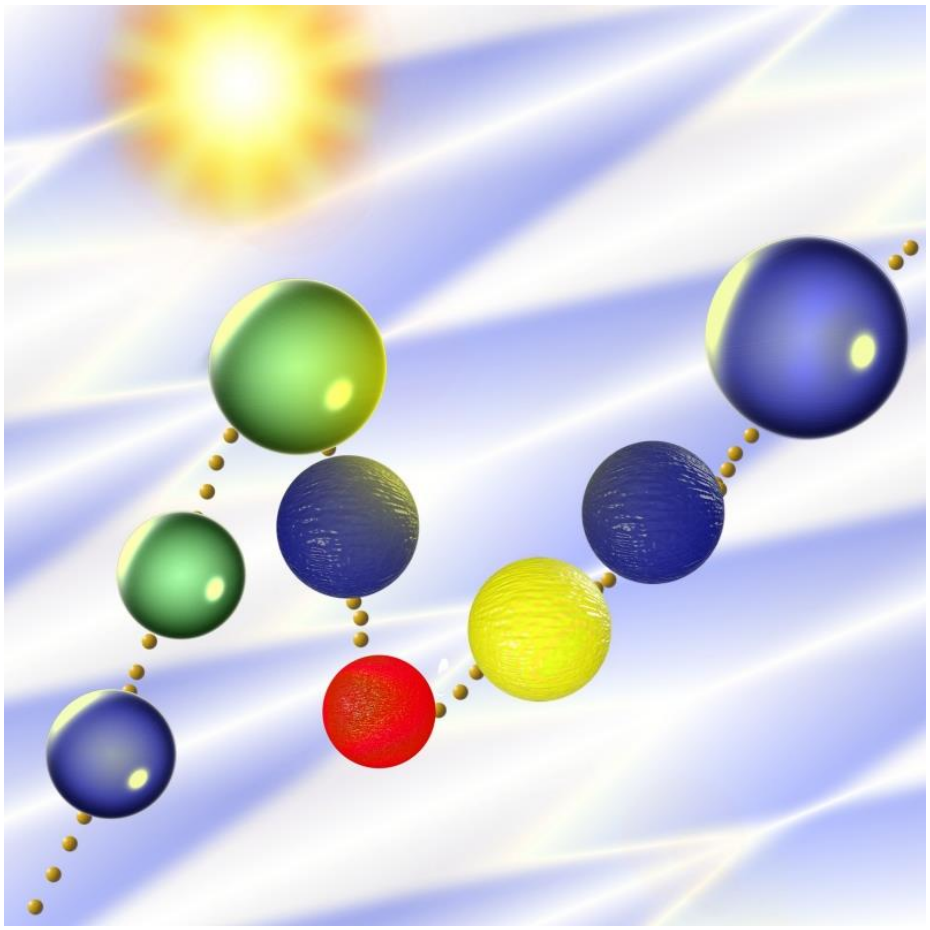
Laatu ja testaus

Uudet teknologiat ja laatu

1/2015



TestausOSY
Finnish Association of Software Testing

Julkaisija

Testauksen osaamisyhteisö TestausOSY

Toimituskunta

Matti Vuori, Soile Sainio, Tuula Pääkkönen

Julkaisupaikka

Verkko: http://www. testausosy. fi

Lehti on laadittu vapaaehtoisvoimin. Artikkelien oikeudet kuuluvat kirjoittajille.

Tekninen taittotyö: Matti Vuori

Kansikuva: Matti Vuori

Sisällysluettelo

3. Pääkirjoitus. Matti Vuori.

4. Suhteestamme uuden teknologian laatuun ja sen varmistamiseen. Matti Vuori

11. Big data ja testauksen analogiat. *Tuula Pääkkönen*

14. Uusi aika vaatii uudet kujeet. *Heikki Virtanen*

20. Laatu ketterässä yrityksessä. *Jukka Rapinoja*

21. Kiitos vai syytös. Kas siinä pulma*. Antti Niittyviita*

22. Luovuus testauksessa. *Jussi Rautio*

24. Täysautomaattisesti sovellukset testaava ja korjaava uusi ohjelmistoalusta. *Janne Nokiantausta*

Pääkirjoitus

Matti Vuori

Tässä numerossa on pääteemana uusien hype-teknologioiden laatu. Tavallisessa ja sosiaalisessa mediassa puhutaan paljon big datasta, tekoälystä, roboteista, esineiden ja teollisuuden Internetistä.

Tämä uusi teknologian maailma tuo meille valtavia mahdollisuuksia, mutta myös uusia haasteita teknologiaa soveltaville yrityksille. Tarvitaan uutta osaamista, tuotteet saattavat uudistua perus-konsepteiltaan ja uusi teknologia tuo uusia hallit-tavia riskejä, niin teknologisia kuin liiketoimintaan liittyviä – puhumattakaan vaikutuksista asiakkaisiin ja kuluttajiin.

Tässä numerossa on ydinteemana sen kaiken hallitseminen yrityksissä ja varsinkin niin tuote-kehityksessä ja liiketoiminnassa. Mitä kaikkea tämä tarkoittaa testaamiselle? Millaisia laadun ja sen "varmistamisen" haasteita syntyy uudenlaisten tuotteiden ja järjestelmien valmistajille ja integroijille ja miten niistä haas-teista selvitään?

...Ja sitten palataan realiteetteihin...

Tuo uusi teknologian maailma ei kerännyt kovinkaan montaa juttua, mikä kertoo siitä, että testausyhteisö elää enimmäkseen tutun olemassa olevan teknologian parissa ja teknologian pohdinta on jätetty muille. Positiivista siinä on keskittyminen asioihin, jotka ovat. Testauksen arvo on juuri siinä, että katsellaan asioita niinkuin ne ovat eikä haaskata aikaa visioihin ja skenaa-rioihin, jotka eivät kenties toteudukaan. Se ei kuitenkaan saa merkitä proaktiivisuuden puutetta, sillä hektisessä maailmassa on otettava hieman ennakkoa, että pysyy mukana kuvioissa.

Ja numerossa on pääteeman lisäksi juttuja muiltakin kiinnostavilta alueilta. Mielenkiintoisia lukuhetkiä!

Suomessa 1.4.2015

Suhteestamme uuden teknologian laatuun ja sen varmistamiseen

Matti Vuori, Tampereen teknillinen yliopisto, Tietotekniikan laitos

Maailmassa on joka hetki joitakin tärkeiksi tulevia uusia teknologioita tai tuotteiden piirteitä. Joskus ne ovat vanhojen asioiden uusia ilmentymiä, kuten esineiden Internet, joskus pitkään kypsyneitä paradigmoja, kuten tekoäly, joka on vähitellen kypsynyt ja on nyt mullistamassa monia tuoteryhmiä. Samaan aikaan toimintaympäristömme muuttuu. Kaikki tämä tuottaa haasteita tuotteiden ja järjestelmien testaamiseen ja siksi onkin aika luoda katsaus tähän kokonaisuuteen.

Teknologia toteuttaa unelmia

Uusi teknologia toteuttaa aina jotain yhteisiä haaveita. Kännykät yhdistävät ihmisiä, teollinen Internet tekee samaa koneille, robotit korvaavat ihmisen vaarallisissa töissä. Ja tietysti uusi teknologia on myös kuin myyttinen Sampo, joka tuottaa vaurautta soveltajilleen.

Akseli Gallen-Kallela: Sammon taonta, 1893

Niinpä teknologiaan suhtaudutaan lähtökohtaisesti positiivisesti ja mikään muu ei mahdollistaisikaan innovaatioiden syntymistä.

Joskus tästä seuraa sokeaakin hypetystä ja unohtuu riskien ja potentiaalisten ongelmien arviointi. Pahimmillaan ongelmat voivat olla tabu, mistä ei sovi puhua alan yhteisössä.

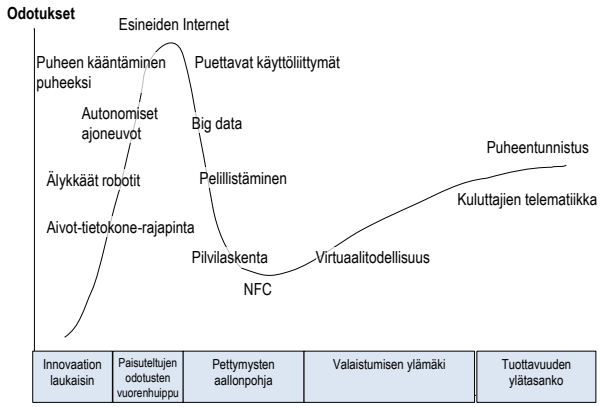
Kypsässä organisaatiossa on siksi hyvä olla erilaisia ihmisiä ja näkökulmia. Testausihmiset edustavat riskien näkökulmaa ja ovat dialoginen vastapainoa innovaattoreille ja tuotepäälliköille. Hyvä tulos syntyy ihmisten yhteispelillä. Unelmia pitää kuvata skenaarioilla ja tarinoilla (ja joidenkin ihmisten nimenomaan pitääkin intoilla!), mutta niiden vastapainoksi on tehtävä riskianalyysejä eri näkökulmista – uuden teknologian asiakkaat ja käyttäjät, teknologia, liiketoiminta. Riskejä voidaan paljastaa konkreettisemmin, kun teknologiaa toteuttavista tuotteista tehdään prototyyppejä ja niitä arvioidaan ja testataan. **Jos joskus on kuultu, että ”testaus on kuollut”, niin uusien teknologioiden tapauksessa on päivänselvää, että erityisosaavaa testausta tarvitaan paljon enemmän kuin aikaisemmin.** Yksi testauksen haaste on osoittaa, milloin uudesta konseptista ei olekaan markkinakelpoiseksi. Kun se tajutaan mahdollisimman aikaisin, säästyy valtavasti rahaa ja aikaa.

Koska teknologia sinänsä on äärimmäisen kiehtovaa, uusia teknologiaan perustuvia tuotteita kehitetään helposti teknologiavetoisesti ja unohtuu käytön ja kulttuurin näkökulma. Teknologialla on suuri itseisarvo! Mietitään, voisiko teknologiaa käyttää, eikä mikä olisi paras keino toteuttaa ihmisten tarpeita. Tässä ei ole mitään väärää, mutta se vaatii täydentäviä näkökulmia tuotekehityksen kuluessa – testauksen puolelta edes käytettävyystestausta.

Teknologialähtöisyys ei ole mitenkään hype-teknologioiden soveltamisen ominaispiirre. Jopa perinteisiä tietojärjestelmiä kuvataan joskus luokkakaavioilla sen sijaan, että kuvattaisiin liiketoimintaprosessien ja käyttäjien maailman muutoksia.

Teknologioiden kypsyminen

Klassinen esitys teknologioiden kypsymiselle on konsulttiyritys Gartnerin ”hype cycle”[1]. Alla on kaavio, johon on poimittu muutamia tämän lehden kannalta relevantteja asioita (käännökset kirjoittajan).



Hypekäyrä

Teknologiat kohtaavat siis kivikoita matkan varrella. Kaavio kuvaa globaalia tilannetta ja yksittäisistä teknologioiden sovelluksista voi tietysti tulla menestyksiä, mitä yleensä edistää niiden tekeminen tarkkaan mietittyyn käyttötarkoitukseen. Ja erilaiset testaukset ja arvioinnit auttavat sellaisten menestystarinoiden luomisessa.

Prototyypitys ja prototyyppien arviointi ja testaus on perinteinen tapa tutkia uutta teknologiatuotetta. Testausta voidaan tehdä pilottiasiakkailla tai loppukäyttäjillä. Olennaista on käyttökokemuksen testaus, sillä se ratkaisee paljon. Tuotteet tarvitsevat aina jonkin referenssin, mihin niitä verrataan. Vain siten voi ostaja ymmärtää tuotetta ja tehdä valintoja. Siksi on tärkeää tehdä vertailutestejä kilpailijatuotteisiin. Huomattakoon, että kilpailijan ei tarvitse toteuttaa samaa konseptia, kunhan liittyy samaan tarkoitukseen, jolloin voidaan arvioida tuotteen kykyä suoriutua jostain käyttäjän tavoitteesta – ja verrokin voivat siten olla asiakkaille mittatikkuna. Mikä sitten on verrokki markkinoilla, on osin valmistajan itse päätettävissä ja viestittävissä mainonnassa.

