Opinnäytetyö (AMK)

Tieto- ja viestintätekniikka

2023

Niklas Nordman

Mobiilisovelluksen testausmenetelmät

TURKU AMK
TURKU UNIVERSITY OF APPLIED SCIENCES

Opinnäytetyö (AMK) | Tiivistelmä

Turun ammattikorkeakoulu

Tieto- ja viestintätekniikka

2023 | 38 sivua

Niklas Nordman

Mobiilisovelluksen testausmenetelmät

Mobiilisovellusten testaaminen on tärkeä osa-alue laadukkaan sovelluksen tuottamisessa. Sovelluksen huolellinen testaaminen sisältää muun muassa toiminnallisuuksien, käytettävyyden ja tietoturvan tarkkaa arviointia. Tässä opinnäytetyössä tutkittiin mobiilisovelluksen testaamiseen liittyviä menetelmiä, tavoitteita ja työkaluja. Testausmenetelmät tutkittiin kategorioittain ja analysoitiin niiden vaikutusta julkaistavan tuotteen lopulliseen laatuun. Työssä tutkittiin myös eroja mobiilisovellusten sekä työpöytäsovellusten testausmenetelmissä.

Työn tuloksena selvisi, että testausmenetelmien käyttöönottoaikaisessa vaiheessa säästää koko kehitysprojektin kustannuksissa, koska virheiden korjaaminen projektin loppuvaiheilla on työlästä sekä aikaa vievää. Automaatiotyökalujen hyödyntäminen osoittautui välttämättömäksi apukeinoksi suurimmassa osassa testausmenetelmistä. Erityisesti mobiililaitteiden monimuotoisuuden vuoksi automaatiolla saavutetaan huomattavasti laajempi testikattavuus manuaalisiin keinoihin nähden. Tutkimuksen tulosten perusteella voidaan todeta, että parhaimman lopputuloksen mobiilisovelluksen testaamisessa saavuttaa valitsemalla manuaaliset testausmenetelmät tarkoin ja suorittaa suurin osa testauksesta automaation keinoin.

Työtä voisi jatkaa jakamalla mobiilisovellukset kategorioittain käyttötarkoitusten sekä toimintamekanismien mukaan ryhmiin ja tutkia eri testausmenetelmien vaikutuksia ryhmäkohtaisesti.

Asiasanat:

Mobiilisovellus, testaaminen, testausmenetelmät

Bachelor’s Thesis | Abstract

Turku University of Applied Sciences

Information and Communication Technology

2023 | 38 pages

Niklas Nordman

Mobile application testing methods

This thesis consists of research about the methods, goals and tools related to mobile application testing. The purpose of this thesis is to map and research mobile application testing methods as a part of application development quality assurance. The testing methods were mapped by category and their effect on the final product was analyzed. The work also explored the differences between mobile and desktop testing methods.

As a result of the work, it became clear that the introduction of testing methods at an early stage of the application development project will cut costs, because correcting errors in the final stages of the project is time-consuming and laborious. The use of automation tools proved to be a necessary aid in most testing methods. Especially due to the mobile device fragmentation, automation achieves a significantly higher coverage of tests compared to manual methods. Based on the result of the research, it can be stated that the best outcome in mobile application testing is achieved by carefully choosing the manual testing methods to use and getting automated tests to cover most of the test cases.

The work could be continued by dividing the mobile applications into groups by categories according to the purpose of use and operating mechanisms, and studying the different testing methods on a group-by-group basis

Keywords:

Mobile application, testing, testing methods

Sisältö

Kuvat

Käytetyt lyhenteet ja sanasto

API	sovellusohjelmointirajapinta, joka sallii useiden sovellusten kommunikoida toistensa kanssa.
Iterointi	toistettava työvaihe
Resoluutio	esitetyn kuvan yksityiskohtien tai pikselien määrä
Responsiivinen	käyttöliittymäsuunnittelu, jonka avulla sovellusta voi käyttää eri laitteilla ilman että käyttökokemus merkittävästi kärsii
Suoritusindikaattori	arvo, joka osoittaa kuinka tehokkaasti yritys saavuttaa tavoitteensa

Johdanto

Mobiililaitteet ovat kasvattaneet suosiotaan vuosi vuodelta, ja näin ollen myös yli puolet verkkovierailuista tänä päivänä suoritetaan käyttämällä mobiililaitetta. Mobiilisovellukset helpottavat laitteiden käyttäjiä suorittamaan arkea helpottavia tehtäviä paikasta riippumatta lataamalla sovelluskaupasta sovelluksen omalle mobiililaitteelleen. [1.]

Mobiilisovelluksen testaamisessa tulee ottaa huomioon käyttöjärjestelmän versio, ruudun koko, verkkoyhteyden laatu, akun kestävyys ja monia muita elementtejä, jotka ovat myös web-sovelluksille ominaisia. Mobiilisovelluksen testaaminen on olennainen osa laadunvarmistusta ennen kuin sovellus julkaistaan käyttäjien saataville sovelluskauppaan. Parhaimman käyttökokemuksen saavuttaminen ja käyttäjien tietoturvasta huolehtiminen on ensiarvoisen tärkeää kaupallisen mobiilisovelluksen menestyksen kannalta. [2.]

Opinnäytetyön tarkoituksena on tutkia ja selvittää mobiilisovelluksen testausmenetelmien vaikutuksia julkaistavan tuotteen laatuun. Testausstrategian valitseminen vaikuttaa suuresti sovelluksen laatuun, käyttökokemukseen sekä kulurakenteeseen. Jatkuvasti muuttuvassa sovelluskehityksen kentässä uusimpien trendien seuraaminen on tärkeää, koska tehokkaita automaatiotyökaluja julkaistaan jatkuvasti, ja ne voivat tehostaa kehitystyötä sekä testausta valtavasti [3].

Maailmanlaajuisten megatrendien vaikutus mobiilisovellusten testausmenetelmien kehitykseen on aihe, jota tarkastelemalla voi tehdä havaintoja myös testausmenetelmien kehityssuunnasta. Energiatehokkuus ja tekoälyn hyödyntäminen avaavat testaukseen uusia mahdollisuuksia, joilla on suoria vaikutuksia prosessin kustannustehokkuuteen sekä lopputuotteen energiatehokkuuteen. [4.][5.]

Mobiilisovelluksen testaaminen

2.1 Yleistä

Mobiilisovelluksen testaaminen kuuluu tärkeänä osana sovelluskehitystyötä osana laadunvarmistuksen prosessia. Mobiilisovelluksen tyyppi vaikuttaa testausmenetelmien valintaan. Web-sovelluksien testaamisessa tärkeitä testattavia osa-alueita ovat responsiivisuus ja selainten yhteensopivuus. Natiivi- ja hybridisovelluksien testaamisessa huomiota vaatii laitekohtaiset resurssit, kuten kameran, paikannuksen, eleiden ja ilmoitusten toiminta. [6.]

Mobiililaitteiden monimuotoisuus eli huomattava variaatio laitteiden ruudun kokojen, kuvan tarkkuuden, valmistajan, käyttöjärjestelmän ja laitteistokokoonpanojen osalta tuo mukanaan mobiilisovellusten testaamiseen lisää haastetta. Laitteiston ja sovelluksen välinen yhteensopivuus saadaan testattua aloittamalla testaaminen aikaisessa vaiheessa ja tekemällä sitä jatkuvasti projektin edetessä. [7.]

