



TTY:n Testauspäivät, Tampere
15.8.2006

**Lyhyt johdatus ketterään testaukseen
eli
Ketterän ohjelmistokehityksen laatukäytäntöjä**

Juha Itkonen
SoberIT
Teknillinen korkeakoulu
Juha.Itkonen@tkk.fi



Ketterä ohjelmistokehitys

- ❑ Ketteriä menetelmiä on paljon
 - ❑ ASD, XP, Scrum, Crystal, FDD, DSDM, "pragmatic programming", ...
- ❑ Kaikissa tarve vastata jatkuvaan muutokseen
- ❑ Arvostavat enemmän
 - ❑ **yksilöt ja kommunikointi** vs. prosessit ja työkalut
 - ❑ **toimiva ohjelmisto** vs. kattava dokumentaatio
 - ❑ **asiakasyhteistyö** vs. sopimusneuvottelu
 - ❑ **muutokseen vastaaminen** vs. suunnitelman noudattaminen
- ❑ + 12 yksityiskohtaisempaa periaatetta
- ❑ Eroavat yksityiskohtaisissa projektitason toimintatavoissa



Agile Software Development Manifesto

Individuals and interactions over processes and tools
Working software over comprehensive documentation
Customer collaboration over contract negotiation
Responding to change over following a plan

<http://www.agilemanifesto.org/>



Joitain ketteriä periaatteita

- Satisfy the customer through **early and continuous delivery** of valuable software.
- **Working software** is the primary measure of progress.
- Deliver working software frequently, from a **couple of weeks to a couple of months**.
- Welcome **changing requirements**, even late in development.
- The most efficient and effective method of conveying information to and within a development team is **face-to-face conversation**.
- Business people and developers must **work together daily** throughout the project.
- **Simplicity**--the art of maximizing the amount of work not done--is essential.

Build projects around **motivated individuals**.
Give them the environment and support they need,
and **trust them to get the job done**.



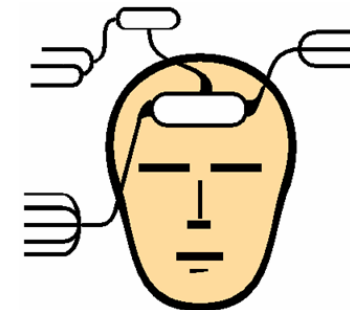
Ketteryyttä ohjelmistokehitykseen - entä testaukseen?

- ❑ Ketteryys ohjelmistokehityksessä
 - ❑ Toimivalla ohjelmiston asettaminen tärkeimmäksi
 - ❑ Muutokseen vastaaminen – ei muutoksen vastustaminen
 - ❑ Nopea kehitys- ja julkaisusykli
 - Iteratiivinen ja inkrementaalinen prosessi
 - ❑ Yhdessä tekeminen ja tehokas kommunikointi
 - Myös asiakkaan kanssa

- ❑ Testauksen näkökulmasta
 - ❑ Ketteryys asettaa monia haasteita
 - ❑ **Ketterässä kehityksessä myös laadunvarmistus on ketterää**
 - ❑ **Ketteryydestä voi ottaa paljon oppia**



Testaus ketterässä ohjelmistokehityksessä
vs.
Ketterä ohjelmistotestaus





Miten ketterät laatuikäytännöt toimivat?





Laadunvarmistus ketterässä kehityksessä

- ❑ Suunnitelmaperustainen laadunvarmistus ei toimi
 - ❑ V-mallia ja vesiputousta ei löydy
 - ❑ Dokumentaation ja speksien rooli on toissijainen
 - ❑ Tekemisen rytmi on liian tiheä
 - ❑ Roolit ja vastuut on jaettu eri tavalla
- ❑ Laadunvarmistus on erottamaton osa ketterää kehitystä
 - ❑ Ei erillinen toiminto
- ❑ Kehittäjillä päävastuu laadunvarmistuksesta
 - ❑ Testaaja tiimin jäsenenä
 - ❑ Kaikki testaavat

Ketterän laadunvarmistuksen kivijalkoja

- ✓ säännöllinen ja tiheä julkaisurytmi
- ✓ kommunikointi ja yhdessä tekeminen
- ✓ kurinalaiset matalan tason (laatu)käytännöt
- ✓ testivetoinen ohjelmistokehitys