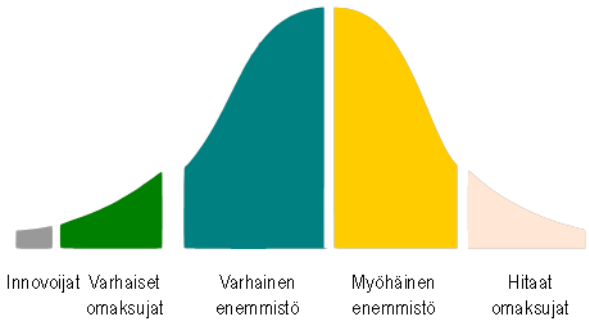
Nykyään on oivallettu, että yrityksen ei kannatakaan olla ensimmäinen markkinoilla uudessa tuoteryhmässä juuri sen takia, että ostajat tarvitsevat selkeän verrokin. Laadun kannalta tämä on hieno asia, sillä markkinoiden ensimmäinen auttaa kehittäjiä ymmärtämään konseptia ja toimii mittatikkuna vertailussa omaan tuotteeseen.

Innovaatioiden polku markkinoille

Teknologioiden odotetaan kulkevan markkinoille erilaisilla tavoilla teknologiaan suhtautuvien ryhmien kautta. Niiden klassinen kuvaus on Rogersin[2] jäsennys erilaisista innovaatioiden käyttöönottajista, joiden oletetaan omaksuvan asioita järjestyksessä.

Käännös alla poimittu Wikipediasta[3]:

- Innovoijat (innovators) – 2–3 % väestöstä: uskaliaita, koulutettuja, omaavat useita informaation lähteitä;
- Varhaiset omaksujat (early adopters) – 10–15 % väestöstä: sosiaalisia johtajia, suosittuja, koulutettuja;
- Varhainen enemmistö (early majority) – 30–35 % väestöstä: vastaanottavaisia tultuaan vakuuttuneiksi innovaation omaksumisen hyödyistä, useita eri sosiaalisia kontakteja;
- Myöhäinen enemmistö (late majority) – 30–35 % väestöstä: skeptisiä, perinteisiä, alempi sosio-ekonominen asema;
- Hitaat omaksujat (laggards) – 10–20 % väestöstä: vastustavat aktiivisesti uusia innovaatioita, naapurit ja ystävät pääasiallinen tiedonlähde, pelkäävät velkaantumista.



Innovaation omaksujat

[1] http://www.gartner.com/newsroom/id/2819918

[2] Rogers, Everett M. (2003) [1962]: Diffusion of Innovations. 5. edition. Free Press, New York

[3] http:/fi.wikipedia.org/wiki/Innovaatio

Teknologiasta innostuneille ei tietenkään ole traditionaalisella laadulla väliä. Olennaista on se, että uusi teknologia suunnilleen toimii, että voidaan yhdistyä sen lupaukseen ja saada osviittaa tulevasta. Tämäkin on ensimmäisen tuotetoteutuksen laatua, mutta luonteeltaan hyvin erilaista kuin se, mitä normaalisti tarkoitetaan laadulla – vähävirheistä, oikeaa toimintaa, käytettävyyttä, turvallisuutta jne.

Mutta kun tavoitellaan massamarkkinoita (varhainen enemmistä) laatuvaatimukset kasvavat radikaalisti. Teknologian on toimittava niin hyvin kuin tuotteet yleensä ja paremmin kuin kilpaileva vanha teknologia. On tehtävä muunnos unelmista ja estetiikasta arvoksi. Tässä vaiheessa tarvitaan hyvää toiminnallisuutta, luotettavuutta, käytettävyyttä ja turvallisuutta. Eli tarvitaan niiden suunnittelua ja toteutusta ja testausta, jolla tuotetaan niihin liittyvää tietoa ja voidaan selvittää, miten hyvin kehittämisessä on onnistuttu.

Tämä vaatimusten kasvu ei ole aina ollut itsestään selvää. On usein ajateltu, että kunhan polun alkuun pääsee, kaikki sujuu ”lähes itsestään kovalla työllä”. Moni yritys ajattelee, että kun kaksi tuttua avainasiakasta ovat tyytyväiset heille tehtyyn räätälöityyn sovellukseen, se kelpaa kaikille muillekin, kunhan vain tehdään siitä esite. Mutta pääsy varhaisen enemmistön suosikiksi on hyvin vaativaa. Geoffrey Moore on käsitellyt tätä teemaa kirjassaan Crossing the Chasm, josta ilmestyi vuonna 2013 kolmas painos. Se on hyvin suositeltavaa lukemista asiasta kiinnostuneille.

Tämä haaste on samankaltainen kuin startupeilla, jotka muutaman avainasiakkaan jälkeen haluavat skaalata tuoteliiketoimintansa suuremmalle asiakaskunnalle. [4] Silloin huomataan, että laadun ”varmistamiseen” tarvitaankin enemmän aktiviteettia ja osaamista. Haastetta korostaa se, että monella startupilla voi tuotteen ytimessä olla aivan uutta teknologiaa, joka on kenties vielä kypsymässä. Silloin pitää todella herätä asiaan ja uudistaa tuotekehitys ja tuotteiston hallinta vastaamaan yrityksen seuraavan vaiheen vaatimuksia.

Uusi teknologia eri rooleissa

Mahdollistaja

- Disruptiiviset tuotteet, jotka luovat avainpiirteiltään erilaisen tuoteryhmän ja uuden markkinan.
- Uudenlaiset toiminnot.
- Parempi suorituskyky, laatu.
- Uusien ekosysteemien luominen.
- Uudet yhteydet systeeminen välillä.
- Oman organisaation uudistaminen, uusi alku, henkilöstön herättäjä.

Rationalisoija

- Geneerisempi ja tehokkaampi tuotteiden kehitysalusta (esim. geneerinen käyttöjärjestelmä, Internet-teknologiat).
- Suuremman valmiusasteen komponentit.

Houkuttelija

- Asiakkaiden ja käyttäjien unelmien täyttymys.
- Markkinointiin uusi kärki.
- Signaali markkinoille ja investoijille.

Markkinapakko

- Pakollinen asiakasvaatimus.
- Kilpailijatkin soveltavat.

Kaivataan siis monenlaista uudistusta:

- Tuotteen seuraavan vaiheen menestystekijöiden tunnistaminen.
- Kompetenssien ja toimintamallien arviointi.
- Testauksen ja muun ”laadunvarmistuksen” kohteiden laajentaminen ja fokuspisteiden uudelleenarviointi.
- Henkilöstön virittäminen uusiin tarpeisiin. [5] Johtajuutta, viestintää, teknologiakoulutusta, erityistehtäviä.

Kaikki tällainen ajattelumallien muuttaminen on vaikeaa.

[4] Startupin haasteista ks. ” Software Startup Patterns - An Empirical Study” http://dspace.cc.tut.fi/dpub/handle/123456789/22067

[5] Olin kirjoittamassa ”uusiin haasteisiin”, mutta kulttuurissamme se tarkoittaa uuden työpaikan etsimistä...

Fokusoinnille on erityistä tarvetta kolmessa eri tilanteessa:

1. Tuotteen konseptia ja piirteitä etsittäessä.
2. Muutostilanteessa, kun suunnitellaan tuotteen tai tuotteiston laajentamiseen.
3. Suuremman volyymien tilanteessa – enemmän asiakkaita, enemmän toimituksia, enemmän ominaisuuksia.

Iso osa fokusointia on sen ymmärtäminen, mitä EI kannata tehdä. Kun vasta haetaan konseptia tai pyritään saamaan ensimmäinen tuote ulos, ei sitä työtä todellakaan nopeuta jatkuvan käyttöönoton putken automaattisen testausinfrastruktuurin rakentaminen. Sen sijaan sitä nopeuttaa älykäs testaus, jolla selvitetään, mitä tuotteessa pitäisi olla ja miten tuotteesta ja sen liiketoiminnasta saadaan skaalautuva. Testiautomaation aika on vasta myöhemmin.

Insinöörityötä ja tuotekehitystä

Teknologiatuotteiden kehittäminen on yleensä implementoivaa insinöörityötä. Sille työlle on tyypillistä fokuksen kohdentuminen matalan tason teknologiaan ja testauksen tekeminen minimivaivalla. Uusien hype-teknologioiden kypsyttämisessä tarvitaan matalan tason fokusta, jotta järjestelmistä saadaan robusteja, mutta keskittyminen siihen merkitsee vähäisempää keskittymistä muihin asioihin. Jo alussa mainitsimme käyttäjien maailman, mutta yleisemmin on kyse tuotetason suunnittelusta, parhaiden konseptien löytämisestä. Silloin nähdään paremmin myös tarvittavat testattavat asiat. Tätä muutosta edistää ketterän kehittämisen kulttuuriin kuuluva heterogeenisten tiimien ajatus. Kun tiimissä ei ole vain viittä samanlaista koodaajaa, vaan myös ihmisiä, jotka pohtivat tietoturvallisuutta ja käyttäjäkokemusta, nämä ”uudet” asiat saadaan helpommin työlistalle. Ei sorruta niin helposti siihen, että markkinoiden mullistajaksi suunnitellun uuden teknologian tuotteen käytettävyys ”varmistetaan” tekemällä liian myöhään käytettävyystestaus (jonka perusteella ei enää ehdi tehdä parannuksia), mutta ei mitään muuta.

Toinen tärkeä ajattelutavan muutos – mikä ei siis ole vain retoriikkaa – on Minimum Viable Product -ajattelu yhdessä Lean Startup -menetelmän kanssa[6]. Perusidea on tehdä ”pienin tolkullinen tuote”, jota sitten testataan ja siten opitaan ymmärtämään, mitä asiakkaat ja käyttäjät odottavat ja millainen tuote voisi olla hyvä. Tämä on oleellista, sillä uuden teknologian disruptiivisia tuotteita ei koskaan ymmärretä riittävästi ja ilman ymmärryksen luomista kehitystyö menee metsään. Oikeastaan on väärin puhua tuotteiden ymmärtämisestä, sillä se mitä kaivataan on asiakkaiden ymmärtäminen: mitä he tarvitsevat ja haluavat ja miten luodaan tuote joka vastaa tarpeita ja haluja. Siksi yksi tuotekehityksen uusista käsitteistä onkin ”customer development”[7].

Tuotekehityksen tyylien tasoja

Jatkokehitystä tehdään variaatioiden ja niiden testaamisen avulla. Tämä kuulostaa perinteiseltä prototyypitykseltä, ja jossain määrin onkin sitä uudelleen paketoituna – jos puhutaan hyvin määritellystä ”tieteellisestä” prototyypityksestä, mitä yritysten arki ei aina ole. Prosessia ohjaamassa on selkeitä hypoteesejä ja näkemys siitä, mistä asioista halutaan saada lisää ymmärrystä. Testaukselle tämä tuottaa monia uusia mahdollisuuksia ja osaamishaasteita.

[6] Ks. Erik Ries. 2011. Lean Startup ja UX-kehittämiseen liittyen Jeff Gothelf & Josh Seiden. 2013. Lean UX.
[7] Ks. tiiviissä muodossa Brant Cooperin ja Patrick Vlaskovitsin kirja The Enrepreneur’s Guide to Customer Development – A ”cheat sheet” to The Four Steps to the Epiphany (joka on paksu Gary Blankin kirja).

