Teksti: Timo Jokela

FT Timo Jokela on toiminut käytettävyysalalla 90-luvulta lähtien: VTT:llä, Nokialla, Oulun yliopistolla ja Joticon Oy:ssä. Hän on kehittänyt käytettävyyssuunnittelun prosesseja ja –menetelmiä, sekä niiden arviointimenetelmiä, ja tehnyt käytettävyyteen liittyviä julkaisuja yli 80. Hän on mukana käytettävyyden ISO -standardoinnissa, sekä alan kansainvälisissä verkostoissa.

Sama ongelma – erilaiset tulokset

Miten varmistaa käytettävyystestauksen laatu?

Käytettävyystestaus on keskeinen käytettävyyden varmistamismenetelmä. Käytettävyystestaukseen menetelmänä liittyy kuitenkin laatuongelmia: tutkimusten mukaan testauskäytännöt ja testien tulokset riippuvat voimakkaasti siitä, kuka sattuu tekemään testit. Testien tekijöiden tulisi varmistaa testaustoimintansa laatu arvioimalla kriittisesti omia käytäntöjään ja oppimalta muilta. Käytettävyystestien asiakkaiden kannattaisi käyttää useampaa käytettävyysnäkemystä, ja harkita muitakin käytettävyystoimenpiteitä.

CUE= comparative usability evaluation

Tutkimukset osoittavat käytettävyystestausten laatuongelmat

Käytettävyystestaus on ehkä keskeisin yksittäinen menetelmä sovellusten käytettävyyden varmistamisessa. Käytettävyystestausta tehdään yritysten ja organisaatioiden sisäisesti, sekä kaupallisten toimijoiden toimesta.

Käytettävyystestauksellahan tarkoitetaan menetelmää, jonka pääpiirteitä ovat etukäteen valmistellut testitehtävät, testitehtävien teettäminen sovelluksen loppukäyttäjillä, ja tulosten perustuminen käyttäjien havainnointiin (miten käyttäjien suoriutuvat annetuista tehtävistä). Mutta miten hyvää laatua tällainen käytettävyystestaus on menetelmänä? Miten laadukkaita ovat käytettävyystestausten tulokset?

Kuvitellaan tilanne, että usea käytettävyystestausryhmä testaa saman sovelluksen. Jos käytettävyystestaus on menetelmänä laadukas, niin jokaisen ryhmän

- tulisi löytää samat ongelmat
- tulisi päätyä samoihin tuloksiin
- tulisi raportoida ne samalla tavoin
- ryhmät olisivat yhtä mieltä ongelmien vakavuudesta ja niiden luokittelusta

Asiaa on tutkittu tieteellisesti kansainvälisissä vertailututkimuksissa. Näiden vertailevien käytettävyysarviointien (”comparative usability evaluation, CUE”) vetäjänä on toiminut tunnettu tanskalainen käytettävyyskonsultti Rolf Molich.

CUE -tutkimusten idea on siis se, että annetaan sama sovellus usealle eri käytettävyystiimille itsenäisesti arvioitavaksi, ja verrataan saatuja tuloksia. Testien tekijät ovat olleet ammattimaisia käytettävyystoimistoja tai –ryhmiä: niin kaupallisia toimistoja kuin yliopiston käytettävyysryhmiä. Käytettävyystiimien tekemät evaluoinnit olivat pääasiassa käytettävyystestauksia; joissakin CUE -tutkimuksissa on tehty myös asiantuntija-arviointityyppisiä evaluointeja.

CUE -tutkimusten tuloksia voidaan pitää jokseenkin tyrmäävinä: käytettävyystestausten tulokset erosivat huimasti riippuen siitä, kuka testejä oli tehnyt. Tiivistettynä

- ryhmät eivät raportoineet kuin muutaman saman ongelman (joissakin tapauksissa ei yhtään!)
- ryhmät löysivät enimmäkseen vain ”omia” ongelmia, ts. ongelmia, joita kukaan muu ryhmä ei löytänyt

Kuvassa 1 on esitetty esimerkkejä CUE -tutkimusten tuloksista.

Lisäksi löytyi merkittäviä eroja testien suorittamisessa: siitä, miten testejä raportoitiin, miten testitehtävät oli laadittu jne. Lisätietoja CUE -tutkimuksista löytyy mm. sivuilta www.dialogdesign.dk.