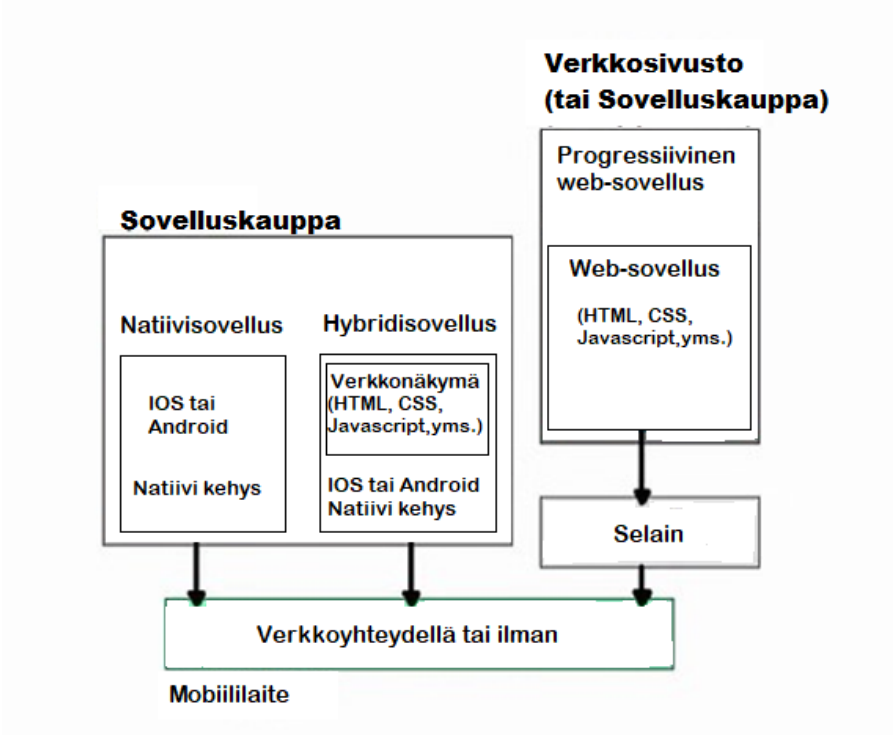
Mobiilisovelluksien kehittämiseen ja testaamiseen on useita malleja ja metodeja, joista yksi suosituimmista on V-malli. V-mallissa testauksen tasot määritellään pienemmistä komponenteista suurempaan kokonaisuuteen aloittamalla testaus yksittäisen yksikön toimivuuden varmistamisella siirtyen yksikköjen ja moduulien yhteistoiminnan tarkastelemiseen. Kun komponenttien välinen toimivuus on varmistettu, siirrytään koko järjestelmän kattavaan testaamiseen, jossa mitataan suorituskykyä, luotettavuutta ja tietoturvaa. Lopulta saavutetaan hyväksyntätestaus, joka suoritetaan usein testikäyttäjien avulla varmistaen, että kaikkiin tavoitteisiin on päästy ja että käyttökokemus on riittävällä tasolla. [8.]

Tarkkoihin tasoihin määritellyn V-mallisen kehitysmallin vastakohta on toinen suosittu metodi nimeltä ketterä sovelluskehitys. Ketterän sovelluskehityksen

projekteissa sovelluksen kehittäminen ja testaaminen perustuu jatkuvasti kehittyviin ominaisuuksiin ja tarpeisiin. Asiakkaan ja kehitystiimin välinen kommunikaatio on keskiössä, jolloin sovelluksen lopputulos on enemmän räätälöitävissä asiakkaan muuttuvien tarpeiden ja toivomusten mukaan. [9.]

2.2 Mobiilisovellusten tyypit

Mobiilisovellukset jaetaan kolmeen kategoriaan niiden toimintaperiaatteen mukaisesti. Osa sovelluksista voi toimia täysin ilman yhteyttä verkkoon taikka tietokantaan, kun taas osa sovelluksista vaatii jatkuvan yhteyden verkkoon. Mobiilisovellusten tyypit ovat havainnollistettu kuvassa 1. Natiivi- sekä hybridisovellusten lataaminen laitteelle tapahtuu sovelluskaupassa, kun taas web-sovellus saattaa toimia täysin verkkoselaimen kautta ilman sovelluksen lataamista mobiililaitteelle.



Kuva 1. Mobiilisovellusten tyypit [10 muokattu alkuperäisestä].

2.2.1 Natiivisovellukset

Natiivisovellukset on tehty varta vasten tiettyä käyttöjärjestelmää varten hyödyntäen kyseisen järjestelmän ominaisuuksia sekä kehitystyökaluja. Natiivisovellus ladataan sovelluskaupasta laitteen muistiin, ja tällöin sovellusta voi laitteellaan käyttää verkkoyhteydellä tai ilman. Natiivisovelluksen etuna selainpohjaiseen sovellukseen verrattuna on käyttökokemus sekä suorituskyky. Sovellus pääsee hyödyntämään laitteen kameraa, paikannuspalvelua ja ilmoituspalveluita laaja-alaisesti taaten sulavan käyttökokemuksen. [10.]

2.2.2 Web-sovellukset

Web-sovellukset ovat responsiivisia verkkoselaimella käytettäviä sovelluksia, joita ei ladata erikseen käytettävälle laitteelle. Web-sovellukset toimivat kaikilla alustoilla, joiden kautta voi selailla verkkoselaimen kautta internetiä. Selainpohjaisen web-sovelluksen tekeminen on yksinkertaisempaa ja edullisempaa, mutta sujuvan käyttökokemuksen saaminen mobiililaitteille vaatii testausmenetelmien tarkkaa suunnittelua. [10.]

2.2.3 Hybridisovellukset

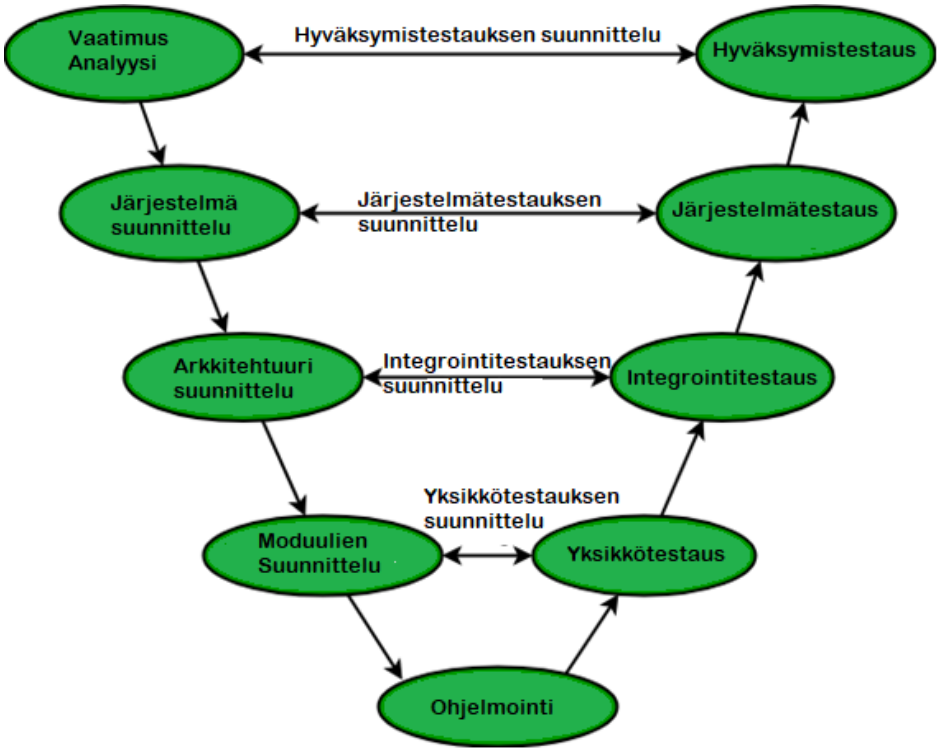
Hybridisovellukset ovat nimensä mukaan yhdistelmä sekä natiivin että web-pohjaisen sovelluksen ominaisuuksia. Hybridisovelluksen taustalla on web-pohjaisten teknologioiden työkaluilla luotu runko, joka on paketoitu natiivisovelluksen kaltaiseen ulkoasuun. Tämä tuo mukanaan yksinkertaistetun kehitysprosessin sekä alustariippumattomuuden, jolloin tarvitaan vain yksi lähdekoodi sovelluksen toimimiseen useammilla mobiililaitteiden käyttöjärjestelmillä. Hybridisovelluksen testaaminen on monimutkaisempi prosessi koska se ei natiivisovelluksen tapaan pääse suoraan hyödyntämään laitteen ominaisuuksia tai täyttä prosessointitehoa. [10.]