Teknologia monimutkaistaa maailmaa

Teknologiset innovaatio harvemmin korvaavat vanhaa teknologiaa, vaan tulevat sen lisäksi. Tuloksena on seuraavanlaisia vaikutuksia:

- Tuotteet ja järjestelmät monimutkaistuvat.
- Teknologioiden ja eri järjestelmien vuorovaikutukset kasvavat.
- Tuotteiden kehittäjät ovat koko ajan oppimiskäyrällä.
- Uusi teknologia ei ole kypsää ja siksi uudet tuotteet ovat epäkypsiä.
- Olemassa oleviin tuotteisiin tulee uusia piirteitä uuden teknologian myötä.
- Joskus uuden teknologian avulla uudistetaan tuoteparadigmoja ja koko organisaation on uudistettava ajatuksiaan. Esimerkiksi koneenrakentajat voivat löytää olemuksensa metalliosien kokoonpanijan sijaan logistisen järjestelmän tekijästä.
- Joskus uusien teknologioiden tuottamat tuotteet ovat niin erilaisia, että suhdetta niihin on vaikea määrittää. Onko esimerkiksi älykäs ihmisenkaltainen robotti ”väline”, vai sosiaalinen toimija.

Joskus usea hype-teknologia esiintyy yhdessä ja samassa paketissa. Esimerkiksi älykkäissä roboteissa yhdistyvät uusi anturitekniikka, vuorovaikutusteknologiat, tekoäly ja tiedonhallinnan uudet tuulet. Sellainen kokonaisuus on erityisen haastava.

Joskus harvoin hype-teknologiat yksinkertaistavat maailmaa. Funktionaaliset ohjelmointikielet ovat olleet pitkään tulossa ja nyt ne näyttäisivät tulevan hieman lähemmäksi valtavirtaa. Niiden avulla pitäisi esimerkiksi rikkaan datan käsittelyn olla aiempaa helpompaa tietojärjestelmissä.

Haasteita testaukselle

Tämä kaikki aiheuttaa monenlaisia haasteita testaukselle. Tässä niistä muutamia:

- Älykkäiden järjestelmien testaus edellyttää testauksen lähtökohtien muutosta. Testattavaa kohdetta ei voida nähdä deterministisenä automaattina, vaan älykkäänä subjektina, jonka käyttäytymisen selvittäminen edellyttää psykologisia ja sosiologisia taitoja. Kuitenkaan testaaja ei saa kunnioittaa sellaisia kohteita samoin kuin älykkäitä ihmisiä, vaan edelleen pitää pyrkiä ”rikkomaan” kohteet eli löytämään niiden käyttäytymisessä olevat virheet ja ongelmat.
- Kun järjestelmien välinen vuorovaikutus lisääntyy vaikkapa kodeissa, tulee tarvetta tehdä testausta kokonaisjärjestelmälle. Muuten ei vuorovaikutuksen ongelmia löydy. Tämä on haaste testiympäristöille – miten rakennetaan ympäristö, jossa on kaikki tarvittava muu laitteisto. Samoin se on haaste testiautomaatiolle – miten kyetään rakentamaan sellainen testausjärjestelmä, joka pystyy ”rääkkäämään” heterogeenistä kokonaisuutta?
- Kun alustojen ja teknologioiden diversiteetti ja järjestelmien kokoonpanon dynaamisuus lisääntyy, ne ovat herkempiä ongelmille. Elementeistä pitää tehdä nykyistä robustimpia, minkä vuoksi jo yksittäisten elementtien testauksen on oltava perinpohjaisempaa. Se nostaa työmäärää ja kuluja kaikilla testaustasoilla. Mutta sama vaatimus koskee kokonaisjärjestelmiä. Jos mikroaaltouunin dataliikenne sekoaa päivityksen myötä tai se alkaa tehdä palvelunestohyökkäyksiä kodin järjestelmiin, tilanne on voitava hallita.
- Jos kaikki tavarat ovat kytkettyjä Internetiin joko suoraan tai keskuslaitteiden kautta, tietoturvallisuuden merkitys on valtava. Kräkkeri voi halvaannuttaa yksilön, perheen tai kodin koko elämän. Tietoriskejä pitää analysoida jokaisen tuotteen osalta, tutkia ratkaisuja ja testata niiden turvallisuus. Ja päivittää tilanne, kun systeemin muuttuvat. Riskit koskevat sekä selvästi luottamuksellista dataa, mutta kaikkea muutakin – kodin laitteiden käyttöprofiilista on rikollisten mukava selvittää perheen elintapoja.

Olennaisiksi testauksen periaatteiksi voisi muotoilla:

- Riskipohjainen asenne. Ollaan tekemisissä uuden asian kanssa, johon liittyy monenlaisia riskejä niin liiketoiminnan kuin teknologian tasolla. On tunnistettava ne, jotta niitä voidaan hallita.
- Uusi teknologia sitoo aina ihmisiä enemmän teknologiaan. Siksi käyttäjien ja asiakkaiden riskien analysointi on yhtä tärkeää kuin valmistajan riskien analysointi. Mitä tapahtuu, jos teknologia ei toimikaan?

- Uuden teknologian toimintaa ei ymmärretä. Siksi sitä pitää selvittää avoimin mielin tutkivalla testauksella.
- Koska erilaisia vuorovaikutusilmiöitä ja ympäristön muun tekniikan laatua ei tunneta, on jokaisesta komponentista testauksen avulla tehtävä mahdollisimman robusti.
- Kaikkeen uuteen teknologiaan liittyy uusia tietoturvahaasteita. Haasteita on analysoitava ja ratkaisuja tutkittava ja testattava.
- Uusi tekniikka on dynaamisempaa. Tarvitaan enemmän elinkaariajattelua testaukseenkin. Asennuksen ja käyttöönoton lisäksi on testattava omien tuotteiden päivitykset ja reagointi muutoksiin ympäröivän teknologian muutoksissa.
- Kaikkiaan on hyvä olla sopivan vainoharhainen…
- Uudenlaisten tuotteiden menestys on usein kiinni käyttäjäkokemuksesta. Siksi sitä on suunnittelun ohella tärkeää testata hyvin.

Entäpä testauksen hypet?

Testaus on myös täynnä uudenlaisia hypejä ja jos yritys on profiloitunut teknologian hyödyntämisen etulinjaan, on suuri psykologinen paine myös soveltaa uusimpia testausteknologioita. Siinä tietysti mennään helposti metsään. Esimerkiksi testiautomaatiota tärkeämpää uusia konsepteja arvioitaessa on käyttökokemuksen testaus ja tutkiva testaus, joiden avulla voidaan ymmärtää uutta tuotetta, teknologiaa ja sen käyttäytymistä. Automaatioon panostaminen tulee relevantiksi vasta hieman myöhemmin. Samoin nettisysteemin pienten variaatioiden A/B-testaus mittaamalla on relevanttia vasta sitten, kun palvelun kokonaiskonsepti on oikein suunniteltu. Ei kannata optimoida jotain, mikä ei peruspiirteitään vastaa parhaiten asiakkaiden ja käyttäjien tarpeita. Pienet resurssit pitää aina kohdentaa sinne, missä niitä eniten tarvitaan. Tällä periaatteella ei luoda prosessivelkaa, vaan juuri sellainen tapa toimia, jota tarvitaan kussakin tilanteessa.

Ymmärryksen luominen

Uuden teknologian käyttöönotto edellyttää yhteisen ymmärryksen luomista, jonka pohjalta voidaan tehdä päätöksiä, ohjata tuotekonseptien kehittämistä ja niiden laatuun liittyviä toimia.

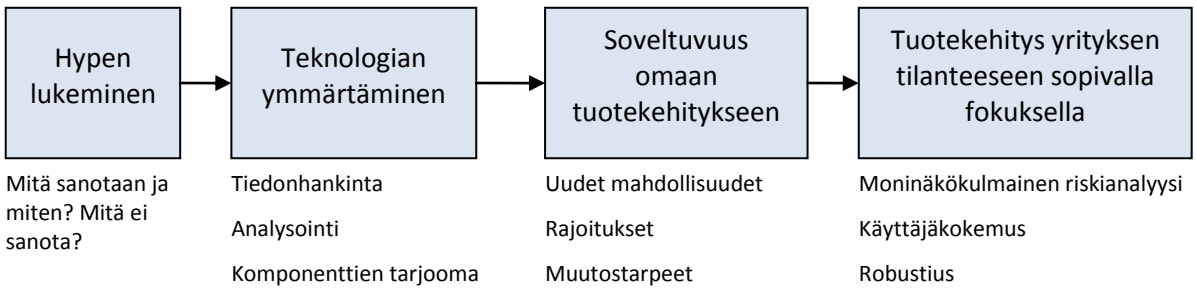
Oma analyysi ja ajattelu on tärkeää, sillä tietenkin uuden teknologian markkinoijat esittävät vain sen positiivisia puolia (*Coke add life* vs. *Coke rots your teeth and makes you fat!*).

Hypen lukutaito onkin tärkeää. On hyvä osata lukea markkinaviestien rivien välistä. Muutamia avainsanoja:

- Uusi = Kypsä vasta jonkin ajan kuluttua, käyttäjäkunnalle vieras. Ei sovi kulttuuriin.
- Monipuolinen = Kompleksinen. Vaikea.
- Älykäs = Sisältää tietoturvariskejä. Vaikea.
- Integroitu = Kokonaisuuden toiminta on haastavaa.
- Nopeampi = Kalliimpi. Vie enemmän virtaa akkulaitteista.
- [Brändi] = Sidottu tiettyyn ekosysteemiin ja suljettuun teknologiaan. Ei toimi yhteen oman puhelimesi kanssa.

Tässä vaiheessa on syytä muistuttaa, että uuden teknologian tuotteissa on kolme tärkeää tasoa:

1. Uusi teknologia abstraktilla tasolla, unohtaen vielä sen implementaatioiden



Vaiheita uuden teknologian soveltamiseen ryhdyttäessä

laadut. Tällä tasossa näkyvät sen yleiset piirteet, tärkeys, houkuttelevuus, riskit ja yleinen tilanne markkinoilla nyt ja jatkossa.
2. Tarjolla olevat komponentit ja implementaatiot. Niiden kypsyys, saatavuus ja sovellettavuus omissa tuotteissa.
3. Omat tuotekonseptit ja tuotteet, jotka soveltavat uutta teknologiaa.

Näitä kaikkia tasoja kannattaa arvioida systemaattisesti. Tämä arviointityö yhdistää kaikkia tuotteen parissa toimivia. Esimerkkejä:

- Teknologioiden kartoittaminen. Mikä on potentiaalisten teknologioiden tilanne? Miten ne kypsyvät? Millaisia sovelluksia on tulossa saataville teknologiatoimittajilta? Ovatko väitteet eduista luotettavia? Missä tilanteissa ne pätevät? Millaisia ongelmia on tiedossa? Onko asiantuntijoiden piirissä vaihtoehtoisia mielipiteitä?
- Soveltuvuuden analysointi. Miten teknologia soveltuu omaan tuotepalettiin? Mitä kaikkea pitää uudistaa sen soveltamiseksi? Tuottaako se lisäarvoa vai monimutkaisuutta? Riittääkö oma osaaminen vai tarvitaanko apua? Sopivatko sen lisenssit? Vastaavatko edut lisäkustannuksia? Onko tiedossa ristiriitoja avoimen lähdekoodin lisenssien kanssa? Jne.
- Teknologian riskianalyysi käytännön näkökulmasta osana omaa tuotekonseptia. Ottaen huomioon sen ymmärtäminen, kypsyys, luotettavuus, haluttavuus asiakaskunnassa, turvallisuus, skaalautuvuus, tuotteiden elinkaaren eri vaiheet jne.
- Tarjolla olevien implementaatioiden kartoitus ja arviointi. Ostetaan vaikka sitä soveltavia tuotteita ja testataan niitä.
- Kokeiluprojekti, jossa testataan käytännössä teknologian toimivuutta omassa teknologiapaletissa ennen sen viemistä tuotteisiin.
- Ja sitten järkevä tuotekehitys, jossa teknologialla ei ole itseisarvoa, vaan vain tuotteiden kokonaislaadulla. Nyt tarvitaan kaikenlaisten ominaisuuksien testausta ja arviointia. Olennaista ja usein unohdettua on vertailu tuotteen verrokkeihin markkinoilla, sillä laatu on aina suhteellista ja hankintapäätökset ovat valintoja vaihtoehtojen välillä.
- Käyttäjäkokemuksen analysointi ja testaus on tässä vaiheessa keskeistä. Tuotteen teknisessä analysoinnissa on hyvä tutkia arkkitehtuuria ja teknologiavalintoja, jotta ymmärretään tuotteen skaalautuvuus joko erilaisiin variaatioihin tai toiminnallisuuden suurempaan määrään.

