Sovelluksen käytettävyyden testaaminen

2.6.2020

Standardin ISO 9241-11 mukaan tuotteen käytettävyys tarkoittaa tarkoituksenmukaisuutta, tehokkuutta ja tyytyväisyyttä, jolla tuotteen määritellyt käyttäjät saavuttavat määritellyt tavoitteet tietyssä käyttöympäristössä.

Sovellusohjelman käytettävyyden arviointiin on kehitetty useita erilaisia menetelmiä. Menetelmiä on kahta päätyyppiä: käyttäjätestaus ja heuristinen analyysi. Näistä päätyypeistä on erilaisin vivahtein johdettuja menetelmiä. Varsinaisten testausmenetelmien lisäksi käytetään varsinkin suunnittelun alkuvaiheessa erilaisia läpikäyntimenetelmiä karkeina testeinä.

Käyttöliittymän ja toimintalogiikan läpikäynti

Perinteinen tapa arvioida käytettävyyttä jo suunnitteluvaiheessa on käydä sovellusta läpi kohta kohdalta käyttäjien ja/tai asiantuntijoiden kanssa. Läpikäynti voi tapahtua aluksi ”paperiversiona” tai prototyypillä ja viimeisessä suunnitteluvaiheessa varsinaisella kehitettävällä sovelluksella. Suunnitteluvaiheessa arviointi kohdistuu pääosin vain käyttöliittymään, toimintalogiikka jää välttämättä vähemmälle huomiolle. Toimintalogiikka on kuitenkin yhtä tärkeä osa sovelluksen käytettävyyttä kuin varsinainen käyttöliittymäkin. Tietenkin käyttäjä ohjaa toimintoja käyttöliittymästä käsin, joten osittain myös rutiinit ja toimintalogiikka tulevat testatuiksi prototyypilläkin testattaessa.

Käytettävyystesti

Käytettävyystestauksessa sovellus tai sovelluksen prototyyppi testataan käyttäjien avulla ohjatussa ja valvotussa tilanteessa. Testin jälkeen asiantuntija analysoi testin tulokset. Käytettävyystesti on kohtuullisen suuritöinen järjestettävä, vaikkakin tutkimusten mukaan jo viidellä testihenkilöllä päästään kohtuullisen kattaviin tuloksiin. Jos vain mahdollista, käytettävyystestauksia olisi hyvä järjestää useita kertoja sovelluksen suunnittelun edetessä. Mitä pidemmälle virheet toimintalogiikassa menevät, sitä kalliimmaksi niiden korjaaminen tulee.

Käytettävyystestaus soveltuu erinomaisesti iteratiiviseen tuotekehitykseen, jossa tuotetta kehitetään asteittain paremmaksi. Käytettävyystestaus voi olla luonnollinen jatkumo toimintalogiikan ja käyttöliittymän läpikäynnin jälkeen. Käyttäjät saadaan heti mukaan tuotteen kehittelyyn ja käyttäjät ovat mukana koko tuotekehitysprojektin ajan. Toisaalta käytettävyystestausta voidaan tehdä myös jo käytössä olevalle järjestelmälle, jota ollaan uusimassa tai parantamassa. Testillä voidaan tuoda esiin sovelluksen hyviä ja huonoja puolia. Kehitystyössä voidaan ottaa opiksi vanhan järjestelmän hyvät ominaisuudet ja välttää huonoiksi havaittuja ominaisuuksia.

Jos sovellus on kovin laaja, kannattaa käytettävyystestaus kohdistaa kerrallaan vain tiettyihin

toiminnallisuuksiin. Toiminnallisuuksista muodostetaan sopivia käyttötapauksia, joista testaajien tulisi selvitä.

Käytettävyystesti voidaan toteuttaa siten, että seurataan käyttäjää käyttämässä sovellusta tiettyihin tehtäviin. Seurannan aikana arvioidaan käyttäjän kykyä suorittaa annetut tehtävät. Testissä tulee esiin todellisia käytännön tilanteissa vastaantulevia ongelmia. Testin onnistumisen yksi edellytys on testiin osallistuvien käyttäjien valinta, jotta saataisiin edustava otos käyttäjäkunnasta. Testaajina pitäisi olla sekä kokeneita tehtävän suorittajia että vasta-alkajia.