2.3 Mobiililaitteiden monimuotoisuus

Mobiililaitteet eroavat toisistaan huomattavasti enemmän verrattuna suhteellisen vakioituihin laitteisiin pöytätietokoneiden sekä kannettavien tietokoneiden saralla. Älypuhelinten ruudun koko on kasvanut vuosi vuodelta, mikä tarkoittaa, että sovelluksen käyttäjäkunnan laitteet eroavat resoluutioltaan toisistaan melko paljon. Ruudun koon ja tarkkuuden lisäksi mobiililaitteiden käyttöjärjestelmät jakaantuvat sekä Applen IOS-pohjaisten laitteiden sekä Googlen Android-laitteiden lisäksi näiden käyttöjärjestelmien versioihin. Vanhemmat mobiililaitteet eivät päivity enää uudempiin käyttöjärjestelmiin, mutta mobiilisovelluksen tulisi silti toimia saumattomasti sekä uudemman että vanhemman version käyttäjällä. Sovelluksen testaajien pitää ottaa huomioon kaikki mahdolliset laitekokoonpanot, joilla sovelluksen käyttäjät saattavat haluta sovellusta käyttää mukaan lukien tulevat laitteet. Toisinaan sovelluksen kehittäjät joutuvat tekemään sovelluksesta kokonaan oman version käyttöjärjestelmän eri versioiden käyttäjille yhteensopivuuden takaamiseksi. [11.]

2.4 Testauksen V-malli

Testauksen V-malli on testausstrategia, jossa sovelluksen kehittäminen jaetaan neljään tuotannon vaiheeseen, joita varten on vastaavasti neljä erilaista testauksen tasoa. Testaus suoritetaan järjestelmällisesti pienimmästä yksiköstä suurempiin kokonaisuuksiin testaten lopulta koko sovelluksen toimintaa sekä liiketoiminnallisten vaatimusten toteutumista. Kuvassa 2 näkyy V-mallinen rakenne, joka etenee testauksen tasolta toiselle. Testauksen tasot etenevät vaihe vaiheelta mallin pykälien mukaan. [8.]



Kuva 2. Testauksen V-malli [8 muokattu alkuperäisestä].

2.4.1 Yksikkötestaus

Yksikkötestaus on matalimman tason testausmenetelmä, jolla varmistetaan pienimpien komponenttien toimivuus lähdekoodissa. Yksikkötestauksella pyritään löytämään virheitä yksittäisistä komponenteista kuten funktioista tai metodeista varhaisessa vaiheessa jolloin niiden korjaaminen on helpompaa ja edullisempaa. [12.]

2.4.2 Integrointitestaus

Integrointitestaus on sovelluksen yksikkötestauksesta seuraava testauksen taso. Integrointitestaus tarkoittaa yksiköiden välisten toimintojen sekä rajapintojen testaamista. Integrointitestaamisen suorittamiseen on neljä erilaista lähestymistapaa: Big-Bang, Bottom-Up, Top-Down sekä Mixed Integration Testing. Big-Bang integrointitestaus on yksinkertaisin integrointitestauksen muoto, jossa sovelluksen kaikki moduulit yhdistetään ja testataan niiden

toiminta kerralla. Big-Bang integrointitestaus sopii vain yksinkertaisille sovelluksille, joiden yksiköiden erillinen testaaminen on hankalaa. Bottom-Up integrointitestaus perustuu matalimpien tason moduulien testaaminen yksitellen korkeamman tason moduulien kanssa yhteensopivuuden takaamiseksi. Matalan tason ja korkean tason moduulien testaamisessa käytetään usein ajureita datan ohjaamiseen moduulien välillä. Top-Down-integrointitestauksessa testataan korkean tason moduuleja simuloiden matalan tason moduuleja, joita ei ole vielä integroitu kokonaisuuteen. Korkean tason moduulit testataan yksitellen samalla integroiden matalamman tason moduulit osaksi sovellusta. Mixed Integration Testing tunnetaan myös nimillä Kerrosintegrointitestaus tai Hybridi-integrointitestaus. Tämä testauksen muoto on hyödyllinen isompien projektien osalta, joilla on useampia osaprojekteja. Kerrosintegrointitestaus mahdollistaa matalan sekä korkean tason moduulien testaamisen samanaikaisesti, mikäli niiden välinen riippuvuus ei ole suuri. [13.]

2.4.3 Järjestelmätestaus

Järjestelmätestaus on integrointitestauksen jälkeinen testaustaso, jonka avulla testataan sovelluksen suorituskykyä, skaalautuvuutta, paineensietokykyä käyttäjämäärän kasvaessa ja asiakkaan asettamien tavoitteiden toteutumista. Järjestelmätestaus toteutetaan kehitystiimistä erillään olevan testaustiimin toimesta, joka takaa puolueettoman arvion sovelluksen tilasta ennen kuin se luovutetaan loppukäyttäjien saataville. [14.]

2.4.4 Hyväksymistestaus

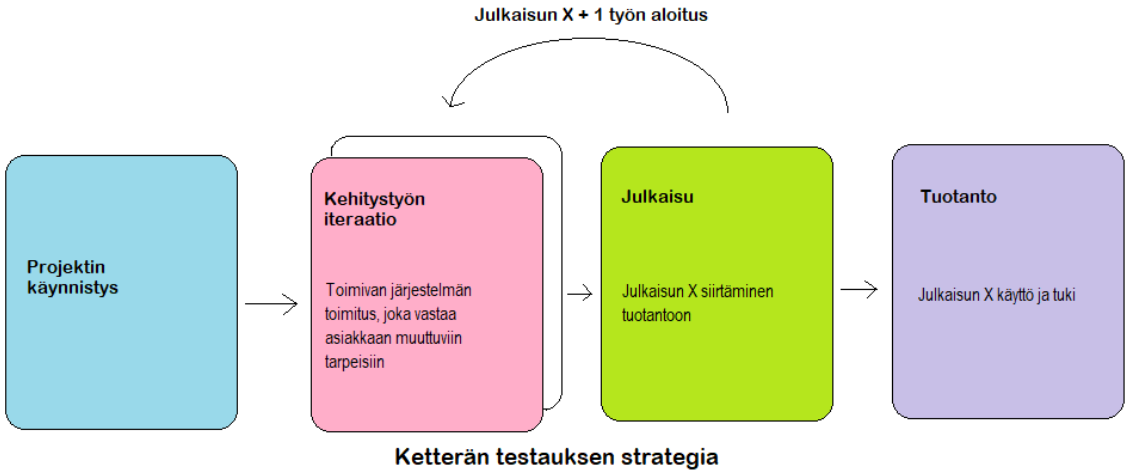
Hyväksymistestaus on viimeinen testauksen taso ennen sovelluksen julkaisemista käyttäjien saataville. Hyväksymistestauksen tavoitteena on tarkastella liiketoiminnallisten vaatimusten toteutumista ja lopulta päättää, hyväksytäänkö sovellus julkaistavaksi vai ei. Hyväksymistestauksen aikana koekäyttäjät antavat arvionsa sovelluksesta, ja tämän palautteen pohjalta

kehittäjät voivat tehdä viimeiset parannukset sovelluksen käytettävyyteen ennen sen julkaisemista. [15.]

2.5 Ketterä testaus

Ketterä testaus soveltaa ketterän sovelluskehityksen periaatteita sovellustestauksessa. Erillisen testausvaiheen sijasta ketterä testaus suoritetaan rinnakkain sovelluksen kehityksen kanssa. Ketterä testaus mahdollistaa jatkuvan palautteen jakamisen testaajien, asiakkaan ja kehitystiimin välillä soveltuen projekteihin, joissa asiakkaan vaatimukset sovelluksen suhteen kehittyvät projektin edetessä. Projektit, joissa asiakas pystyy osallistumaan aktiivisesti kehitystyöhön, ja keräämään palautetta testiryhmältä kehitystyön aikana, on ihanteellinen tilanne ketterän testauksen hyödyntämiselle. Sen avulla varmistetaan, että lopputuote vastaa parhaiten asiakkaan tarpeita.