Lopuksi…

Teknologia on aina kiehtovaa, mutta siinä juuri siksi vaaransa. On hyvä ymmärtää teknologian luonne ja haasteet ja millaisia toimia sen hyödyntäminen edellyttää kussakin yrityksen ja tuotteiden elinkaaren vaiheessa. Uuden teknologian tuotteiden kehittäminen äärimmäistää monia testauksen perusperiaatteita ja samalla korostaa erilaisia testauksessa tarvittavia kompetensseja. Riskit pitää ottaa todella vakavasti. Erittäin kriittinen asenne on tarpeen. Käyttäjäkokemuksen selvittämiseen on todella panostettava. Tarvitaan toimintaa korkealla koodin nippelien ja yksikkötestauksen tason yläpuolella. Mutta samaan aikaan on ”konepellin alle” luotava robustiutta. Laadukkaiden uuteen teknologiaan perustuvien tuotteiden aikaansaaminen on haastavaa ja siinä tarvitaan ”laadunvarmistuksen” panostusten kohteiden tarkkaa harkintaa, jossa on otettava huomioon myös yrityksen ja sen tuotteiston elinkaaren nykyinen ja seuraavat vaiheet – millaisia kohderyhmiä kulloinkin tavoitellaan? Hyvä uutinen on se, että haasteet vaihtelevat tilanteen mukaan ja järkevästi toimiva organisaatio ei ylikuormitu halutessaan tehdä asiat hyvin.

Erilaiset näkökulmat korostavat erilaisten ihmisten tarvetta tunnistamaan asioita, toimimaan niiden eteen ja käymään dialogia valinnoista. Ja ennen kaikkea yhteistyötä.

Matti Vuori on toiminut laatuun, testaukseen ja tuotekehitykseen liittyvien asioiden parissa pitkään ja aikoinaan tutkinut mm. tulevaisuuden tuotteiden kehittämistä ja tuotekehityksen riskienhallintaa ja konsulttina tehnyt teknologiaselvityksiä. Nykyisin hän on töissä Tampereen teknillisellä yliopistolla.

Jutun englanninkielinen käännös on osoitteessa http://www. mattivuori. net/julkaisuluettelo/liitteet/vuori_relation_to_quality_of_new_technology. pdf

Big data ja testauksen analogiat

Tuula Pääkkönen, Kansalliskirjasto

So wow, much data, big data. Mitä on big data aineisto ja kuinka sitä käsitellään? Tässä artikkelissa pohditaan erään big dataksikin sanotun aineiston käsittelyä ja kuinka tuttuja testauskäsitteitä tässä yhteydessä pääseekin käyttämään. Onko kaikki maailmassa vain testausta ja big data vain uusi kohde?

1. Taustaa

Mistä tietää, että tietty termi on hypeä? Se ilmestyy joka paikkaan, ja ei ole asiaa johon hype-termiä ei liitettäisi. Mistä tietää, että hype-termi on vakiintumassa? Siitä, että termi määritellään sanakirjoihin ja yhteinen ymmärrys soveltuvista käyttökohteista alkaa muodostua.

Esimerkiksi virallinen termi big datalle on tietotekniikan termitalkoissa määriteltiin vuoden 2013 lopulla ”massadataksi”, vaikka toki big data termiä edelleenkin enemmän käytetään. Siellä big data määritellään:

”data, jota on paljon, jota tulee nopeasti lisää ja joka on muodoltaan vaihtelevaa” [1]

Wanha testaaja sanoisi tässä kohtaa, että kuinka paljon on paljon ja kuinka nopeasti sitä tulee lisää ja mitä ne vaihtelevat muodotkaan ovat? Onneksi huomautuksessa jatketaan vielä:

Massadatan käsittely on vaikeaa perinteisillä tietojenkäsittelysovelluksilla. Laitteilta ja sovelluksilta vaaditaan valtavaa tallennuskapasiteettia ja suorituskykyä, jotta ne pystyisivät käsittelemään massadataa. [1]

Tämä on helppo allekirjoittaa. Yleensähän esimerkiksi erilaisten sensorien keräämä data on big dataa, koska sitä voi kertyä useita gigoja tunneissa.

Käyttämäni big data -esimerkki on suomalainen historiallisten sanomalehtien arkisto, jossa on kaikkiaan yhdeksän miljoonaa sivua ja tuoreesti laskettuna noin 5,2 miljardia sanaa. Ehkä kyse ei ole isoimmasta mahdollisesta aineistoista, mutta Suomen tasolla kuitenkin suhteellisen laajasta aineistosta ja näyttää jo suuntaa mitä big datasta on odotettavissa.

— Muistiinpanoja Haminan seuduilla.

Ensimäiset muuttolinnut nähtiin:	Kuu ja päivä vuosina 1885	1886	1887	1888	1889	1890
Leivo	13/4	31/3	3/4	15/4	5/4	25/3
Kottarainen	29/3	31/3	6/4	16/4	4/4	22/3
Peippo	1/4	5/4	11/4	1/5	14/4	2/4
Laulurastas	—	11/4	20/4	—	18/4	—
Räkätti rastas	20/4	12/4	22/4	—	20/4	—
Joutsen	1/4	8/4	7/4	20/4	2/4	3/4
Kurki	25/4	10/4	18/4	21/4	15/4	10/4
Hanhi	17/4	9/4	16/4	3/4	12/4	1/4
Västäräkki	15/4	11/4	25/4	18/5	15/4	12/4
Sinisorsa	12/4	6/4	5/4	19/4	15/4	2/4
Sotka	25/4	4/4	21/4	—	18/4	1/4
Isokuovi	22/4	15/4	18/4	—	10/4	14/4
Kivitasku	26/4	12/4	27/4	18/4	17/4	23/4
Tavi	1/5	11/4	19/4	—	26/4	13/4
Alli	1/5	25/4	26/4	—	—	14/4
Käki	6/5	13/5	4/5	6/5	8/5	5/4

Kuva 1. Lähde: digi. kansalliskirjasto. fi [3]

Sanomalehtiaineistoon pureuduttiin viime vuonna ja siinä projektissa alkoivat tutut testausanalogiat moneltakin kannalta esille. Sekä itse projektin, aineistojen ja käytettyjen menetelmien osalta.

2. Tutkimusta big datasta

Tutkimusprojektin lähtökohtana oli siis käytettävissä olevat digitoidut aineistot, josta ensimmäinen pohdittava asia oli, että minkälaista aineisto onkaan ja mikä sen laatutaso kokonaisuudessaan on. Tästä päästäänkin jo testaajan peruslähtökohtaan, että **oma data on tunnettava** ja tiedettävä mihin aineisto kykenee.

Big datan osalta ongelmat tulevat heti skaalautuvuudesta – jos esimerkiksi avainsanaohjatun automatisoinnin lähtöaineistona on suuri nivaska testidataa esimerkiksi Exceleissä, niin big datassa tuo

nivaska onkin tuhansia, ellei satoja tuhansia aineistoja. Joten missä menee testiaineiston ja varsinaisen aineiston raja? Big data aineistoakin testatessa täytyy lähteä pienistä otoksista ja varmentaa, että menetelmät toimivat niillä ennen kuin mennään täydellisiin koko aineistoja kattaviin ajoihin.

3. Big datan laatu

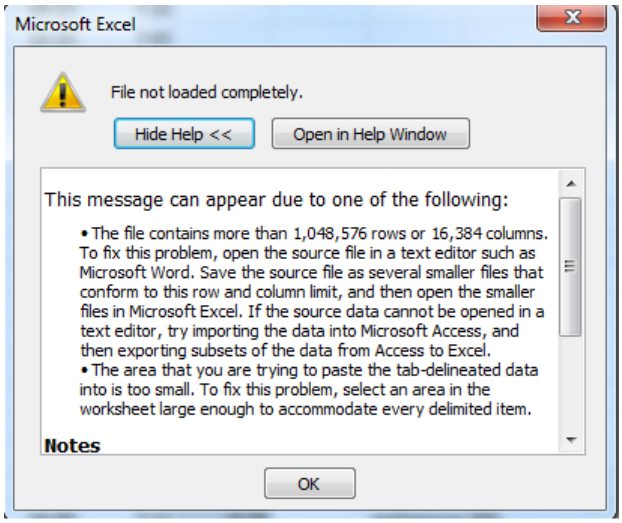
Yksi tapa big data -aineiston laadun arviointiin on vertailu, esimerkiksi meidän tapauksessamme nykyisen tekstiaineiston ja odotetun, ns. puhtaaksikirjoitetun tekstiaineiston välillä. Näin saadaan selkeä numeroarvio, jolla nähdään virheellisten sanojen määrät, [2]. (Puhdasta, korjattua aineistoa on kuitenkin vain vähän saatavilla, joten hyödyntämällä kaikki mikä löytyy voidaan luoda arvioita aineistolaadusta. Tässäkin on itsessään monta vaihetta – jo esitöissä, että ympäristö on soveltuva arviointiin ja aineisto käsittelykunnossa on omat työvaiheensa, on kuitenkin oleellista, että aineistoa ei muuteta sisällöllisesti prosessoinnin aikana vaikka formaatti analysointien takia muuttuisikin.

Toinen laadun tae on myös aineiston luomisen prosessissa. Kun aineiston käsittelyhistoria talletetaan, voidaan luoda jopa auditoitavissa oleva historia, josta näkee kaikki työvaiheet, joiden läpi aineisto on luontinssa jälkeen kulkenut.

4. Big data asenne

Omasta lyhyestä matkasta big datan syövereihin on tähän mennessä oppinut sen, että aineistomääriin pitää suhtautua vakavasti. Pitää muistaa skaalautuvuus koko ajan ja se, että pitää ajatella kaikki suurella mittakaavalla, jotta ajatus ei karahda kiville. Tai jos karahtaa, sitten on parempi saada kokeilu tehtyä ja epäonnistua nopeasti (fail fast), jotta oppii ja voi jatkaa paremmalla strategialla eteenpäin.

Siinä kohtaa, kun Excelissä eivät rivit riitä tai levytila loppuu väliaikaistiedostoilta tai törmää tunnetun ohjelmiston perusversion bugiin suurten tiedostojen kanssa oikeasti alkaa ymmärtää millaisen aineiston kanssa on tekemisissä.



Kuva 2 Ai, niin. Oops.

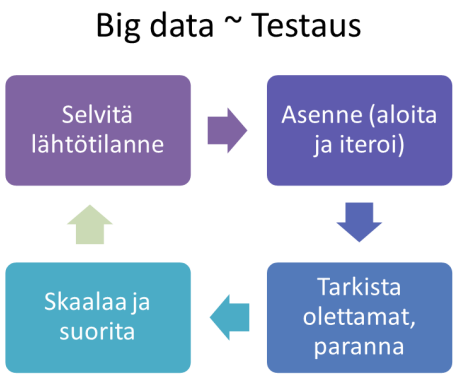
Toisaalta aineistoihin oppii suhtautumaan omana tyyppinään, omanlaisenaan kokonaisuutena. Lisäksi oppii hyväksymään sen, että aineiston käsittely ja prosessointi vain vie aikaa ja konetehoja ja esityöhön kannattaa kunnolla satsata, jotta jälkityöt aineiston kanssa helpottuvat. Suunnittelua ja voimavarojen varaamista oikeaan aikaan siis vaaditaan, kuten testauksessakin… Lisäksi koko ajan täytyy olla tuntuma siihen mitä tavoittelee, jotta aineiston syövereihin ei eksy. Tärkein asia ensin ja siitä järjestelmällisesti eteenpäin pohditun strategian mukaan.