Heuristinen analyysi

Heuristiikat ovat tietynlaisia tarkastuslistoja, joiden avulla järjestelmä käydään läpi selvittäen, täyttyykö heuristiikassa esitetyt näkökohdat tarkastelun alla olevassa sovelluksessa. Heuristiikan avulla tarkastellaan yleensä koko järjestelmää, mutta on toki mahdollista, että sovitaan vain tietyt toiminnot tarkasteluun. Jotta heuristinen analyysi antaisi riittävän hyvän lopputuloksen, on analyysin suorittajalla oltava asiantuntemusta sekä käytettävästä heuristiikasta, että järjestelmän sovellusalasta. Heuristisessa analyysissä käytettävyysasiantuntija käy läpi sovelluksen kohta kohdalta sovellettavaa käytettävyysheuristiikkaa käyttäen. Ehkä tunnetuin käytettävyysheuristiikka on Nielsenin jo 1990-luvulla kehittämä ohjeisto (Nielsen 1994). Tämän heuristiikan pohjalta on kehitetty useita erilaisiin tarkoituksiin soveltuvia versioita.

Nielsenin heuristiikka

Nielsen on nimennyt kymmenen eri tarkastelunäkökulmaa testattavalle sovellukselle.

1. Järjestelmän tilan näkyvyys – riittävä palaute

Käyttäjän on oltava selvillä järjestelmän toimintatilasta.

Järjestelmän pitäisi huolehtia, että käyttäjä tietää mitä on meneillään antamalla kunnollista **palautetta** kohtuullisessa ajassa. Tämä on tärkeää siksi, että käyttäjälle tulee tunne toiminnan hallinnasta eikä käyttäjän tarvitsisi jäädä miettimään, tekeekö sovellus jotain, vai onko tapahtunut jokin häiriö. Käyttäjän tekemät asiat pitäisi olla selvästi näkyvillä, jotta myös vahingossa tehdyt asiat tulisivat huomatuksi ennen pahoja virheitä.

Erilaisia keinoja toimintatilan näyttämiseksi:

- Missä kohdassa toimintaa ollaan menossa: esimerkiksi kohdistimen sijainti, kohteen värin erottuminen, vierityspalkit ja sivu-/sanamäärä tai muu kertymätieto.
- Mitä käyttäjältä odotetaan: esimerkiksi keskusteluikkunat ja täytettävät kentät selityksineen.
- Vahvistus toiminnalle: esimerkiksi merkin ilmestyminen näytölle, värin muuttuminen, äänimerkki, värinä tai muunlainen ilmoitus asiasta.
- Ilmoitus virhetilanteesta.
- Varoitus peruuttamattomasta toimenpiteestä.
- Jos käyttäjä saa palautteen alle 0.1 sekunnissa, tulee tunne välittömästä vasteesta. Ei tarvita erillistä palautetta – lopputulos riittää.
- Jos käyttäjä saa palautteen alle 1 sekunnissa, käyttäjä huomaa viiveen, mutta keskittyminen ei katkea.
- Jos käyttäjä saa palautteen yli 10 sekunnin päästä, käyttäjä haluaa tehdä muita töitä odotellessaan.

2. Järjestelmän ja todellisuuden yhteensopivuus – käyttäjän kieli ja konteksti

Järjestelmän pitäisi puhua käyttäjien kieltä sanoin, lausein ja käsittein, jotka ovat käyttäjälle tuttuja. Käytetään mielellään käyttäjän äidinkieltä ja sovellusalueen ammattisanoja. Tietotekniikan ammattitermistöä ei ole suositeltavaa käyttää. Ei puhuta esimerkiksi tietokannasta vaan tiedoista. Puhelin/sähköpostisovelluksessakin puhutaan ihmisistä tai osoitekirjasta ei numerotiedostosta tai osoitetiedostosta.