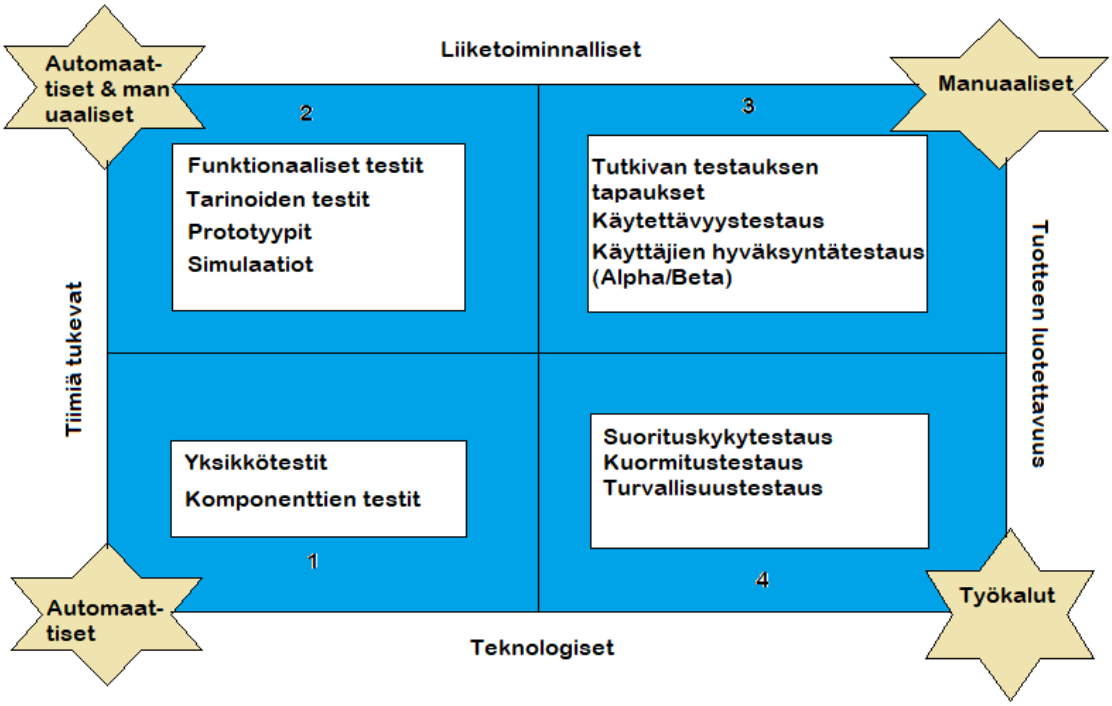
Ketterän sovelluskehityksen elinkaari alkaa testaustiimin ja työkalujen määrittelemisellä projektin käynnistysvaiheessa. Projektin laajuus, vaatimukset ja kustannusarvio arvioidaan ennen varsinaisen kehitystyön alkamista. Projektin seuraava vaihe käsittää itse sovelluksen kehitystyön, ja siihen liittyvät iteraatiot. Kehityksen kiertokulku etenee tyypillisesti noin kahden viikon jaksoissa alkaen kehitystiimin ja asiakkaan välisestä tapaamisesta, jossa määritellään tulevan jakson tavoitteet edellisen jakson ja asiakkaan antaman palautteen perusteella. Toisessa vaiheessa sovellusta testataan ketterän hyväksymistestauksen ja kehittäjätestauksen keinoin. Ketterä hyväksymistestaus testaa sovellusta hyväksymistestauksen ja funktionaalisen testauksen keinoin. Kehittäjätestaus testaa sovellusta yksikkötestauksen ja integrointitestauksen tavoilla. Kehitysvaiheen päättyessä ketterän kehitystyön projekti etenee kolmanteen vaiheeseen, jossa testauksen pääpaino siirtyy järjestelmätestauksen ja hyväksymistestauksen menetelmiin. Aktiivisen kehitystyön päätyttyä sovelluksen käyttöä testataan loppukäyttäjille sekä sitä markkinoidaan tulevaa julkaisua varten. Lopulta sovellus valmistellaan julkaisua varten ja korjataan viimeiset virheet. Ketterän testauksen kiertokulku on kuvattuna kuvassa 3. [16.]



Kuva 3. Ketterän testauksen strategia [16 muokattu alkuperäisestä].

Ketterän sovellustestauksen menetelmät ovat jaettu neljään kategoriaan niillä saavutettavien tavoitteiden mukaan. Ensimmäinen luokka sisältää yksikkötestauksen sekä komponenttitestauksen, joita suoritetaan automaatiotyökalujen avulla ratkoen projektin teknologiaan liittyviä haasteita. Toinen luokka sisältää liiketoiminnallisten haasteiden ratkomista sekä automaattisten että manuaalisten keinojen avulla varmistaen asiakkaan asettamien tavoitteiden toteutumista. Kolmannen luokan menetelmät tarjoavat informaatiota ensimmäisen ja toisen luokan menetelmien käyttöön testaten sovelluksen käytettävyyttä sekä luotettavuutta manuaalisin keinoin. Neljännen luokan testausmenetelmät ovat työkaluilla suoritettavia testaustapoja sovelluksen tietoturvan ja suorituskyvyn parantamiseksi. Kuvassa 4 esitetty testausmenetelmien nelijako on visuaalinen apukeino menetelmän valitsemiseen kulloisessakin tilanteessa. Menetelmien luokitteleminen eri kategorioihin helpottaa testaamisen suunnittelemista, koska ketterässä testauksessa ei ole yhtä tarkkaan määriteltyjä testauksen tasoja kuin V-

mallisessa testauksessa. [16]



Kuva 4. Ketterän sovellustestauksen menetelmät [16 muokattu alkuperäisestä].

Testausmenetelmät

3.1 Testausmenetelmien valitseminen

3.1.1 Sovelluksen käyttötarkoitus

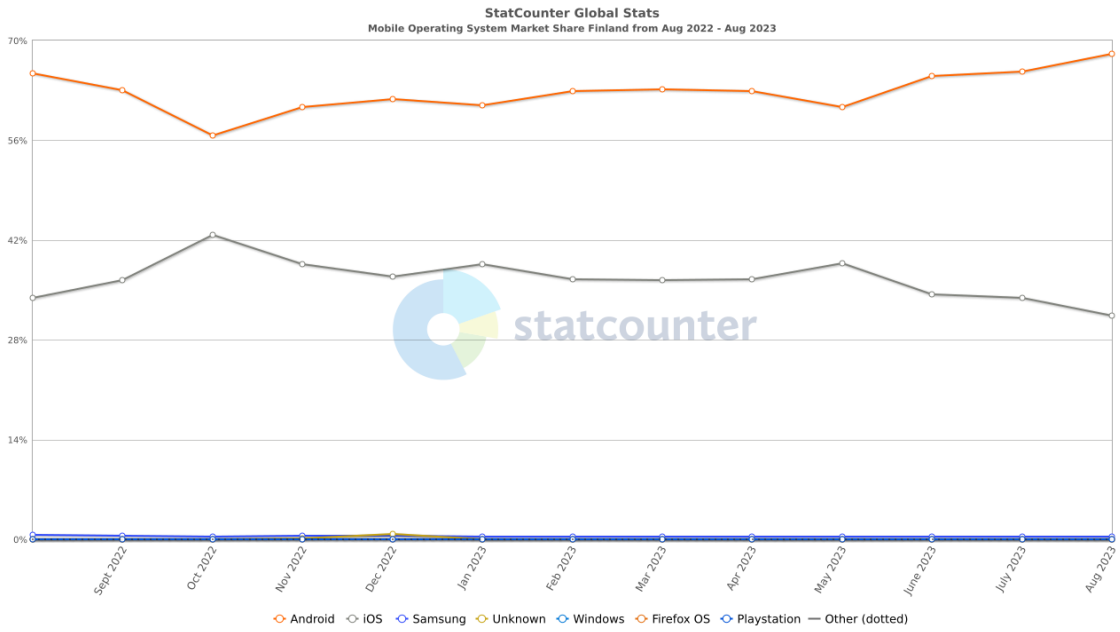
Testausmenetelmien valitsemisessa tulee ottaa huomioon kehitettävän sovelluksen käyttötarkoitus. Sovelluksen käyttötarkoitus voi olla viihteellinen, kaupallinen tai yleishyödyllinen, jolloin myös sovelluksen testaamisessa pääpaino vaihtelee käyttötarkoituksen mukaan. Verkkokauppasovelluksen toiminnassa ostotapahtuman testaaminen ja asiakkaan maksutietojen turvaaminen vaativat tarkempaa testaamista tietoturvaominaisuuksien toteutumiseksi. Mobiilialustalle suunniteltu videopeli vaatii saumatonta käyttökokemusta menestyäkseen kilpailluilla markkinoilla, joten tästä syystä testaamisen pääpaino on audiovisuaalisten elementtien toimivuudessa, suorituskyvyssä ja yhteensopivuudessa useiden alustojen sekä käyttöjärjestelmien osalta. Sovellukset, joiden tarkoituksena on helpottaa käyttäjää suunnistamaan vieraassa paikassa karttapalvelun avulla ovat luonnollisesti hyvin riippuvaisia GPS-paikannuksesta, jolloin testaamisessa tulee ottaa huomioon paikannuspalveluiden toimivuus sekä verkkoyhteydellä että ilman, ja mahdollisimman optimoitu akun kulutus, kun käyttäjä on liikekannalla eikä välttämättä pääse lataamaan laitettaan.