Rytmi



- ❑ Jatkuva yksikkötason integrointi- ja testaussykli
- ❑ Lyhyet inkrementaaliset julkaisusykli
- ❑ Ominaisuuksien saaminen valmiiksi ja testattua lyhyellä syklillä – pienissä palasissa
- ❑ Laadun rakentaminen ja laatutason seuraaminen tiheällä rytmillä
 - ❑ Viat ja ongelmat löytyvät aikaisessa vaiheessa
 - ❑ Todellinen laatutaso ja eteneminen paremmin näkyvää
 - ❑ Testausvaihe paremmin ennustettava
 - ❑ Testaus ja käyttöönotto vaiheen riskit pienempiä

Mahdollistaa ketterän reagoimisen muutoksiin



Kommunikointi



- ❑ Kommunikointi ketterässä kehityksessä ei toimi perinteiseen tapaan
 - ❑ Ei niin, että muut kommunikoivat testaajille vaatimusten ja suunnitteludokumenttien muodossa ja testaajat vastaavat testisuunnitelmien ja bugiraporttien muodossa.
- ❑ “So we can let free of the illusion that documents will save us. We can view them as they are: interesting texts, partly fictional, often useful.” - Brian Marick





Yhdessä tekeminen




- ❑ Yhdessä tekeminen on tehokas keino tiedon ja ymmärryksen välittämiseksi
- ❑ Parantaa yhteistä ymmärrystä, helpottaa kommunikointia
 - ❑ Vähentää väärinkäsityksiä ja arvailujen varassa etenemistä
 - ❑ Tehostaa ja nopeuttaa tiedon siirtymistä
- ❑ Etenkin ketterien menetelmien omimmilla alueilla
 - ❑ Kun tehdään uusia asioita joita ei etukäteen tunneta
 - ❑ Kun vaatimukset ovat epämääräiset ja muuttuvat nopeasti
 - ❑ Kun asiakas on helposti käytettävissä





Matalan tason laatukäytännöt - toteutuksen kivijalka

- ❑ Tiukasti toteutuksen rytmiin kytketyt heartbeat-laatukäytännöt
 - ❑ Jokaisen yksittäisen tehtävän laadun varmistus 
 - ❑ Laadun rakentaminen
 - ❑ Osa toteutustehtävää – ei lykätä tuonnemmaksi
 - ❑ Esim. yksikkötestaus, pariohjelmointi, jatkuva integrointi, refaktorointi, ...
 - ❑ Laatutason takaaminen jo ennen kuin kehittäjä päästää koodin käsistään
 - ❑ Testaajalla ja kehittäjällä sama tavoite: laadukas lopputuote
 - ❑ Työskentelevät yhdessä
 - ❑ Testaaja auttaa omalla erikoisosaamisellaan kehittäjää
- Valmiit ja toimivat ominaisuudet tärkeämpiä kuin dokumentaatio
 - Muutoksia pystytään tekemään kun perusta on kunnossa