Prototyypit, kokeilut auttavat myös – yleisillä malleilla saadaan ”hahmotettua” aineistoa paremmin, jolloin voidaan erikoistapaukset huomata ja käsitellä ennen kuin nämä vaikuttavat koko aineiston käsittelyyn.

5. Lopuksi

Big dataa on kaikkialla. Tutkijoista tai testaajista löytyy samanlaisia arvostettuja piirteitä: yksityiskohtiin tarttuvan, kyseenalaistavan ja uteliaan asenteen, jossa datamäärää pelkäämättä tartutaan toimeen ja varsinaiseen tekemiseen.

Yhteenvetona listaisin kuvassa 3 olevat periaatteet sekä big data- että testauspuolelta: 1. selvitä lähtötilanne, 2. asenne, aloita tutkimaan aineistoa ja toista eri variaatioilla, 3. opi löydöksistä ja paranna tekemistä ja 4. lopuksi skaalaa ja jatka kunnes tavoitteet saavutettu (Demingin laatuympyrää vapaasti hyödyntäen.)



Kuva 3. Samanlaiset syklit.

Näinä digitaalisuuden aikoina dataa tulee koko ajan lisää ja lisää, joten big datan käsittelytarpeet voivat tulla entistä enemmän arkeen ja ehkä entistä moninaisemmilla tavoilla. Toivottavasti vain koneiden prosessointiteho kasvaa ja levytilaa on jatkossakin runsaasti käytettävissä, koska näitä tarvitaan. Erityisesti sitten kun päästään vaiheeseen, jossa päästään yhdistämään erilaisia aineistomassoja toisiinsa luoden uusia, entistä laajempia kokonaisuuksia.

P. S. Tässä yhteydessä kiitokset kuuluvat eräille tietyille avointen lähdekoodien ohjelmistojen tekijöille, joilta on saanut koodikorjauksia, beta-versioita ja vastauksia silloin, kun niitä on tarvittu. Kiitokset näkymättömästä ylläpitotyöstä, jota ei aina tule ajatelleeksi.

Lähteet:

[1] Tietotekniikan termitalkoot, http://www. tsk. fi/tsk/termitalkoot/fi/node/266
[2] Word error rate http://en. wikipedia. org/wiki/Word_error_rate
[3] Muistiinpanoja Haminan seuduilla, Kansalliskirjaston digitoidut aineistot, digi. kansalliskirjasto. fi, Haminan Sanomat, 13. 03. 1891 no 20 s. 2

Tuula Pääkkönen on toiminut ohjelmistokehityksen parissa yli 17 vuotta. Hän aloitti ohjelmistokehittäjänä, päätyi testauksen automatisoinnin kautta testaushallintatyökalujen pariin ja testausprosessien kehittämiseen. Nykyään hän on töissä tietojärjestelmäasiantuntija Kansalliskirjastossa.

Uusi aika vaatii uudet kujeet

Heikki Virtanen

Maailma on jo nyt vuorovaikutteinen. Tietokoneet ovat verkottuneet seitiksi, jota erimuotoiset pilvenkuukaleet täplittävät. Digitalisaation myötä vuorovaikutteisuus vain lisääntyy. Älykäs elektroniikka valtaa tehtaat, työkoneet, kulkuneuvot, ihmisten asunnot, korut, jopa vaatteet, ja liittää ne osaksi suurta kudelmaa. Toivoa sopii, että kaikki nämä iholle pyrkivät ja hypen säihkyväksi kiillottamat hilavitkuttimet toimivat kunnolla ja tulevat juttuun keskenään. Itsestään selvää tämä ei suinkaan ole. Nyt yleisesti käytetyt ohjelmistojen kehitysmenetelmät alkavat olemaan suorituskykynsä äärirajoilla, ja on melkoinen yllätys, että tietojärjestelmät toimivat niinkin hyvin kuin toimivat. Onneksi tutkijoiden pöytälaatikoista löytyy vanhoja tuloksia, joiden avulla pystyy vastaamaan uuden ajan kasvaviin laatuvaatimuksiin.

Teollinen internet

Teollinen internet (Internet of Things, IoT) [1] sai alkunsa sellaisten autonomisten laitteiden ideasta, jotka tietoliikenneverkkoon liitettäessä alkavat keskenään vuorovaikuttaen ratkaista yhteistä ongelmaa (Kuva 1). Verkon topologia voi olla esimerkiksi perinteinen tähti, mutta vallankin laaja-alaisissa langattomissa sensoriverkoissa laitteet muodostavat itse tilanteeseen sopivan verkkotopologian ja oman varsinaisen tehtävänsä ohella myös reitittävät tarvittavassa määrin tietoliikennettä.

Tällaiselle rakenteelle on luontevaa laatia sovelluksia tilannekuvausten ja tapahtumien avulla. Esimerkiksi jos jonkin huoneen savu- ja häkäilmaisimet huomaavat ilmanlaadun poikkeavan normaalista, niin kaikki talon summerit alkavat hälyttää ja Internetiin tai matkapuhelinverkkoon kytketty viestiväline lähettää tilanteesta kertovan viestin ennalta määriteltyihin osoitteisiin. Jos tilanne pahenee edelleen ja hälytykseen ei kukaan reagoi, niin mahdollisesti myös automaattinen sammutuslaitteisto kytkeytyy toimintaan.

Kuva 1. Tulos syntyy yhdessä, tarkkaan koordinoidulla yhteistyöllä. (Kuva: Matti Mattila)

Siinä missä sovelluksen suunnittelijalle riittää laitteiden tarkastelu semanttisella tasolla, laitteiden valmistaja joutuu ottamaan kantaa kaikkiin vuorovaikutuksen tasoihin. Semanttisen tason ja tietoliikenteen lisäksi vuorovaikutusta on myös laitteen sisäisillä rajapinnoilla. Niiden kautta vuorovaikutetaan oheislaitteiden eli ympäristöä tarkkailevien antureiden ja sähkömekaanisten toimilaitteiden kanssa.

Teknisesti ottaen laitteen semanttisen tason toiminnallisuus ja vuorovaikutus toteutetaan oheislaitteiden kanssa vuorovaikuttaen ja välittämällä tietoa laitteelta toiselle tietoliikenteen yli. Tietoliikenne puolestaan toteutetaan protokollapinon ja viestipaketteja välittävien oheislaitteiden avulla. Koska oheislaitteet reagoivat ympäristön tilan muutoksiin ja toisten laitteiden tekemisiin, yksinkertaisinkin sensori (anturi + tietoliikenne) toimii reaktiivisesti. Lisäksi toiminta on rinnakkaista, sillä

esimerkiksi muutokset ympäristössä ja toisten laitteiden toiminta ovat toisistaan riippumattomia. Ja ihan kuin ei rinnakkaisuudessa ja reaktiivisuudessa olisi tarpeeksi hanskattavaa, sekä toimilaitteiden että tietoliikenteen toteuttavien komponenttien kanssa pitää huolehtia myös reaaliaikavaatimusten toteutumisesta.

Erillisistä ja palikoista koottaviin etäohjattaviin järjestelmäkokonaisuuksiin liittyy aina myös uhkia ja riskejä. Joku vihamielinen taho voi murtautua järjestelmään ja valjastaa sen vahingontekoon. Tai järjestelmän toteutukseen jääneet virheet voivat aiheuttaa turvallisuusriskin tai suuria liiketaloudellisia menetyksiä. Niinpä tavanomainen laatutaso ja luotettavuus ei enää riitä, vaan kehitysprosessilta vaaditaan uudenlaista suorituskykyä vastata rinnakkaisuuden, reaktiivisuuden ja reaaliaikavaatimusten mukana tuleviin haasteisiin.

Rinnakkaisohjelmoinnin lyhyt oppimäärä

Rinnakkaisuus tulee vastaan jo yksinkertaisinta sensoria rakennettaessa. Tietoliikennettä ja anturia pitää pystyä hallitsemaan toisistaan riippumatta, koska tietoliikenteessä tapahtuu paljon muitakin asioita kuin vain kerätyn datan lähettäminen eteenpäin. Niinpä törmätään klassiseen yhteisen muistin käytön ongelmaan eli miten varmistetaan, ettei puskurimuistista lueta lähetettävää dataa silloin kuin kirjoittaminen on kesken ja puskurissa oleva data pelkkää roskaa.

Ongelmaan on kaksi periaatteeltaan erilaista ratkaisua. Ensimmäisessä vaihtoehdossa huolehditaan, ettei kukaan pääse lukemaan, kun muistiin ollaan kirjoittamassa, eli käytetään perinteistä poissulkemista. Pseudokoodina:

```
pyydetään lupaa lukea/kirjoittaa
odotetaan vuoroa
      luetaan/kirjoitetaan
luovutaan oikeudesta käyttää muistia
```

Toisessa vaihtoehdossa lukija varmistaa datan eheyden dataan liittyvän kahdennetun versionumeron avulla. Datan kirjoittaminen pseudokoodina:

```
kasvatetaan versionumeroa A
kirjoitetaan dataa
asetetaan versionumero B samaksi kuin A
```

ja datan lukeminen vastaavasti:

```
luetaan versionumero B
luetaan data
luetaan versionumero A
data hyväksytään, jos A ja B samat,
      muuten luetaan uudestaan
```

Olennaista on huomata, kuinka sotkuiseksi yksinkertainen muistin lukeminen menee, kun joku toinen osapuoli voi kirjoittaa sitä kesken lukemisen. Samalla nähdään, miten helposti ja huomaamatta voi pienellä huolimattomuusvirheellä tuhota koko järjestelmän toiminnan. Esimerkiksi jos luku- tai kirjoitusoikeus jää luovuttamatta pois tai toinen datan versionumeroista jää päivittämättä, järjestelmä jää jumiin. Tällainen virhe tapahtuu helposti silloin, kun suoritus ei päätykään normaalisti vaan keskeytyy esimerkiksi poikkeukseen.

Laadun kannalta edellä hahmotellut virheet ovat todella ongelmallisia. Fyysisesti ne ovat syvällä matalimman tason koodissa, mutta tulevat esiin vain järjestelmätason toiminnassa. Niinpä niitä on vaikea huomata testaamalla, mutta loppukäyttäjä törmää niihin varmasti. Ja kun loppukäyttäjä normaalin työn tuoksinassa huomaa virheen, virheraporttiin ei päädy kaikkia niitä tietoja, jotka tarvittaisiin virheen toistamiseen ja paikantamiseen.

Noista muutamista koodinpaloista on huomattavissa myös reaaliaikavaatimusten toteutumisen kannalta ongelmallisia piirteitä. Algoritmeissa on epämääräisen mittaisia odotusaikoja tai asioita voidaan joutua tekemään määräämättömän monta kertaa, ennen kuin tarvittavat tiedot ovat käytettävissä. Keinoja ajoituksen hallintaan toki on, mutta ne tekevät koodista entistä sekavampaa ja vaikeammin ymmärrettävää.

Vuorovaikutuksen teoria

Teollisen internetin laitteiden ja sovellusten valmistajien näkökulmasta hyvä uutinen on, että rinnakkaisohjelmointiin on olemassa käyttökelpoinen teoria. Huono uutinen puolestaan on, että kyseinen teoria on syntynyt formaalien menetelmien tutkimuksen yhteydessä ja vain harvat tuntevat sen ja osaavat hyödyntää sitä ohjelmistotyössä. Onneksi teorian perusteet ovat melko yksinkertaiset.