3.1.2 Testattavat käyttöjärjestelmäversiot

Kehitettävän sovelluksen kohderyhmän laitteiston monimuotoisuus vaikuttaa siihen, kuinka monella eri laitteella sekä niiden käyttöjärjestelmien versioilla sitä tulisi testata. Sovelluksen yhteensopivuus mahdollisimman monen laitteen kanssa kasvattaa sen käyttäjäkuntaa ja mahdollisuutta menestyä markkinoilla, mutta monimutkaistaa yhteensopivuuden testaamista. Tarkastelemalla eri

käyttöjärjestelmien markkinajakaumaa, voidaan kohderyhmä rajata ja kohdistaa laitekohtainen testaus tarkemmin. [17.]

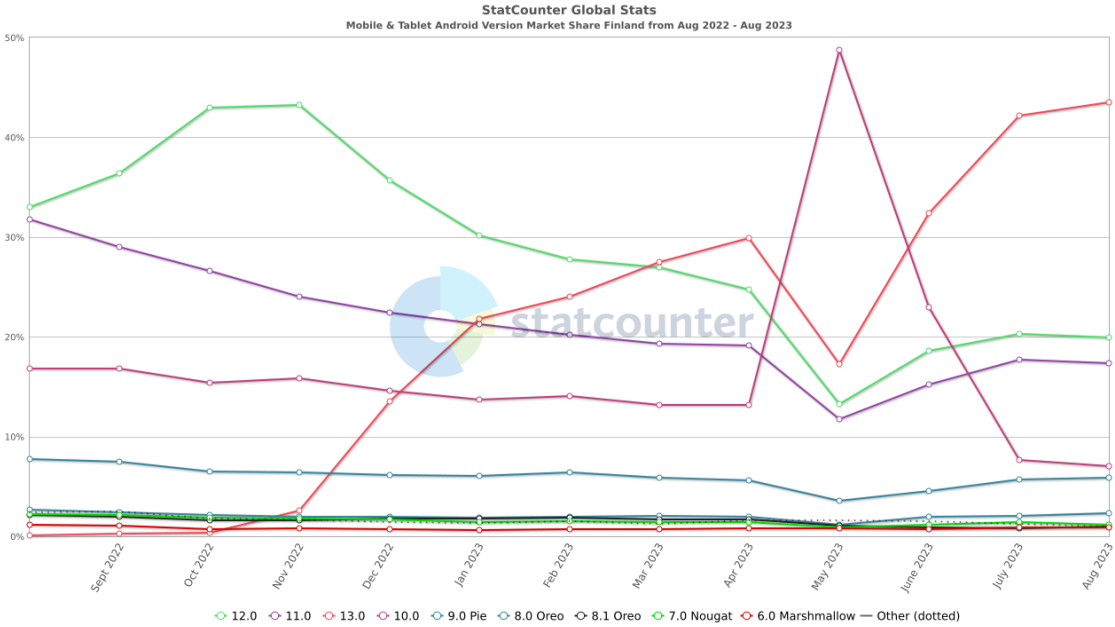
Kuvan 5 kaaviota tarkastelemalla voidaan tehdä päätelmä, että Android käyttöjärjestelmä on markkinajohtajana Suomessa käytettävistä mobiililaitteiden käyttöjärjestelmistä vuosina 2022–2023. Aina kuitenkaan päätöstä kohderyhmästä ei tehdä pelkästään enemmistöä edustavan käyttöjärjestelmä mukaan. Kohderyhmää rajataan muun muassa iän, tulotason, sukupuolen ja kiinnostusten mukaan [19]. Mikäli sovellus rajattaisiin pelkästään Android-käyttöjärjestelmälle, niin seuraavaksi tulisi tarkastella käyttöjärjestelmän versioiden jakaumaa markkina-alueella.



Kuva 5. Mobiililaitteiden käyttöjärjestelmät Suomessa vuosina 2022–2023 [18].

Android-käyttöjärjestelmää käyttävien laitteiden kaaviosta, joka esiintyy kuvassa 6, voi nähdä, että noin 80 % laitteista käyttää versioita 11–13. Testaamisessa pääpaino tulisi siis olla enemmistöä edustavissa käyttöjärjestelmän versioissa,

ellei sovellusta ole suunnattu erityisesti vanhemman sukupolven laitteille.



Kuva 6. Android-järjestelmän versiot Suomessa vuosina 2022–2023 [20].

Laitteiden monimuotoisuus ulottuu myös käytettävän ruudun resoluutioon ja valmistajaan. Monimuotoisuuden huomiointia testausmenetelmien valitsemisessa helpottaa laitematriisin tekeminen, jossa kohderyhmän käyttämän laitteiston erot kootaan yhteen taulukkoon, jonka perusteella voidaan päättää testataanko sovellusta oikealla mobiililaitteella, emulaattorilla tai simulaattorilla. [21.]

3.1.3 Manuaaliset ja automaattiset testit

Parhaimman lopputuloksen saavuttaminen mobiilisovelluksen testaamisessa pitää sisällään sekä manuaalisia että automaattisia testauskeinoja. Manuaalisten menetelmien avulla sovelluksesta saadaan arvokasta tietoa sen käytettävyydestä, audiovisuaalisista elementeistä, yhteensopivuudesta ja suorituskyvystä, joita ei automatisoiduilla testeillä voida saavuttaa. [22.]

Erityisesti mobiilipelien osalta viimeisimmän testausvaiheen aikana sovelluksesta tehdään Alfa- ja Beta-versiot, joita testataan oikeiden

testikäyttäjien avulla. Betatesteihin kutsutaan kehitystiimin ulkopuolisia käyttäjiä, kuten toimittajia taikka blogin kirjoittajia, joiden palautteen avulla saadaan tehostettua julkaistavan tuotteen markkinointia ja korjattua viimeisimpiä virheitä pelin lähdekoodissa. [23.]

Automaattisen testauksen vahvuutena on kustannustehokas jatkuva prosessi, joka avustaa usein toistuvien tapausten testaamisessa isoissa ohjelmistoprojekteissa. Useampia testitapauksia voidaan suorittaa samanaikaisesti automaatiotyökalujen avulla. Automaatiotyökalujen avulla voidaan simuloida valtavien käyttäjämäärien aiheuttamaa rasitusta sovelluksen käytössä, jonka toteuttaminen manuaalisesti on huomattavasti vaikeampaa. Automaation käyttäminen helpottaa virheiden etsimistä erityisesti sovelluksen kehitysvaiheessa. [22.]

3.2 Funktionaalinen testaus

Funktionaaliset testausmenetelmät varmistavat sovelluksen perustoimintojen täsmällistä toteutumista. Funktionaaliset testausmenetelmät rajoittuvat ainoastaan järjestelmän toiminnallisuuksien testaamiseen, jota ovat esimerkiksi vuorovaikutus eri komponenttien välillä, tietojen käsittely, käyttäjien vuorovaikutus ja järjestelmän reaktio erilaisiin olosuhteisiin. Funktionaalinen testaus pitää sisällään yksikkötestauksen, regressiotestauksen, sanity-testauksen, savutestauksen sekä integrointitestauksen. [25.]

Mobiilisovelluksen käyttäjä ei epäröi poistaa sovellusta laitteeltaan, mikäli sen käytössä ilmenee ongelmia. Funktionaalisten testien tärkeimpiin testattaviin osa-alueisiin kuuluu sovelluksen käynnistyminen, käyttäjän kirjautuminen, tekstialueet, nappulat, ostot ja push-ilmoitukset. [34.]