Järjestelmän on seurattava reaalimaailman käytäntöjä ja esitettävä tiedot luonnollisessa ja loogisessa järjestyksessä. Myös vertauskuvien käyttö auttaa hahmottamisessa. Painikkeet/toiminnot kuvitetaan niihin helposti yhdistettävillä kuvakkeilla esimerkiksi kirje, puhelimen luuri, sakset ja maalipurkki.

3. Käyttäjän hallinta ja vapaus – poistumistiet

Käyttäjällä on aina oltava näkyvä poispääsy ohjelmasta tai sen osasta. Silloin käyttäjä uskaltaa kokeilla, kun hän tietää, että toiminnon voi perua. Jos paluuta ei ole, se on ilmoitettava ennen toiminnon käynnistymistä.

Käyttäjät valitsevat usein järjestelmän toimintoja vahingossa ja tarvitsevat selvästi näkyvän “hätäuloskäynnin” jättääkseen tilan, johon ei haluttu ilman, että heidän täytyy käydä läpi pitkää dialogia. Olisi hyvä olla olemassa myös mahdollisuus palata suoraan alkutilaan.

Käyttäjällä on oltava kontrollin tunne.

4. Yhdenmukaisuus ja standardit

Käyttäjien ei pitäisi joutua ihmettelemään tarkoittavatko eri sanat, tilanteet tai toimenpiteet samaa asiaa. Komentojen ja valintojen on toimittava yhdenmukaisesti. Esimerkiksi käyttäjät tunnistavat tulostuskomennon kirjoittimen kuvakkeen perusteella tai tulostustoiminto löytyy Tiedosto-valikosta. Tekstin elävöittämiseen käytetään synonyymejä, mutta sovelluksiin ne eivät kuulu. Painikkeilla on aina sama järjestys ja saman ohjelmaperheen sisällä myös samanlainen ulkoasu. Yhdenmukaisuus auttaa vähentämään käyttäjän muistikuormaa.

Järjestelmän on seurattava alustan käyttämiä tapoja (esimerkiksi Windows-alustalla käytetään Windows-käytäntöjä).

5. Virheiden estäminen

Hyviäkin virheilmoituksia parempi on huolellinen suunnittelu, joka estää ongelmien syntymisen.

Käytetään esimerkiksi valintalistoja tai muita muistin tukia, varmistetaan peruuttamattomat toiminnot, sijoitetaan kriittiset toiminnot kauas rutiineista, esimerkiksi Poista- ja Tallenna-painikkeita ei sijoiteta vierekkäin. Edelleen estetään etukäteen sellaisten toimenpiteiden valinta, jotka eivät tilanteeseen sovi, esimerkiksi Liitä-toimenpide harmaana, kunnes jotain tietoa on kopioitu tai leikattu.

6. Tunnistaminen muistamisen sijaan – muistikuorman minimoiminen

Tietokone on hyvä muistamaan asioita, ei rasiteta käyttäjän muistia tarpeettomasti. Jos tietoa on jo kertaalleen käsitelty, pidetään se tallessa ja käytetään sitä. Käyttä6. än ei pitäisi joutua muistelemaan yhden dialogin tietoa toisessa dialogissa. Samoin järjestelmän oliot, toiminnot ja vaihtoehdot on oltava näkyviä ei muistettavia.

Jos käyttäjän pitää syöttää tieto tietyssä muodossa, käyttäjälle esitetään malli. Mahdollisuuksien mukaan esitetään valmiiksi oletusarvo, esimerkiksi tilauspäivämäärä merkitään kuluvaksi päiväksi. Numeerisesta tiedosta esitetään mahdolliset raja-arvot ja käytettävä yksikkö. Esimerkiksi Discovery-avaruussukkula menetti avaruuspeilin yksikkövirheen vuoksi. Asetuksessa piti käyttää maileja ja astronautti käytti yksikkönä jalkoja, 10023 jalasta tuli 10023 mailia.