CUE –tutkimusten tuloksista on vaikea tehdä muuta johtopäätöstä, kuin että käytettävyystestaus menetelmänä ei ole – ainakaan tänä päivänä - tieteellisesti vahva. Voidaankin puhua mieluummin kehikosta (”framework”) kuin menetelmästä: on tiettyjä yhteisiä piirteitä – loppukäyttäjät mukana, annetut testitehtävät, jne. – mutta tarkemmalla tasolla testaus on tekijänsä näköinen.

Kaikkiaan, tulokset tarkoittavat esimerkiksi, että lausuma ”tämä sovellus on käytettävyys-

CUE-1

- 4 ryhmää testasi Windows Task Timerin
- Löytyi 141 ongelmaa, joita vain 1 sellainen ongelma, jonka kaikki ryhmät löysivät

CUE-2

- 9 käytettävyysryhmää arvioi Microsoft Hotmailin
- Kaikkiaan löytyi 310 käytettävyysongelmaa
  Ei yhtään sellaista ongelmaa, jonka kaikki ryhmät olisivat löytäneet
  Vain 2 sellaista ongelmaa, jonka löysi vähintään 6 ryhmää
  75% ongelmista oli sellaisia, jotka löysi vain yksi ryhmä
  29 vakavista ongelmista sellaisia, jotka löysi vain yksi ryhmä

CUE-4

- 17 käytettävyysryhmää testasi hotellin varausjärjestelmän
- Kaikkiaan löytyi 340 virhettä
  Vain 9 virheistä sellaisia, jotka löysi enemmän kuin puolet ryhmistä
  205 (60%) virheistä sellaisia, jotka raportoi vain yksi ryhmä
  Näistä 61 kriittisiä tai vakavia

Kuva 1. Vertailevien käytettävyystestien (Comparative Usability Evaluation, CUE) tulosten perusteella testien tulokset riippuvat voimakkaasti testien tekijöistä.

testattu” ei välttämättä kerro kovinkaan paljon. Testattu sovellus on luultavasti parempi kuin jos ei olisi testattu ollenkaan. Mutta auki jää, onko testauksen perusteella löydetty ja korjattu oleelliset ongelmat; puhumattakaan, että onko sovellus riittävän käytettävä.

Mitä käytettävyystestien tekijä voi tehdä?

Miten käytettävyystestien tekijä voisi osaltaan varmistaa, että hänen tekemänsä testit ovat laadukkaita?

Perusasia tietenkin olisi varmistaa, että omaa hyvät perustiedot käytettävyyden teorioista, käytettävyyden suunnittelun (”käyttäjäkeskeisen suunnittelun”) prosesseista, eri käytettävyysmenetelmistä sekä käytettävyysohjeistoista ja –standardeista.

Tässä on kuitenkin haasteena se, että vaikka käytettävyyskoulutusta annetaan eri oppilaitoksissa, käytettävyysasiantuntijuudelle ei ole ei ole olemassa yleisesti hyväksyttyä kriteeristöä (ns. ”body of knowledge” on kehityksen alla, mutta ei vielä olemassa; ks. http://www.upassoc.org/upa_projects/body_of_knowledge/bok.html). Periaatteessa siis kuka tahansa voi väittää olevansa käytettävyysasiantuntija. Toisaalta kukaan ei oikeastaan voi väittää, että juuri hän on pätevä.

Toinen perusneuvo – joka myös esitetään CUE -tutkimusten suosituksissa – on se, että tulisi varmistaa ammattitaitoaan oppimalla toisilta käytettävyysasiantuntijoilta. Oma kokemukseni on, että se, missä ja keneltä käytettävyystestauksen on oppinut, vaikuttaa huomattavasti siihen, miten testauksia tekee. Alan julkaisujen seuraaminen sekä seminaareihin, luentoihin, työpajoihin ja konferensseihin osallistuminen olisi hyödyllistä - alan tutkimuksen ja julkaisujen tekemisestä puhumattakaan. Tai voi vaikka osallistua CUE -tyyppisiin tutkimuksiin, joissa pääsee vertaamaan omia käytäntöjään toisten käytäntöihin. Ammattimainen verkostoituminen yleensä on hyödyllistä; verkostojen kautta kun on mahdollista luontevaan ajatusten ja kokemustenvaihtoon. Koska Suomessa piirit ovat pienet, kansainväliset verkostot ovat tärkeitä.

Yksi mahdollisuus on myös käyttää ulkopuolista käytettävyysasiantuntijaa auditoimaan käytettävyystestauksen käytäntönsä. Tätä kautta on mahdollisuus saada seikkaperäinen ”toinen näkökulma” omiin käytäntöihinsä. Tällainen ulkopuolinen näkemys on joka tapauksessa luultavasti vähintään ajatuksia tuulettava.