3.2.1 Yksikkötestaus

Yksikkötestauksen rooli mobiilisovelluksen testauksessa on varmistaa lähdekoodin virheettömyys ajoissa ja usein. Virheiden havaitseminen ajoissa

laskee testaamisen kustannuksia ja helpottaa korkeamman testaustason testien suorittamista. Yksikkötestaus tekee lähdekoodista helpommin ymmärrettävää, muutettavaa ja toimii osana dokumentaatiota. [26.]

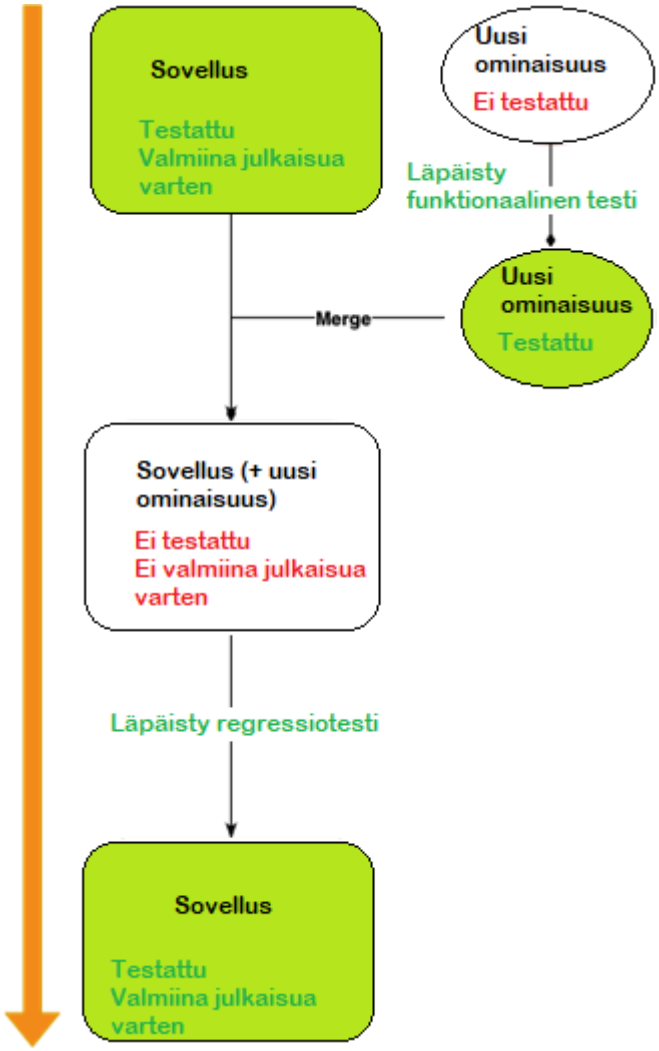
Yksikkötestaus koostuu sarjasta toistettavia väittämiä, jotka testaavat yksikön tuottaman tulosteen oikeellisuuden. Yksikkötestausta voidaan suorittaa manuaalisin taikka automaattisin keinoin, mutta automaattiset testaustyökalut ovat kustannustehokkaampia yksikkötestauksessa. Testiskriptit ovat uudelleenkäytettäviä työkaluja, joita voidaan käyttää toistuvasti uudelleen säästäen testaajan ylimääräiseltä manuaaliselta työltä. [27.]

Yksikkötestauksen vaikutus lopulliseen mobiilisovellukseen on erittäin tärkeä, koska se kustannustehokas tapa paljastaa matalan tason vikoja, joita ei laajan projektin lähdekoodista ole helppoa silmämääräisesti havaita. Yksikkötestaus myös mahdollistaa korkeampien testaustasojen testaamisen perustusten ollessa kunnossa.

3.2.2 Regressiotestaus

Regressiotestauksen avulla uudet toiminnallisuudet testataan ennen niiden liittämistä osaksi sovellusta. Tarkoituksena on etsiä uuden ominaisuuden aiheuttamia virhetilanteita osana sovellusta. Läpäisty regressiotesti toimii tarkistuspisteenä sovelluksen versiohistoriassa mihin voi aina palata, mikäli uusi ominaisuus aiheuttaa virheitä sovelluksen arkkitehtuurissa. [28.]

Kuvassa 7 näkyy regressiotestauksen koko rakenne. Alussa näkyy kokonaan testattu sovellus, josta irrotetaan uusi haara uuden ominaisuuden rakentamista varten. Uusi ominaisuus testataan funktionaalisin testein, jonka jälkeen se lisätään osaksi koko sovelluksen rakennetta. Regressiotestauksen avulla varmistetaan, että vanhan ja uuden yhdistämisestä ei seuraa virheitä.



Kuva 7. Regressiotestaus [28 muokattu alkuperäisestä].

Yksikkötestauksen tapaan regressiotestaus on olennainen osa sovelluksen kehitystyön sujuvuutta ketterän sovelluskehityksen projekteissa. Uusia toimintoja lisätään sovelluksen osaksi jatkuvalla syötöllä, joten regressiotestaus suoritetaan jokaisen iteraation päätteeksi taaten ohjelmiston toimintavarmuuden seuraavaa iteraatiota varten. Testitapausten kierrättäminen tilanteen tarpeen mukaan tulee tehdä säännöllisesti, jotta regressiotestaus säilyy tarkoituksenmukaisena. [29.]

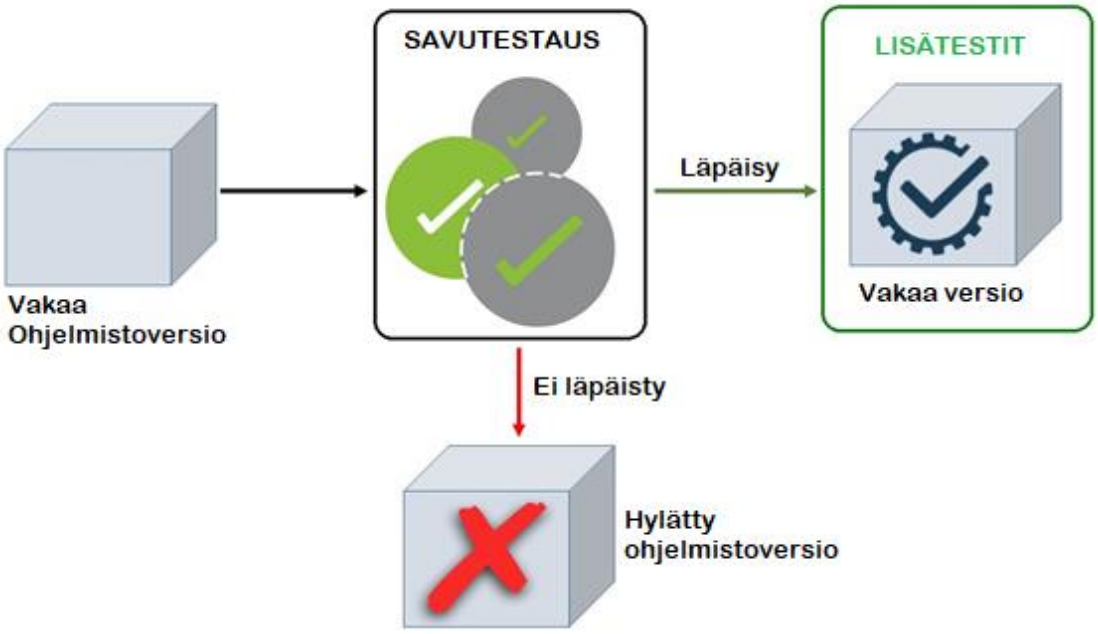
V-mallinen testaus hyödyntää regressiotestausta osana integrointitestausta. Integrointitestauksessa ohjelmiston osa-alueet liitetään yksi kerrallaan

testattavaan kokonaisuuteen, jolloin regressiotestauksen sisältämät testitapaukset varmistavat sujuvan integroinnin. [30.]