Teorian keskeinen käsite on tapahtuma, jota sovelluslogiikan näkökulmasta voidaan pitää jaka-

mattomana. Esimerkiksi viestin lähettäminen radioteitse on toteutusteknisesti monivaiheinen tapahtuma, mutta sovelluksen toiminnan kannalta olennaista on, lähetettiinkö viesti vai ei. Siispä sovelluksen toiminnan kuvauksessa viestin lähettäminen voidaan esittää jakamattomana tapahtumana.

Tapahtumaan liittyy aina tekijä. Esimerkiksi hälytyksen kytkeytyminen päälle on hälyttimen tapahtuma ja ilman häkäpitoisuuden nouseminen yli raja-arvon ympäristön tapahtuma. Ympäristön tapahtumiin järjestelmä ei voi vaikuttaa suoraan eli ei voi estää niitä tapahtumasta tai pakottaa tapahtumaan. Toisaalta teoria antaa luvan jättää kiinnostamattomat ympäristön tapahtumat huomiotta.

Vuorovaikutus esitetään yhteistapahtumien avulla. Yhteistapahtuma on tapahtuma, jolla on monta tekijää eli osallistujaa. Esimerkiksi mekaaninen lämpömittari kytkee releen päälle lämpötilan noustessa yli raja-arvon. Relettä tarkkaileva I/O-linjan ohjain havaitsee jännitteen nousun ja nostaa prosessorin keskeytyslinjan päälle, minkä seurauksena prosessori alkaa suorittamaan keskeytysten käsittelijää. Eli semanttisessa mielessä tapahtumia on yksi, lämpötilan nousu yli raja-arvon, ja osallistujia neljä, lämpömittari, rele, I/O-linjan ohjain ja prosessori, joista kukin osallistui tapahtumaan omalla tapahtumallaan. Ominaista yhteistapahtumalle on, että kaikki osallistujat tietävät myös toistensa tietävän, mitä on tapahtunut.

Viestinnän suuntaa kuvaavat roolit, lähettäjä ja vastaanottaja, ovat tarpeettomia, koska vuorovaikutuksessa välittyvä data on osa osallistujien kohtaamista kuvaavaa yhteistapahtumaa. Kuin paperi, jota kaikki katsovat yhdessä. Jos jokin osallistujista haluaa yksikäsitteisen arvon tapahtumaan liittyvälle datalle, tapahtuma voi realisoitua vain kyseisellä datan arvolla. Toinen osallistuja taas voi hyväksyä datalle useita vaihtoehtoisia arvoja ja tallettaa realisoituneen arvon itselleen myöhempää käyttöä varten. Näin kunkin järjestelmän osan toimintaa voidaan tarkastella itsenäisenä kokonaisuutena, ja järjestelmän toiminta kuvata osien toimintojen avulla pelkästään kertomalla, millä tapahtumalla mikin osa osallistuu mihinkin yhteistapahtumaan vai jättääkö osallistumatta.

Rinnakkaisuus ja reaaliaika

Vuorovaikutuksen teorian esittelyssä tuli vastaan jo yksi rinnakkaisuuden muoto, yhteistapahtumat. Teoria ei kuitenkaan ota kantaa, suorittavatko osallistujat oman osuutensa yhteistapahtumasta rinnakkain vai peräkkäin tai jossain määrätyssä järjestyksessä. Olennaista on, että kaikki osallistujat hoitavat omat osuutensa ja ettei yksikään osallistujista suorita itsekseen yhteistapahtumaan osallistuvaa tapahtumaansa.

Toisistaan riippumattomiin tapahtumiin suhtaudutaan samoin kuin osallistujiin yhteistapahtuman sisällä. Teoria lähtee oletuksesta, ettei riippumattomien tapahtumien keskinäiseen suoristusjärjestykseen oteta kantaa. Jos tapahtumat voivat tapahtua missä tahansa järjestyksessä, ne voivat tapahtua myös rinnakkain.

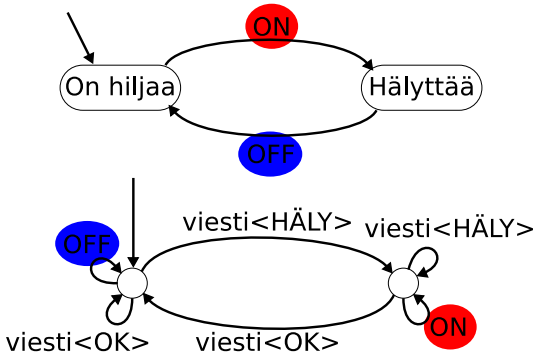
Tästä tavasta esittää rinnakkaisuus seuraa, millaisiin pienimpiin osasiin järjestelmän kuvaus pitää jakaa. Jokainen osanen, joka ei koostu pienemmistä osasista, ei voi suorittaa tapahtumiaan rinnakkain. Matalimman abstraktiotason osaset suorittavat siis omia tapahtumiaan pelkästään peräkkäin.

Perustason toimintakuvaus ei myöskään ota kantaa reaaliaikavaatimuksiin. Kuvauksissa tapahtumien suoritusaika oletetaan infinitesimaaliseksi, ja kuvauksissa esitetään kaikki loogisesti mahdolliset tapahtumien suoritusjärjestykset. Käytännössä tapahtumilla on reaalinen suoritusaika ja reaaliaikavaatimukset asettavat rajoituksia, missä järjestyksessä tapahtumat voidaan suorittaa. Ajettavassa järjestelmässä tapahtumien suoritusjärjestyksen valitsee toimintakuvauksesta erillinen vuorontaja, joka siis huolehtii myös reaaliaikavaatimusten toteutumisesta.

Esimerkintynkä

Sanallisesti on vaikea selittää, mitä järjestelmä tekee. Tapahtumiin ja vuorovaikutuksen teoriaan perustuvat toiminnan kuvaukset voidaan kuitenkin piirtää. Esimerkiksi kuvassa 2 on kuvattuna yksinkertaisen äänimerkillä hälyttävän summerin toiminta. Nimetyt nuolet kuvaavat tapahtumien suoritusta, ja niitä seuraamalla näkee, mitä järjestelmä voi tehdä. Mustalla reunaviivalla piirretyt pallukat ovat liitospisteitä, joista lähtevät nuolet kertovat, mitä voi tapahtua seuraavaksi. Värikkäässä lätkässä olevat tapahtumat esittävät

yhteistapahtumia, joita kuvassa on kaksi, hälytyksen aktivoiva ON ja sen lopettava OFF.



Kuva 2: Summeri.

Ideana on, että tietoliikenteen kautta tulevien viestien perusteella määräytyy tilamuuttujan arvo ja soittokellon tila seuraa kahden yhteistapahtuman kautta tilamuuttujan arvoa. Vuorovaikutuksen teoriaan liittyvä suoritusmalli pitää tapahtumia atomisina ja matalimman tason osanen voi suorittaa vain yhtä tapahtumaa kerrallaan, joten ohjelmoinnin kannalta hankala poissulkeminen ja sitä kautta monet rinnankkaisohjelmoinnin kummallisuudet hoituvat automaattisesti.

Suoritusmallista saa helpotusta myös reaaliaikavaatimusten kanssa elämiseen. Tapahtuman suorittaminen on ylipäätään mahdollista vain, jos se voidaan suorittaa keskeytyksettä alusta loppuun. Näin saadaan arvioitua yläraja tapahtuman suorittamiseen kuluvalle ajalle. Nyt toiminnan kuvausta eteenpäin simuloimalla saadaan selville sellaiset tapahtumien suoritusjärjestykset, jotka ovat aikarajojen puitteissa mahdolliset. Tässä esimerkissä toiminnan kuvaus oli piirretty käsin erityisefektien vuoksi. Käytännössä toiminnan kuvaukset kirjoitetaan tarkoitukseen sopivalla kuvauskielellä ja piirretään automaattisesti. Täysin painokelpoista jälkeä piirtoalgoritmit eivät tuota, mutta kuvista näkee kyllä, mitä kuvattu järjestelmä voi todellisuudessa tehdä.

Varusohjelmistot

Ohjelmiston toteuttaminen vaatii aina tuekseen työkaluja, ohjelmointikielen ja varusohjelmiston. Nykyisin yleisessä käytössä olevat kielet ja varusohjelmistot ovat kehittyneet vähitellen käytännön tarpeista lähtien, ja evoluution alkupisteenä oli ensimmäisten tietokoneiden ohjelmointi konekielellä. Rakennettavat järjestelmät olivat pitkään eräajotyyppisiä peräkkäisohjelmia.

Reaktiivisuus ja rinnakkaisuus ovat vaatimuksina paljon nuorempia kuin ohjelmistojen tekemisen käytännöt. Kielten ja varusohjelmistojen tuki reaktiivisten rinnakkaisjärjestelmien rakentamiseen edustaakin ad hoc -tyylisuuntaa, ja rinnakkaisohjelmien tekeminen koetaan tuskaisen vaikeaksi ja vaikeasti opittaviksi. Siitä puuttuu täysin se kauneus ja joustavuus, mitä vuorovaikutuksen teoria lupaa.

Vuorovaikutuksen teoriaan perustuvia ohjelmointikieliä on, mutta niitä ei yleisesti käytetä rakennettaessa sulautettuja järjestelmiä. Olisi mielenkiintoista tietää, miksi näin on. Yksi vaihtoehto toki on, että ohjelmointi on niin kaaoksen reunalla olevaa puuhaa, ettei teoriasta lähtien ole mahdollista tehdä käytännössä toimivia ohjelmointikieliä ja varusohjelmistoja. Todennäköisempi syy kuitenkin on, että vuorovaikutuksen teoria on rakennettu formaalien menetelmien, erityisesti verifioinnin yhteydessä, ja kielet vain unohdettiin samalla kun formaali verifiointi menetti uskottavuutensa noin viisitoista vuotta sitten.

Myöhempi tutkimuskin tukee unohdusteoriaa. Vuorovaikutukseen teoriaan perustuvaa ja käytännönläheiseksi koettua ohjelmointia on demottu useammassakin yhteydessä. Esimerkiksi professori Hannu-Matti Järvisen aktiolaskentaa käsittelevässä seminaarissa [2] oli harjoitustöiden toteutusympäristönä hänen tekemänsä aktiojärjestelmäsimulaattori, jolle kaikki osallistujat pystyivät tekemään ajettavan ohjelman. Tämä siitä huolimatta, että todellisuudessa kyseinen simulaattori ei ole varsinainen varusohjelmisto vaan tuotekehitysdemo hänen haaveilemastaan prosessorista, jossa aktiovuorontaja on toteutettu suoraan raudalla.

Tapahtumapohjainen ohjelmointi on tuttua myös mallipohjaisen testauksen kautta. Muutamissa suomalaista alkuperää olevissa mallipohjaisen testauksen työkaluissa, TEMA [3] , fMBT [4] ja OSMO Tester [5] , testit generoidaan tapahtumapohjaisena toiminnankuvauksena annetusta testimallista. Osa toteutetuista testimalleista on sen verran isoja kokonaisuuksia, että ne osaltaan todistavat vuorovaikutuksen teoriaan perustuvan ohjelmoinnin olevan mahdollista käytännössäkin.

Eikä aina edes tarvita varsinaista varusohjelmistoa. Joskus tapahtumien vuorontajaksi riittää pelkkä yksinkertainen tilakoneen samoilija. Ideaa

on kokeiltu yhdessä ohjelmointityötä helpottavassa apuvälineessä, joka automatisoi versionhallinnan käyttöä [6]. Sama kikka voisi olla käyttökelpoinen myös web-ohjelmoinnissa, koska JavaScript on pohjimmiltaan funktionaalinen ja dynaaminen kieli, jolla on helppo kikkailla, ja rakennettavat sovellukset ovat graafisia eli luonnostaan reaktiivisia.

Mitä seuraavaksi?