Testivetoisuus

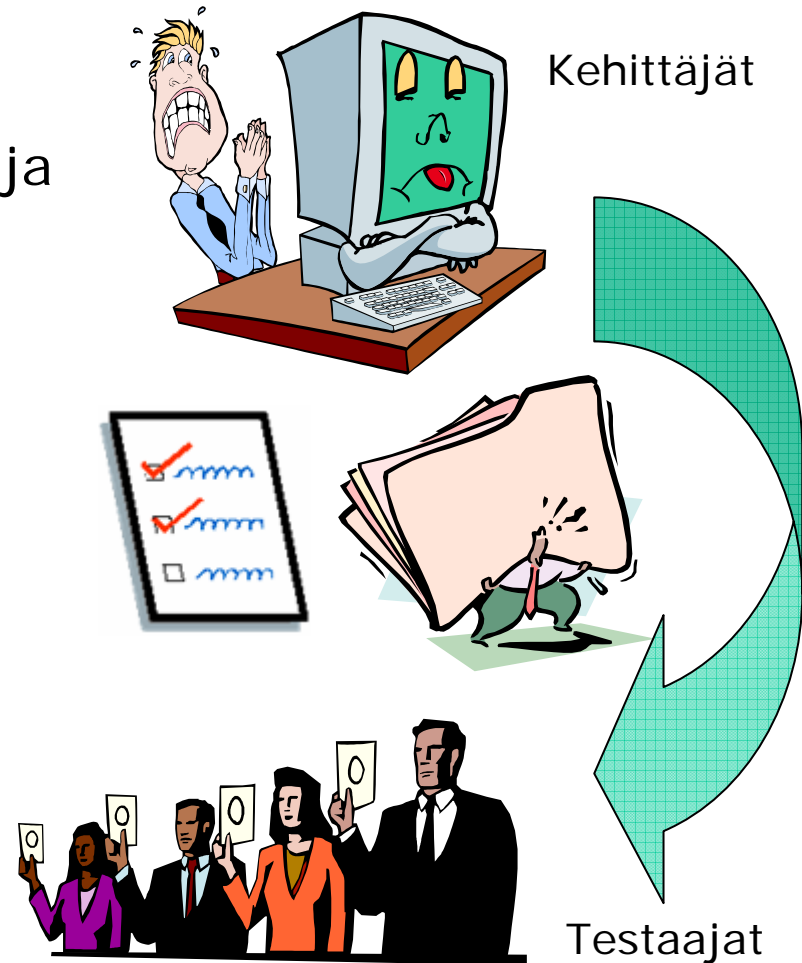


- ❑ Kehittäjien *Test-Driven Development* (test-first programming) tekniikka yksikkötestaustasolla on saanut paljon suosiota
- ❑ **Testivetoisuus on tärkeää myös asiakasvaatimusten ja hyväksymistestauksen tasolla**
- ❑ Ketterä hyväksymistestaus tehdään asiakkaan ja toteutustiimin yhteistyönä
 - ❑ Testeillä dokumentoidaan ja konkretisoidaan vaatimuksia ja niiden yksityiskohtia
- ❑ Testit ohjaavat toteutustyötä
 - ❑ eivät pelkästään mittaa lopputulosta
 - ❑ automatisoituina antavat nopeaa palautetta kehittäjille
- ❑ Testit toimivat etenemisen mittarina



Tavallinen toimintatapa

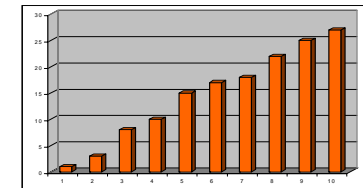
- ❑ Kehittäjät toteuttavat oivallisia ominaisuuksia ja tuottavat laadukasta koodia vaatimusten ja speksien mukaan
- ❑ Testaajat tahollaan suunnittelevat mahdollisimman ovelat testit, paljastaakseen ongelmia kehittäjien koodissa
- ❑ Kun koodi kehittäjien mielestä valmista, testaajat käyttävät nerokkaita testejä vikojen paljastamiseksi kehittäjien koodista





Testivetoisuus: Test-Driven toimintatapa

- ❑ Kehittäjät, testaajat ja asiakas suunnittelevat yhdessä mahdollisimman hyvät testit
 - ❑ jotka kuvaavat miten ominaisuuksien halutaan toimivan
 - ❑ jotka paljastaisivat merkittävät ongelmat toteutuksessa (riskit)
 - ❑ jotka dokumentoivat tärkeitä yksityiskohtia ja potentiaalisia ongelmakohtia toteutuksessa etukäteen
- ❑ Kehittäjät käyttävät näitä testejä ohjaamaan toteutustyötä
 - ❑ mitä pitää saada aikaiseksi, milloin se on valmis
 - ❑ mikä on tärkeää ja mihin kiinnittää huomiota
- ❑ Testitulosten säännöllinen seuraaminen kommunikoi toteutuksen etenemistä toimivien ominaisuuksien muodossa
 - ❑ Ketterien periaatteiden mukaisesti
- ❑ Testauslähtöinen ohjelmistokehitys edellyttää ainakin jonkinasteista testien automatisointia





Kysymyksiä ja kommentteja?



Yhteystiedot

Juha Itkonen

juha.itkonen@hut.fi

+358 9 451 6041

<http://www.soberit.hut.fi>

<http://www.soberit.hut.fi/jitkonen>