Mitä käytettävyystestien asiakas voi tehdä?

Jos CUE -tutkimusten tuloksia tarkastellaan käytettävyystestauksen asiakkaan näkökulmasta, niin tilanne on tietenkin huolestuttava. On selvää, että jotkut testit ovat laadukkaampia kuin toiset. Mutta testien asiakkaalle voi olla haastavaa arvioida sitä, mikä testi on enemmän laadukas ja mikä vähemmän. Mistä tietää, että juuri minulle tehtävien testausten ja niiden tulosten laatu on hyvää?

Yksi peruslähtökohta käytettävyystestin tekijän valinnassa on testin tekijäehdokkaan käytettävyyshistorian läpikäynti. Tässä on huomattava, että kokemusta testien tekemisestä on vain osatotuus – voihan testauskäytännöt olla lähteneet väärille urille alusta lähtien. Erityisesti tulisi tarkastella sitä, mistä edellä oli puhetta: missä määrin tekijäkandidaatti on varmistanut ammattitaitoaan.

Toisena vaihtoehtona voi ajatella tarkistuslistaa, jonka avulla tarkistaa testin ammattimaisuutta:

- edustavatko testitehtävät reaalimaailman tilanteita?
- ovatko tehtävät muotoiltu siten, että ne eivät sisällä vihjeitä?
- sisältääkö testiraportti oleelliset kohdat
- erotellaanko tuloksissa selkeästi faktat ja johtopäätökset
- jne.

Tällaisten arviointien tekeminen on kuitenkin haasteellista käytettävyysalaa tuntemattomalle. Voi olla vaikea arvioida tekijän taustatietojen relevanttiutta. Käytettävyystestien muuttujia taas on paljon, eikä niiden perusteella tehty arviointi ole mekaanista.

Käytännössä asiakkaalla tulisi olla käytettävissä käytettävyysasiantuntijuutta, joka on riippumaton testien tekijästä. Tällainen asiantuntija voi auttaa joko suoraan arvioimaan käytettävyystestin ammattimaisuutta tai valitsemaan sopivaa käytettävyystestin tekijää.

Tilatako käytettävyystestejä ollenkaan?

On tietenkin tilanteita, joissa asiakas ei tilaa suoraan käytettävyystestausta, vaan esimerkiksi konsultointityyppistä palvelua siitä, mitä hänen kannattaisi käytettävyyden eteen yleensä tehdä (mistä aloittaa, mitä tehdä seuraavaksi jne.).

Tällaisessa tilanteessa eri käytettävyysasiantuntijoiden ehdottamat toimenpiteet luultavasti eroavat vielä enemmän kuin käytettävyystestauksen suhteen. Tämä yksinkertaisesti siitä syystä, että käytettävyystestaus on kuitenkin ehkä perinteisin ja eniten käytetty yksittäinen menetelmä. Jos eri asiantuntijoiden näkemykset ja käytännöt eroavat tällaisessa perusasiassa, niin käsitykset luultavasti eroavat vielä enemmän silloin, kun puhutaan muista käytettävyyssuunnittelun aktiviteeteista.

Käytettävyystestaus ei todellakaan välttämättä aina juuri se oikea toimenpide. Jokin toinen käytettävyysaktiviteetti voi olla kriittisempi, riippuen tilanteesta. Esimerkiksi strategisten ja operatiivisten käytettävyystavoitteiden määrittäminen (ks. http://www.joticon.fi/JFunnel.html) yleensä ottaen olisi erittäin hyödyllistä käytettävyystestien perustaksi.

Lopuksi

Kaikesta tässä artikkelissa esitetystä kriittisestä arvioinnista huolimatta käytettävyystestaus on ehdottoman hyödyllinen ja suositeltava menetelmä. Mutta tulee vain ymmärtää ja huomioida menetelmään liittyvät rajoitukset. Ja se, että käytettävyys varmistus ei ole ensi sijassa sen testausta.

Artikkelissa on korostettu, että käytettävyysasiantuntijoiden tulisi ottaa oppia toisilta. Niinpä itsekin otan mielelläni vastaan palautetta, jos artikkeli sellaisia herättää. Vaikkapa tuosta lopussa mainitusta käytettävyystavoitteiden määrittämisen hyödyllisyydestä.