Vuorovaikutuksen teoriaan perustuvalla ohjelmoinnilla on todettu olevan muutamia etuja perinteiseen ohjelmointitapaan verrattuna. Formaalien mallien rakentaminen on jo sinänsä kustannustehokas tapa oppia ymmärtämään järjestelmän toimintaa ja kuvata se yksikäsitteisesti. Kannattaa myös huomata, että tapahtumiin perustuvilla esitystavoilla voi mallin rakentaa sovelluslogiikan käsittein ja sen abstraktiotasolla ja että mallia voi tarkastella kaikille helposti avautuvassa visuaalisessa muodossa, esimerkiksi yksinkertaisina tilakoneina. Ohjelmistosuunnittelun kannalta mielenkiintoisin ja lupaavin mahdollisuus on, että formaalilla toiminnan kuvauksella on operationaalinen semantiikka eli sitä voi testata ja verifioida sellaisenaan ilman loppuun asti vietyä toteutusta eli jo pelkän suunnitelman tasolla.

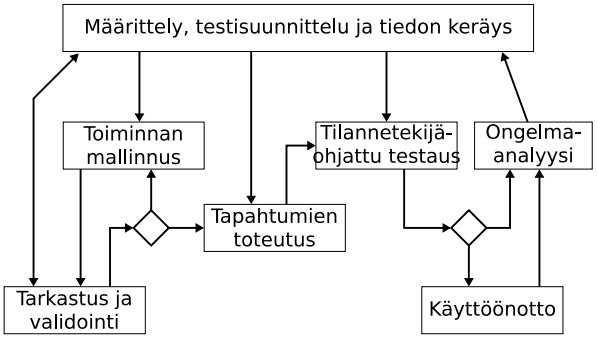
Lopullisesti vuorovaikutuksen teoriaan perustuvan ohjelmoinnin käyttökelpoisuus määräytyy siitä, miten hyvät ohjelmointiympäristöt ja -kielet sekä varusohjelmistot saadaan määriteltyä ja toteutettua. Tämä on tutkimusta vaativaa työtä, joka luonnostaan lankeaa akateemiselle maailmalle, mutta jota on mahdoton tehdä ilman teollisuuden tukea ja apua. Yksi ideana ilmaan heitetty lähtökohta on muokata joku pieni käyttöjärjestelmä, esimerkiksi RIOT-OS [7] tai Tiny os [8], ajamaan aktiojärjestelminä kuvattuja ohjelmia.

Parhaimmillakaan työkaluilla ei kuitenkaan tee mitään, jos kukaan ei osaa niitä käyttää. Niinpä järjestelmien suunnittelijoiden ja toteuttajien pitää opetella vuorovaikutuksen teoriaa ja tapahtumapohjaista ajattelutapaa. Tämä opettelu on kuitenkin sillä tavalla kiitollista, että saatuja oppeja pystyy hyödyntämään ja harjoittamaan käytännössä myös tavanomaisessa ohjelmistokehityksessä, erityisesti suunnittelussa ja testauksessa.

Toki uusien asioiden opettelu vaatii aikaa, joka on pois ihmisen normaalista työstä. Mutta onneksi, toisin kuin yleisesti uskotaan, ohjelmistoprojektiin pystyy ottamaan mukaan uuden ihmisen uusine tietoineen niin, että hänestä on hyötyä alusta alkaen ja ettei hänen opettamisensa vie aikaa muilta. Temppu tosin vaatii uutena mukaan tulevalta hieman erikoisosaamista ja normaalista poikkeavaa mielenlaatua, mutta on silti varteenotettava vaihtoehto perinteisille konsultteihin ja kursseihin perustuville koulutusjärjestelyille.

Testaus

Vuorovaikutuksen teoriaan hyödyntävässä ohjelmoinnissa testauksen rooli muuttuu aikaisemmasta. Mukaan tulee verifiointi, jota käytetään suunnittelun tukena samaan tapaan kuin automaattisia yksikkötestejä toteutusvaiheessa. Vastaavasti ajettavan ohjelman testaamista voidaan vähentää, koska testit saadaan kohdennettua tarkemmin. Uutta kehitysprosessia on hahmoteltu kuvassa 3.



Kuva 3: Ehdotetun kehitysprosessin luonnos.

Aikaisemmin jo mainittiin, että toimintamalli rakennetaan sovelluslogiikan käsittein ja sen abstraktiotasolla. Toimintamalli puolestaan muunnetaan ajettavaksi ohjelmaksi kirjoittamalla tapahtumille toteutukset, joten toimintamallin ja sitä kautta sovelluslogiikan mukainen rakenne säilyy myös toteutuksen tasolla. Ja koska toimintamallissa on yksikäsitteisesti kerrottu, milloin jokin yksittäinen tapahtuma voidaan suorittaa, tapahtuman toteutukseen kohdistuvat testit voidaan johtaa suoraan järjestelmän vaatimuksista ja jo aikaisemmin verifioidusta toimintamallista.

Nykyistä keskeisemmäksi nousee järjestelmätason tutkiva testaus. Sitä tarvitaan varmistamaan, että ollaan rakentamassa oikeaa järjestelmää, eitoiminnalliset vaatimukset toteutuvat ja etteivät varusohjelmiston virheet yllätä. Lisäksi pitää huolehtia kehitysprosessin laadusta eli arvioida käytettyjen työtapojen kykyä tuottaa laadukas lopputulos.

Toiminnalliseen testaukseen käytetty testiautomaatio sen sijaan muuttuu hyödyttömäksi. Testiautomaatioon kirjoitettu testi on jonkin vaatimuksen formaali esitys, joten se on kannattavampaa kirjoittaa suoraan toimintamalliin. Näin vaatimus tulee huomioitua suoraan jo suunnittelu-vaiheessa eikä sitä tarvitse kierrättää testaa, etsi ja korjaa silmukan kautta. Paitsi ehkä se ensimmäinen kerta, kun osa vaatimuksista kuitenkin löytyy ja tarkentuu vasta järjestelmää käytettäessä. Poikkeuksena ovat toki tilanteet, jossa saman ohjelman on tarkoitus toimia monella erilaisella alustalla tai testattava toiminnallisuus on sen luonteista, etteivät inhimilliset kyvyt riitä tarkastamaan sen kelpoisuutta.

Mahdollisuus

Uuteen teknologiaan perustuvan digitalisaation ja erityisesti teollisen internetin uskotaan tuovan Suomelle tilaisuuden luoda uutta kannattavaa tieto- ja viestintätekniikan alaan kuuluvaa liiketoimintaa [9]. Ja koska nykyiset ohjelmistojen kehitysmenetelmät ja -välineet alkavat käydä muutenkin vanhanaikaisiksi ja riittämättömiksi, näitä uusia järjestelmiä kehitettäessä olisi mahdollista luontevasti kehittää, testata ja ottaa käyttöön myös uuden sukupolven kehitysmenetelmiä. Näin suomalaisella ohjelmistoteollisuudella olisi paremmat mahdollisuudet säilyttää kansainvälinen kilpailukyky.

Viitteet

1) Wikipedia: Teollinen internet. http://fi.wikipedia.org/wiki/Teollinen_internet
2) Post graduate seminar TIE12206: Action based computations. http://www.tut.fi/idcprod/groups/public/@l814/@web/@p/documents/liit/p059073.pdf
3) TEMA Model Based Testing. http://tema.cs.tut.fi/, https://github.com/temambt
4) fMBT. https://github.com/01org/fMBT
5) OSMO Tester. http://www.testautomation.fi/index.php?page=osmo
6) autogitmode for Gnu Emacs. https://github.com/hessuvi/autogitmode
7) RIOT-käyttöjärjestelmä. http://www.riotos.org
8) Tiny OS käyttöjärjestelmä. http://tinyos.net
9) Tekesin ohjelma: Teollinen internet. http://www.tekes.fi/ohjelmat-ja-palvelut/ohjelmat-ja-verkostot/teollinen-internet/

Heikki Virtanen on sielultaan laatua rakastava ohjelmistokehittäjä ja tieteilijä, joka tuli testaajayhteisön jäseneksi keittiön kautta. Erääseen tutkimusryhmään tarvittiin testimallien teoriaa, työkaluohjelmistojen rakentamista ja Linux-palvelimen ylläpitoa taitava vahvistus.

Laatu ketterässä yrityksessä

Jukka Rapionoja, Knowit

Kun yritys tuo tuotteen tai palvelun markkinoille, asiakas tekee päätöksen tuotteen tai palvelun hankkimisesta usean tekijän summana: esim. ajankohta, koska tuote tulee markkinoille, hinta, toiminnallisuus, look&feel, brändi, laatu, jne.

Yrityksen, joka toimii markkinaehtoisesti ja haluaa toimia markkinatilanteen mukaan, kannattaisi pyrkiä, ja pitäisi pystyä, tekemään näihin tekijöihin liittyviä tietoisia päätöksiä painottaen eri markkinatilanteissa eri tekijöitä.

Tuotteen markkinoille tuomisen kestoon ja kuinka usein uusia versioita tuotteesta tai palvelusta voidaan tuoda markkinoille, laatuun, toiminnallisuuteen, hintaan ja jne. pystytään vaikuttamaan tuotteen tai palvelun kehittämistoimintatavoilla.

Yksi markkinatilanne on neitseelliset markkinat, jolloin uutta tuotetta tai palvelua ollaan tuomassa markkinoille, joilla ei toimijoita juuri vielä ole tai markkina on vasta kehittymässä. Neitseellisillä markkinoilla voi olla tärkeää saada tuote tai palvelu ennen kilpailijoita markkinoille. Tuotetta tai palvelua pitää sen jälkeen pystyä nopeasti kehittämään uusien tuote/palvelu-julkistusten kautta. Laatu voi olla tässä tilanteessa vähemmän merkittävä tekijä.

Toinen erityyppinen markkinatilanne on kiihkeän markkinaosuuskilpailun hetki, jolloin kilpailijat vyöryttävät nopeaan tahtiin uutta markkinoille. Tällöin voi olla tärkeää kyky luoda nopeasti pienissä inkrementeissä uutta ja tuoda muutokset nopeasti markkinoille. Tuotteen / palvelun luomis- ja markkinoilletuomistapojen on oltava kunnossa ja ohjattavissa.

Ja kolmas tilanne on kypsillä markkinoilla toiminen, jolloin kehitys on hitaampaa, markkina ei juuri kasva ja kilpailu käydään pääosin pienillä katteilla markkinaosuuksista. Kypsillä markkinoilla tuotteessa tai palvelussa painotus voi olla esim. look&feel, brändissä tai laadussa. Epäonnistuminen laadun kanssa oi käydä kalliiksi ja vaikeasti saatu markkinaosuus voi heiketä nopeasti. Toimintatapojen laadun rakentamisessa ja varmistamisessa on oltava kunnossa.

Ketterästi toimiva yritys on erityisesti markkinoille tulon ja nopean kasvun aikana vahvoilla. Mutta ketterät toimintatavat eivät myöskään sulje pois toimintaa markkinoilla, joilla tuotteelta vaaditaan enemmän kypsyyttä.

Ketterissä toimintatavoissa laatu pyritään rakentamaan tuotteen sisälle: se ei ole erillinen toiminto tai laatua ei ketterissä toimintatavoissa rakenneta ensisijaisesti testaamalla. Ketterissä menetelmissä laatuun vaikuttavat tekijät ovat mukana vahvasti toimintatavoissa. Säätämällä toimintatapoja, tai tekemällä toimintatapaan liittyviä valintoja, yritys voi säätää tuotteen laatua ja toimituksen nopeutta. Tämän lisäksi yritys voi käyttää tarpeen mukaan perinteisempiä laadun rakentamiseen ja mittaamiseen liittyviä toimintatapoja, liittäen ne omaan tapaansa käyttää ketteriä menetelmiä.