3.2.3 Savutestaus

Savutestauksen avulla on tarkoitus varmistaa ohjelmistoversion vakaus ennen siirtymistä tarkempiin testauksen vaiheisiin. Tämän testausmenetelmän avulla sovelluksen kriittiset osa-alueet ja niiden toiminta voidaan varmistaa ennen siirtymistä testauksen tasolta toiselle. Epävakaan ohjelmistoversion testaaminen on resurssien tuhlaamista, joten savutestaus on tärkeä osa-alue kustannustehokkuuden säilyttämiseksi. [32.]

Savutestauksessa pääpaino on suorittaa toistuva testitapausten sarja, joka tuottaa kuvan 8 mukaisen vakaan ohjelmistoversion ennen sen siirtämistä testaustiimille.

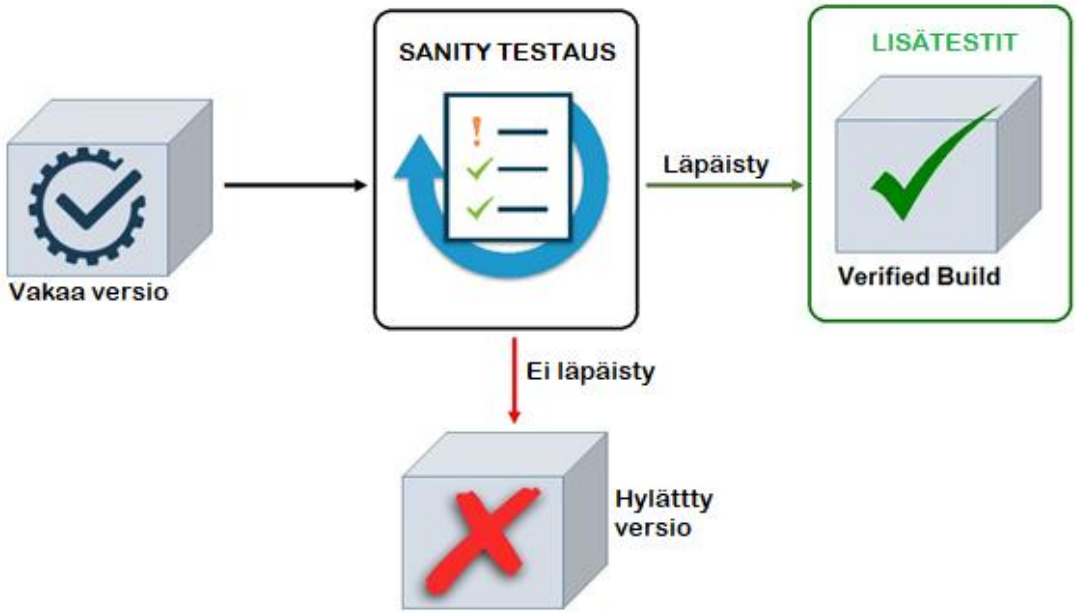


Kuva 8. Savutestaus [32 muokattu alkuperäisestä].

3.2.4 Sanity-testaus

Sanity-testaus on regressiotestaukseen kuuluva osa-alue, joka testaa vakaan ohjelmistoversion virheiden varalta pienten ja tarkkaan rajattujen testitapausten avulla. Sanity-testaus suoritetaan tyypillisesti savutestauksen jälkeisenä vaiheena testauksen kiertokulkua. [31.]

Sanity-testauksen testaamat toiminnot katsastetaan ja vahvistetaan, jonka lopputuloksena saavutetaan kuvan 9 mukainen toiminnallisuuksiltaan vahvistettu ohjelmistoversio.



Kuva 9. Sanity-testaus [31 muokattu alkuperäisestä].

3.2.5 Integrointitestaus

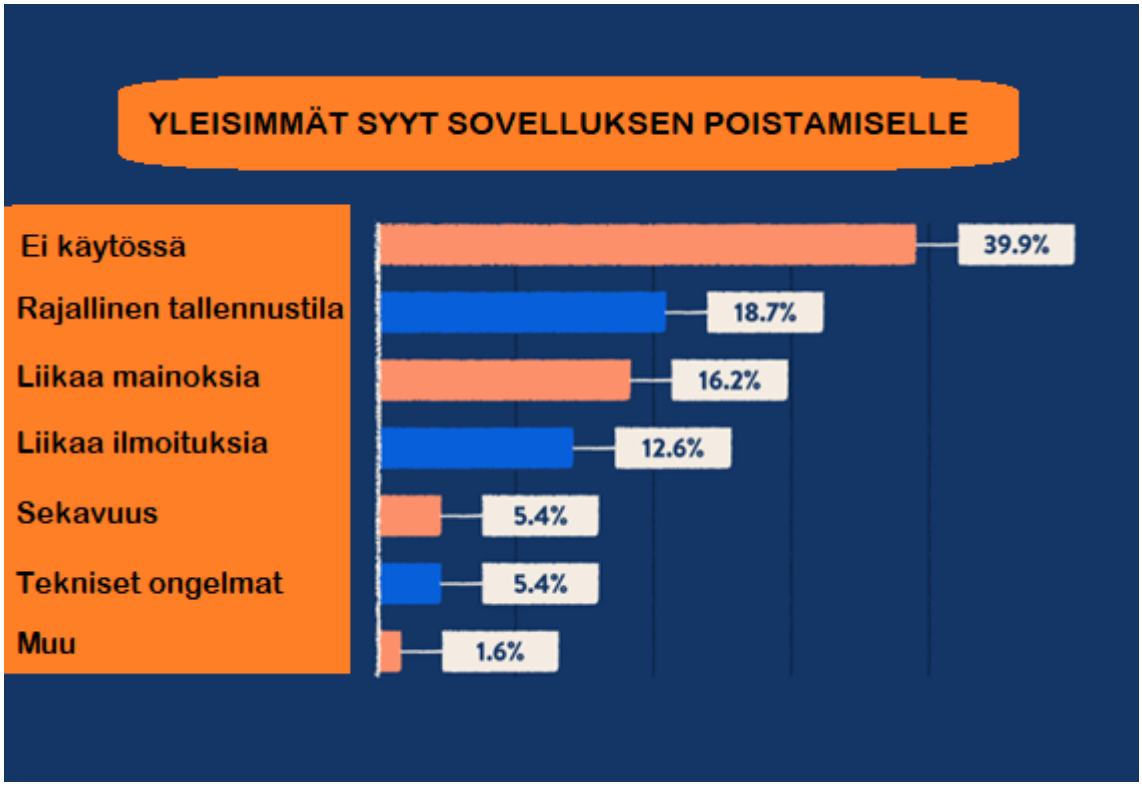
Funktionaalisen integrointitestauksen merkitys julkaistavan sovelluksen laatuun on merkittävä. Integrointitestaus tarjoaa oikein tehtynä laaja-alaisen kuvan sovelluksen ongelmakohdista, ja helpottaa niiden ratkaisemista aikaisessa vaiheessa ennen kuin niiden korjaaminen muuttuu vaikeammaksi ja kalliimmaksi. Integrointitestaus edellyttää kehitystiimin sekä testaustiimin välistä

tehokasta kommunikaatiota. Tehokas kommunikaatio tiimien välillä tarjoaa hyötyä sovelluksen testaukseen, kehitykseen, skaalautuvuuteen, riskienhallintaan ja ylläpitoon. [33.]

3.3 Epäfunktionaaliset testausmenetelmät

Mobiilisovelluksen testausstrategian tärkeimpiä kulmakiviä ovat muun muassa käytettävyys, tietoturvallisuus ja suorituskyky, joita voidaan testata epäfunktionaalisilla testausmenetelmillä.

Kuvan 10 osoittaman tilaston mukaisesti mobiilisovellusten käyttäjät poistavat sovelluksen laitteeltaan hyvin usein johtuen huonosta käyttökokemuksesta. Sovelluksen poistamiseen johtuvista syistä ainakin liiallinen mainosten määrä 16,2 %, liialliset ilmoitukset 12,6 % ja sekavuus 5,4 % voidaan suoraan laskea käytettävyyden puutteellisuuteen. Tekniset viat ja liiallinen muistikapasiteetin vieminen mobiililaitteella voivat johtua yhteensopivuuden, tietoturvan tai suorituskyvyn ongelmista. Epäfunktionaaliset testausmenetelmät ovat näin ollen suuressa roolissa nostaen sovelluksen käyttöastetta ja käyttäjien säilymistä asiakkaina.