Käyttöohjeiden pitäisi näkyä tai olla helposti haettavissa koska vain tarvitaan.

7. Joustavuus ja käytön tehokkuus – oikopolut

Nopeuttajat – joita aloitteleva käyttäjä ei näe – voivat usein nopeuttaa asiantuntijakäyttäjän toimintoja siten, että järjestelmä voi palvella sekä kokemattomia että kokeneita käyttäjiä. Windows-ohjelmissa esimerkiksi Ctrl+C, Ctrl+V jne.

Järjestelmän tulisi sallia käyttäjien räätälöidä säännölliset toiminnot.

8. Esteettinen ja minimalistinen suunnittelu – yksinkertainen ja luonnollinen dialogi

Käyttöliittymän tulee olla mahdollisimman yksinkertainen, koska jokainen lisäpiirre tai asia on:

- yksi lisäasia opeteltavaksi
- yksi lisäasia, joka voidaan ymmärtää väärin
- yksi lisäasia, joka täytyy huomioida, kun käyttäjä etsii jotain asiaa näytöltä.

Dialogien ei pitäisi sisältää informaatiota, joka on epäolennaista tai jota tarvitaan harvoin. Jokainen lisäinformaation palanen dialogissa kilpailee olennaisten kanssa ja vähentää niiden suhteellista näkyvyyttä.

Käyttöliittymän tulee vastata käyttäjän suorittamaa tehtävää. Esillä tulee olla vain tarvittava tieto ja juuri silloin kun sitä tarvitaan. Tietojen esitysmuodon on tuettava käyttäjää jäsentämään näytöllä olevat asiat:

- tiedoilla on luonnollinen ja looginen järjestys
- tiedot, joita tarvitaan yhtä aikaa on sijoitettava lähekkäin
- tiedot ja painikkeet, joilla niitä käsitellään sijoitetaan lähekkäin
- tietojen järjestyksen pitää olla käyttäjän käsittelyjärjestys
- värejä, vilkutuksia ym. käytetään vain harkiten huomion herättämiseen
- käytetään vain muutamaa perusväriä koko sovelluksessa

9. Virheistä toipuminen – selkeät virheilmoitukset

Järjestelmän tulee auttaa käyttäjiä tunnistamaan ja diagnosoimaan virheitä ja palautumaan niistä.

Virheilmoitukset pitäisi ilmaista yksinkertaisella kielellä (ei koodeja).

Virhe tulisi ilmaista tarkasti ja ehdottaa rakentavasti ratkaisua.

10. Apu ja dokumentaatio

Vaikka on parempi, että järjestelmää voidaan käyttää ilman dokumentaatiota, voi olla tarpeen tarjota järjestelmän aputoiminto ja dokumentaatio.

Dokumenttitiedon tulee olla helppoa etsiä, painottua käyttäjien tehtäviin, listata konkreettiset askelet, joita suoritetaan eivätkä ne saa olla liian suuria.

Yhteenvetona voisi käytettävyystestausmenetelmistä sanoa, että ne tuottavat kukin tietoa suunnittelijoille siitä, mikä toimii ja mikä ei toimi. Suunnittelijoiden on valittava omalle järjestelmälle parhaiten soveltuva menetelmä. Menetelmän valintaa ohjaa tietenkin myös resurssit, taloudelliset, aikataululliset ja henkilöresurssit. Käytettävyystestaukseen uhrattu aika ja muut panokset ovat yleensä vaivan arvoisia. Testeillä voidaan välttää täydellinen epäonnistuminen tai marginaalikäyttöön jääminen.

Hilkka Niemelä
Tietotekniikan lehtori
SeAMK Tekniikka

Lähteitä

Nielsen, J. 1994. 10 Usability Heuristics for User Interface Design. [Verkkojulkaisu]. Nielsen Norman Group. [Viitattu 28.5.2020]. Saatavana: https://www.nngroup.com/articles/ten-usability-heuristics