voi olla samaan aikaan voimassa, kuten {taulukon koko=0} ja {minimin sijainti>100}
 - rajoituksilla vähennetään testikehysten ja -tapauksen lukumäärää
- Tuota testikehukset automaattisesti (testaustyökalulla)
- Tuota testikehyksiin liittyvät testitapaukset (testaustyökalulla)
- Yhdistä loogisesti yhteenkuuluvat testitapaukset testijonoiksi (test script)

© Harri Laine, Jukka Paakki

26

Tekijäperustainen testaus (category partitioning)

- Testaa osajärjestelmät suorittamalla testijonot
- Ylläpidä testimäärittelyjä ja -tuloksia testitietokannassa
 - jos saman ekvivalenssiluokan testitapaukset aiheuttavat ristiriitaista toimintaa, muuta luokkajakoa

© Harri Laine, Jukka Paakki

27

Ekvivalenssiluokat

- Luokittelu on pyrittävä tekemään niiden ominaisuuksien mukaan, jotka todennäköisimmin aiheuttavat virheitä
- Testitapauksia on valittava luokan koon mukaan :
 - iso luokka - enemmän tapauksia luokituksen epäonnistumisen huomaamiseksi
- Huono luokittelu paljastaa (yleisrasitteen takia) virheitä tehottomammin kuin puhtaasti satunnainen testaus, jossa koko arvoalue tulkitaan yhdeksi ekvivalenssiluokaksi ja testitapaukset valitaan satunnaisesti

© Harri Laine, Jukka Paakki

28

Toiminnallinen testaus – esimerkki

Find (document, text, direction, match case)

- *document* : työn alla oleva tekstidokumentti
 - *text* : dokumentista etsittävä merkkijono
 - *direction (down, up)* : etsintäsuunta suhteessa kohdistimen (kursorin) nykyiseen sijaintiin
 - *match case (yes, no)* : vaikuttaako kirjainten koko hakuun vai ei (so., tulkitaanko iso ja pieni kirjain samoiksi)
- myös *kohdistimen sijainti* vaikuttaa operaation toimintaan; jätetään tässä yksinkertaisuuden vuoksi pois

© Harri Laine, Jukka Paakki

29

Toiminnallinen testaus – esimerkki

Ekvivalenssiluokat :

- *text* :
 - {merkkijonot, joissa pieniä kirjaimia muttei isoja}
 - {merkkijonot, joissa isoja kirjaimia muttei pieniä}
 - {merkkijonot, joissa sekä pieniä että isoja kirjaimia}
 - {merkkijonot, joissa ei lainkaan kirjaimia}
 - {tyhjät (laittomat) merkkijonot}
- *direction* : {down}, {up}
- *match case* : {yes}, {no}
- *document* : {found}, {not found} – löytyikö vai ko ei

© Harri Laine, Jukka Paakki

30

Menetelmä : ekvivalenssiluokkien yhdistely

text					dir.		match		doc.	
pk	ik	pik	ei-pik	e	d	u	y	n	f	n-f
x					x		x		x	
x					x		x			x
x						x	x		x	
x						x	x			x
...										
x						x	x			x
	x				x			x	x	
	...									
				x		x		x		x

© Harri Laine, Jukka Paakki 31

Toiminnallinen testaus – esimerkki

(Riippumattomien) yhdistelmien eli kombinaatioiden määrä = suoritettavien testien määrä:

$$E_1 * E_2 * \dots * E_k$$

E_j = parametrille / määriteltyjen ekvivalenssiluokkien määrä

Tässä: $5 * 2 * 2 * 2 = 40$

Huom: Joitakin (laittomia syötearvoja sisältäviä) yhdistelmiä ei ehkä voi suorittaa, mutta sekin on testattava!

© Harri Laine, Jukka Paakki 32

Toiminnallinen testaus – esimerkki

Testitapausten hahmot (40 kpl):

- 1 text : pieni-kirjain, direction : down, match case : yes, document : found
- 2 text : pieni-kirjain, direction : down, match case : yes, document : **not found**
- 3 text : pieni-kirjain, direction : **up**, match case : yes, document : found
- 4 text : pieni-kirjain, direction : up, match case : yes, document : **not found**
- ...
- 40 text : **empty**, direction : up, match case : no, document : not found

© Harri Laine, Jukka Paakki 33

Toiminnallinen testaus – esimerkki

- Jokaiselle hahmossa esiintyvälle ekvivalenssiluokalle valitaan sitä edustava parametrin arvo
- Samasta ekvivalenssiluokasta valitaan eri testitapauksissa eri arvoja
- Valitaan raja-arvoja, mikäli mahdollista
 - tekstille sekä lyhyitä että pitkiä merkkijonoja
 - tekstille koko merkkistöä
 - kohdistin sekä dokumentin alussa että lopussa

© Harri Laine, Jukka Paakki 34

document	text	direction	match case
This beautiful text	(1) bea	down	yes
This beautiful text	(2) beatles	down	yes
This beautiful text	(3) lbea	up	yes
This beautiful text	(4) lbea	up	yes
This beautiful text	(5) %1bea	down	no
This beautiful text	(6) %1bea	down	no
This beautiful text	(7) b	up	no
This beautiful text	(8) bea	up	no
This beautiful text	(9) BEA	down	yes
This beautiful text	(10) BEAT	down	yes
This beautiful text	(11) THIS	up	yes
This beautiful text	(12) TH2S	up	yes
This beautiful text	(13) HIS	down	no
this beautiful text	(14) S	down	no
this beautiful text	(15) HIS %&	up	no
This beautiful text	(16) #& BE	up	no

© Harri Laine, Jukka Paakki 35

document	text	direction	match case
This Beautiful Text	(17) Text	down	yes
This Beautiful Text	Text	down	yes
THIS IS beautiful text	IS is	up	yes
This is beautiful text	IS is	up	yes
This text 1-99	EXT 1	down	no
This text 1 and text 2	eXT 1	down	no
This was beautiful text	His Was Beati	up	no
(This) (Was) (12) text	aS()	up	no
123 one-two-three	(25) 123	down	yes
One-two-three 1-2-3	12-3	down	yes
This &007mess	&	up	yes
This Bloody Mess	#%	up	yes
(This) (was) (one) (2)	21	down	no
Q987654321! "#& *//	7654321#	down	no
11273#45%6&7/8/9)-oop&	#45%6&7/8/9)	up	no
This #& beautiful text	(32) 22	up	no
This is beautiful text	(33)	down	yes
1 or two		down	yes
1 or two		up	yes
QK1+(8)Those		up	yes
! & 2		down	no
! & 2		down	no
This #& beautiful text		up	no
-	(40)	up	no

© Harri Laine, Jukka Paakki 36

Jos kombinaatioita on liikaa, pitää niitä karsia
 "Laiskan miehen menetelmä", jossa vaaditaan vain, että jokaisen
 ekvivalenssiluokan on oltava mukana *jossakin* testissä (vähintään yhdessä), saattaa
 johtaa harhateille:

<i>text</i>					<i>dir.</i>		<i>match</i>		<i>doc.</i>	
<i>pk</i>	<i>ik</i>	<i>pik</i>	<i>ei-pik</i>	<i>e</i>	<i>d</i>	<i>u</i>	<i>y</i>	<i>n</i>	<i>f</i>	<i>n-f</i>
x					x		x		x	
	x				x		x		x	
		x			x		x		x	
			x		x		x		x	
				x		x		x		x