Laadunvarmistuksella on myös ketterissä menetelmissä tärkeä osa. Ei samoin toteutettuna kuin perinteisemmissä toimintatavoissa, mutta ketteriin menetelmiin sovitettuna. Testaajien, jotka ovat kehittämässä tuotetta - eivät testaamassa laatua, tulisi osata määrittää omia toimintatapoja ja niiden sisällä olevia laatukriteerejä niin, että toimitusten nopeutta voidaan säätää, jotta tuotteen laatutaso tiedetään koko ajan ja tuotteen markkinoille viennissä päätös voidaan tehdä faktojen perusteella laatu jatkuvasti mielessä, ja valitun laatutason saavuttaen.

Kiitos vai syytös. Kas siinä pulma

Antti Niittyviita, Prove

Raimo on sunnuntaisella iltakävelyllä kevätauringon jo laskettua. Hänen tiensä kulkee läheisen Prisman takaa pyörätietä pitkin sinne, missä kaupungin autokaupat sijaitsevat. Raimo huomaa Porhon autokaupan kohdalla, että portti on tainnut jäädä auki. Uteliaana kaverina hän käy kurkistamassa sisään. Piha on täynnä uudenkarheita ajopelejä ja portin pielessä seisovan Marsun ovetkin ovat jääneet auki. Ei siis muuta kuin uusi auto alle ja menoksi.

Rikoshan siinä tapahtui, kun tilaisuus teki varkaan.

Sama tarina tuli vastaan viimeviikolla monta kertaa. Luin nimittäin joukon artikkeleita Tekesistä ja tietoturva-aukosta[8]. Täytyy tunnustaa, että nyt pikkuisen suututtaa.

Helsinkiläisen it-firman kolme työntekijää olivat törmänneet tietoturva-aukkoon Tekesin järjestelmässä jättäessään oman firmansa hakemusta. Haavoittuvuuden ansiosta kavereiden onnistui ladata itselleen palvelun hakemustietokanta, eli noin 8300 hakemusta. Seuraavana päivänä miehet tekivät löydöksestään ilmoituksen Tekesille, mutta ei auttanut. Poliisi kävi hakemassa ja nyt rikosnimikkeenä on törkeä petos kaikkien kolmen kohdalla.

Minusta on turha selittää, että tämä karmea rötös olisi sama kuin auton varastaminen. Kun testauksen asiantuntija toteaa jonkin olevan pielessä, niin vain tutkimalla on mahdollista selvittää vian vakavuus.

Tietoturva-aukkojen tapauksessa on ensin kokeiltava, mitä sieltä tulee. Pötkö bittejä voi olla täysin joutilasta roskaa tai äärimmäisen arvokkaita yrityssalaisuuksia. On tärkeää tajuta, että softan kanssa asia saattaa selvitä ainoastaan jälkikäteen katsomalla mitä sieltä latautui.

Google muuten kiittää taitavia testaaja jopa sadan tonnin rahapalkinnoilla. Kovimmat sällit saavat poliisitutkinnan sijaan työtarjouksen. Pistää miettimään olisiko suomalaista innovaatiota ja yritystoimintaa edistävä Tekes pystynyt keksimään jotain parempaa ratkaisua tähän tilanteeseen?

Esimerkiksi julkinen kiitos yrityksen ja henkilöiden nimillä olisi saattanut tukea liiketoimintaa paremmin kuin yksikään tuotekehitysavustus. Tietovuodon olisi voinut hoitaa pois päiväjärjestyksestä parilla NDA:lla. Onhan näitä tapoja, kun vaan löysää vähän krakaa ja antaa ideoiden tulla.

Ihan alkuun lienee kuitenkin tärkeintä päättää: Syytös vai kiitos. Kas siinä pulma.

[8] http://www.iltasanomat.fi/digi/art-1425956952941.html

Luovuus testauksessa

Jussi Rautio, Knowit

Luovuutta ei yleensä yhdistetä testaukseen vaan se on useimmiten sisällöntuottajien määre. Mielestäni luovuudesta on hyötyä myös testaajalle ja hyväksi testaajaksi on vaikea tulla ilman luovuutta.

Tässä on joitakin luovuuteen yhdistettyjä termejä:

- Joustavuus
- Ennakkoluulottomuus
- Rajojen rikkominen
- Kokeileminen
- Estetiikan taju
- Reflektio
- Itsenäinen arvostelukyky
- Kyky käsitellä uusia asioita

Mielestäni nämä kaikki sopivat myös hyvän testaajan ominaisuuksiksi.

J**oustavuus** on kykyä muuttaa toimintaansa ulkoisten tekijöiden muuttuessa. Usein testisuunnitelman voi tehdä valmiiksi, mutta kun testauksen aika koittaa huomataan että toteutus on on kauniisti sanottuna elänyt määrittelyistä. Tällöin testaajan on ymmärrettävä muutokset ja toimia näiden mukaan.

Tietenkin myös **uuden omaksuminen** on tärkeää jokaiselle joka työskentelee IT-alalla. Se ei riitä että ATK-ajokortti on kerran ajettuna vaan omaa osaamista ja työtapoja tulee päivittää jatkuvasti.

Ennakkoluulottomuus on sitä, että pystyy ajattelemaan ilman valmiita oletuksia ja kehittää ajatuksensa vain empiirisen kokemuksen pohjalta. Tämä ominaisuus on testaajan toiminnan ydinmehua. Ei nojata valmiisiin olettamuksiin vaan edetään puhtaalta pöydältä ja suhtaudutaan kriittisesti ohjeistukseen. Tähän liittyy myös **itsenäinen arvostelukyky**. Mietitään kuinka tuote toimii pelkästään objektiivisesta näkökulmasta, ilman, että koetaan painetta tiimin aikatauluista tai työmääristä.

Reflektio eli oman toiminnan pohtiminen on mielenkiintoinen aihealue. Nykyisin kilpailu on kiristynyt IT-alalla jatkuvasti ja jokaisen yksilön kannattaa pohtia millä tavoin tuottaa lisäarvoa. Testaajan kannattaa hyödyntää erilaisia testaustapoja monipuolisesti ja hieman pohtia mitä on tekemässä ja miksi. On myös tärkeää miettiä voisiko asiat hoitaa tehokkaammin ja kuinka dokumentoida oma työnsä.

Luovien ominaisuuksien listasta puuttuu yleensä tunnollisuus. Oikeassa elämässä olen huomannut, että usein parhaat testaajat tekevät työtään hyvin systemaattisesti ja kurinalaisesti. Tunnollisuus on ehkä tärkein ominaisuus joka tuo menestystä manuaalisessa testauksessa.

Luovuus testauksen maailmassa liittyykin mielestäni enemmän työn suunnitteluun, organisointiin, tehokkuuteen, prosessien luomiseen ja yhteistyöhön. **Tunnollisuutta ja luovuutta ei pidä pitää toistensa vastavoimana, vaan hyvä testaaja käyttää kumpaakin ominaisuutta**.

Luovia ominaisuuksia siis tarvitaan myös testaajan työssä. Henkilökohtaisiksi näkisin tärkeimpinä:

- Kyky käsitellä uusia asioita
- Itsenäisen arvostelukyku
- Reflektio

Jos jokainen testaaja omaksuisi asiat nopeasti, tekisi oman arvionsa tuotteen laadusta itsenäisesti ja tehostaisi jatkuvasti toimintaansa, maailma olisi täynnä laadukkaita tuotteita!

Toimin puoliammattilaisena luovalla alalla. Tulin juuri Lapista soittamasta tuollaista shamanistisempaa maailmanmusiikkia.

Täysautomaattisesti sovellukset testaava ja korjaava uusi ohjelmistoalusta

Janne Nokiantausta

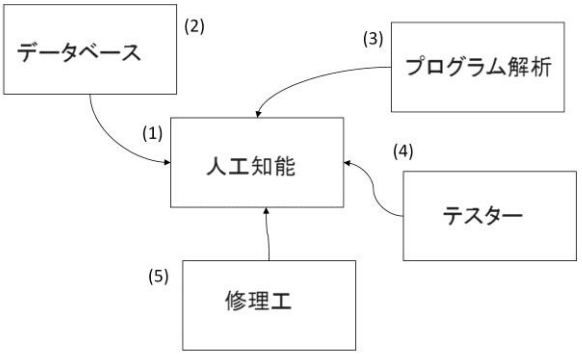
Tekoälystä on paljon kohistu viimeisen vuoden aikana. Se on kehittynyt valtavasti ja muuttaa jatkossa maailmamme, kun tietotekniikka voi korvata ihmisen yhä useammissa tehtävissä mm. robottien muodossa ja kaikki tekniset järjestelmät tulevat älykkäiksi, toistensa kanssa liittoutuviksi ja itseoppiviksi.

Nyt tekoälytekniikka on tullut myös käyttöjärjestelmätasolle osaksi arkista käytön teknistä infrastruktuuria ja sovellusten laadunvarmistusta uudessa Linux-pohjaisessa IX Planeta -käyttöjärjestelmässä, joka testaa itse itsensä ja kaikki sovellukset ja jos testauksessa löytyy virheitä, korjaa ne automaattisesti. Erityistä testausta ei enää tarvita osana sovelluskehitystä – ei yksikkötestausta, ei käytettävyystestausta tai tietoturvatestausta. Tästä koituu tuotekehitykselle valtavia säästöjä ja aivan uudenlaista nopeutta ja ketteryyttä.

Miten tämä magia tapahtuu, on vielä detaljien tasolla epäselvää. Käyttöjärjestelmää ei ole vielä julkaistu, mutta sosiaalisessa mediassa tänään (1.4.) jaettu teknologian japanilainen patenttihakemus ja siinä oleva kaavio systeemin arkkitehtuurista, jossa kuvataan sen pääpiirteet.

Keskeinen komponentti on tekoäly-yksikkö (kuvassa laatikko 1), joka ohjaa kaikkia muita komponentteja. Globaalisti kerättyä virhetietoa ja erilaisia korjausmalleja ja heuristiikkoja tarjoaa jatkuvasti päivittyvä tietokanta (2). Ohjelmia analysoi erityinen analysaattori (3), joka tunnistaa testattavat asiat ja priorisoi ne. Varsinainen testausyksikkö (4) huolehtii testauksesta. Löydettyjen virheiden perusteella korjausyksikkö (5) tekee tarvittavat muutokset ja rakentaa sitten ohjelmasta uuden, tilanteesta riippuen joko täysin virheettömän tai virheettömämmän version.

Korjaukset tehdään toistaiseksi lähdekoodiin, joka sitten käännetään uudelleen. Niinpä teknologia



IX Planeta -käyttöjärjestelmän automaattinen testaus- ja korjausjärjestelmä. Kuva patenttihakemuksesta.

toimii vain sovelluksille, joiden lähdekoodi ja käännöskomennot toimitetaan ohjelman mukana. Käytännössä tämä siis tarkoittaa avoimen lähdekoodin ohjelmia (OSS). Luonnollisestikin, jos sovellus on kovin huonosti ohjelmoitu, testaus-korjaussyklissä voi kulua joitakin minuutteja, mutta koska prosessi on täysin automaattinen, se voi tapahtua taustalla – ja käyttäjä voi samaan aikaan käyttää rikkinäistä versiota, aivan kuin tähänkin asti.

Kirjoittaja on globaalisti toimiva konsultti, joka on erikoistunut teknologian seurantaan ja uusiin hype-teknologioihin.

TestausOSY

Finnish Association of Software Testing

Kaikenlaisten softatestausihmisten yhteisö.
Yli 900 jäsentä ja kasvussa. Laadun ja sen tekijöiden asialla moniarvoisesti ja -äänisesti. Ajan tasalla ja edellä.

Jäsenetuina hyvän yhteisön ajatustenvaihtoa ja tekemistä, ilmaispääsy moniin tilaisuuksiin ja alennuksia kaupallisiin koulutuksiin.

Kaikki tämä ilman jäsenmaksua.

Liity jäseneksi osoitteessa
www.testausosy.fi

Tule LinkedIn-ryhmäämme keskustelemaan

LinkedIn

Matti Vuori 2014

TestausOSY
Finnish Association of Software Testing