You need to be smart!

by Ivar Jacobson

One of the most popular movements in software development in recent years is the move toward agility. Today everyone wants to be agile. That is good! However, the essence of being agile is being smart. I have for several years expressed that the most important character you need to have to be a great software developer is to be smart. In several of my columns I have summarized what you need to be successful by saying: You need to be smart! What does that mean? Most people know intuitively what "being smart" means in everyday language, but what does it mean for software.

It is not smart to model everything in UML for instance when building software. It is not smart to model nothing and go straight to code. It is however smart to find exactly that something that is of importance to model and code.

Advice: What is that something? It is about the most essential use cases and in particular about the most essential scenarios through these use cases. It is about the components and in particular the parts of those components that realize these essential scenarios. Thus, it is about the essentials. Now you may ask what makes a scenario essential. An essential scenario is the response to the question: "what are the most important scenarios the system is supposed to do". Which scenarios exercise the critical parts of the architecture? Which scenarios need to work in order for us to say that the highest technical risks have been eliminated?

It is not smart to write requirements without caring that these requirements are testable. It is smart to make sure the requirements also are test cases.

Advice: try use cases since they are also great test cases.

It is not smart to work in a waterfall manner, first specify all requirements, then do the design and the code, and finally test it all. If you do, you will discover serious problems with performance, architecture, usability too late. It is smart to first build a small skinny system and then build a little bit more, and a little bit more, before you release the system to customers. Each time you build something, you must be able to run, validate and test it.

Advice: use a controlled iterative approach such as the iteration practice in the Unified Process or sprints in scrum.

It is not smart to run off and build a lot of "stuff" before first assessing if you can source the whole or parts of the application (Open Source or commercial offerings).

It is not smart to develop software with a process that cannot scale if your system is successful and customers want much more. It is smart to use a way of working that is no more than what you really need, but that can grow as you succeed with the product.

Advice: make sure your process has a dial for each interesting process parameter so you can turn the dial to the proper position for your project.

It is not smart to throw out your existing process and adopt a completely new process. That is doomed to fail in most cases. Not everything you did in the past was wrong so why should you start all over. It is smart to improve your process in small manageable steps.

Advice: add one practice at the time.

It is not smart to find a shortcut that in reality becomes a detour, such as skipping requirements and going straight to code. It is smart to do enough requirements to find what to use to build your first increment, your skinny system.

In general it is not smart to be extreme in what you do such as: model everything or model nothing, follow a strict waterfall process or an unstructured iterative approach, throw out what you have and start all over. It is smart to be balanced to do what is needed right now but with an eye to the future.

Above I have given a number of examples. In each case, there are some ideas on how to think about being smart. And each case can be expanded further. You will become smart with experience, but experience on the other hand is not a guarantee for being smart. Of course eventually, it comes back to you.

Smart is not the same thing as being intelligent. You can be intelligent without being smart. And you can be very smart without being very intelligent.

Smart is not the same as having common sense. You can have common sense without being smart, but if you are smart you must have common sense.

Smart is to do exactly right, not to find a broad solution that is just about right.

Let us all become smarter.

Tohtori Ivar Jacobson on ohjelmistomenetelmien vaikutusvaltaisimpia hahmoja, joka on ollut mukana kehittämässä monia ohjelmistoalaa muuttaneita menetelmiä. Näitä ovat: komponentit ja komponenttiarkkitehtuurit, käyttötapaukset, aspektisuuntautunut ohjelmistokehitys, Unified Modelling Language (UML) ja Rational Unified Process (RUP). Viime aikoina hän on keskittynyt käytäntöihin perustuvaan ohjelmistotuotantoon, jonka suosio kasvaa nyt ympäri maailmaa.

Ivar on ketterien (Agile) menetelmien vahva puolestapuhuja ja kehittää niitä fiksummiksi. Hänen mottonsa on "Kaiken tulisi tulla ketterämmäksi, mutta ketterä ei ole kaikki". Ivar on kirjoittanut kuusi suosittua kirjaa ohjelmistotuotannosta. Ivarin yrityksellä, "Ivar jacobson International":lla on tomintaa Yhdysvalloissa, Iso-Britanniassa, Kiinassa, Singaporessa, Australiassa, Saksassa ja Ruotsissa.

Tästä Systeemityö-lehden numerosta alkaen Ivar kirjoittaa lehteemme kolumnia ohjelmistotuotannon muutoksen aallonharjalta, tervettä järkeä ja pitkää perspektiiviä unohtamatta.