Kuva 10. Syyt mobiilisovelluksen poistamiselle laitteelta [46 muokattu alkuperäisestä].

3.3.1 Käytettävyystestaus

Käytettävyystestaus tarjoaa testaustiimille todellisen koekäyttäjän mielipiteen sovelluksen käyttämisestä. Käytettävyystesti voidaan suorittaa kasvotusten testiryhmän ja koehenkilöiden kesken tai täysin etänä. Kasvotusten järjestettyihin käytettävyystesteihin osallistuvat testin järjestäjä, koehenkilöt sekä muut mahdolliset sidosryhmäläiset, jotka voivat tehdä koetilanteeseen liittyviä havaintoja. Etänä suoritetut käytettävyystestit ovat jakautuneet ohjattuihin tai itsenäisesti tehtäviin testeihin. Kasvotusten suoritettavat käytettävyystestit voivat tehdä vaikutuksen projektin johtoportaaseen, mutta se on usein kohtuuttoman kallis toimenpide saavutettuun hyötyyn nähden. [35.]

Ohjattu käytettävyystestaus etäyhteyden avulla tarjoaa kustannustehokkaan tavan analysoida käytettävyyttä koekäyttäjän laatiman kirjallisen raportin ja videotallenteella näkyvien reaktioiden perusteella. Videotallenteen, raportin ja

testitapausten onnistumisprosentin perusteella testaustiimi voi purkaa ohjatun käytettävyystestauksen tulokset muille sidosryhmille jälkikäteen ja nopeuttaa tällä tavalla tuotteen julkaisemiseen käytettyä aikaa. [35.]

Ohjaamaton testaus on edullisin tapa toteuttaa käytettävyystestausta sovellukselle ja se voi samalla toimia sovelluksen markkinointikeinona. Suljettuun betatestaukseen kutsutut sosiaalisen median vaikuttajat sekä blogin kirjoittavat voivat muodostaa sovellukselle kiinnostuneen käyttäjäkunnan, mikäli sen käyttökokemus on sujuvalla tasolla. [36.]

3.3.2 Suorituskykytestaus

Mobiilisovelluksen suorituskykytestaus pitää sisällään monta osa-aluetta, jotka varmistavat sovelluksen toimintavarmuuden erilaisissa tilanteissa. Suorituskykytestauksen avulla pyritään havaitsemaan pullonkauloja ja tarkkailla sovelluksen suorituskykyä verrattuna suoritusindikaattoreihin. Mobiilisovelluksen suoritusindikaattoreiksi valitaan usein latausaika, virheprosentti, vasteaika, suurin määrä pyyntöjä, käyttäjien interaktiivisuus julkaisun jälkeen, animaatioiden latausajat ja videokuvan laatu laitekokoonpanojen ja käyttöjärjestelmien välillä. [38.]

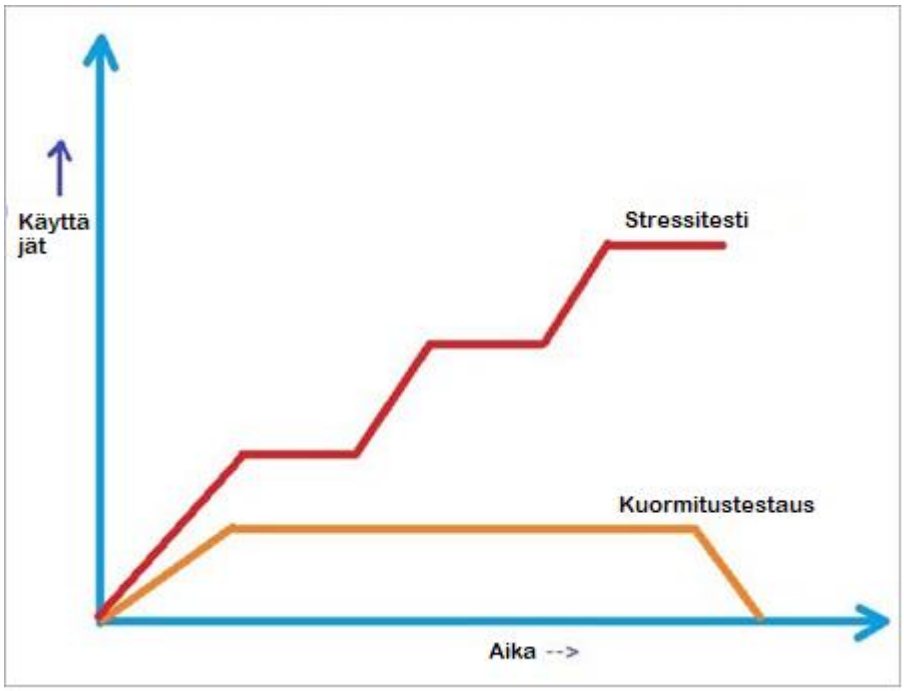
Suorituskykytestauksessa kiinnitetään huomiota laitteen, palvelimen sekä API-rajapintojen ja verkon suorituskykyyn. Laitekohtainen suorituskykytestaus antaa tietoa sovelluksen käynnistämiseen käytettävästä ajasta, virrankulutuksesta ja muistinkulutuksesta. Palvelinkohtainen testaus kertoo tiedonkulun tehokkuudesta ja oikeellisuudesta palvelimen ja sovelluksen välillä. Tiedon kulku API-rajapintojen läpi ulkoisiin sovelluksiin ja takaisin voi aiheuttaa sovelluksen hidastumista, joten siitä syystä pyyntöjen vasteaika on erinomainen mittari tämän testaamiseen. Sovelluksen käyttö vaihtelevissa verkko-olosuhteissa tulee varmistaa osana suorituskykytestausta. Sovelluksen käyttäjän liikkuessa sovellusta käyttäessään, mobiiliverkon yhteyden laatu saattaa vaihtua kesken käytön ja tämän vuoksi suorituskykyä tulee testata eri nopeuksisilla yhteyksillä. [37.]

Kuormitustestaus

Kuormitustestaus on suorituskykytestauksen osa-alue, jonka tavoitteena on altistaa sovellus raskaalle kuormitukselle ja tarkkailla sovelluksen vakautta kuormituksen alaisena. Kuormitustestauksen aikana selvitetään sovelluksen vasteaikoja, tiedonsiirtonopeutta, resurssien käyttöä ja samanaikaisten käyttäjien määrän maksimiarvoa. Kuormitustestaus edesauttaa sovelluksen toimintojen toimintavarmuutta ja auttaa määrittelemään optimaalisen kapasiteetin, jota sovellus tarvitsee toimiakseen. [38.]

Stressitestaus

Stressitestaus toimii saman periaatteen mukaan kuin kuormitustestaus, mutta kuvan 11 mukaisesti kuormaa kasvatetaan yli sovelluksen oletettujen rajojen, jolloin voidaan tehdä havaintoja raja-arvoista, jolloin sovellus hidastuu ja lopulta kaatuu. Stressitestauksen yhteydessä voidaan varmistaa, että kuinka monella eri tavalla sovellus voi lakata toimimasta, ja että kaikki asianmukaiset ilmoitukset käyttäjälle ovat paikallaan eri häiriötilanteissa. [39.]



Kuva 11. Stressitestaus ja kuormitustestaus [39 muokattu alkuperäisestä].

3.3.3 Yhteensopivuustestaus

Sovelluksen yhteensopivuustestauksen tavoite on saada sovellus toimimaan virheettömästi mahdollisimman monella eri laitekokoonpanolla. Yhteensopivuustestaus on tärkeä osa mobiilisovelluksen testausta, koska mobiililaitteiden kirjo on hyvin laaja näytön koon, käyttöjärjestelmän ja komponenttien osalta. Riittävän laajalla yhteensopivuustestauksella sovelluksen sujuva käyttö saadaan varmistettua kohdennetulle ryhmälle laitteita.

Yhteensopivuustestauksen yhteydessä tarkastellaan sovellukseen kuuluvia elementtejä, joiden toiminta vaikuttaa käyttökokemukseen. Yleisimmät tarkasteltavat elementit ovat valikot, nappulat, navigaatio, sisällön sijainti ruudulla, muotoilun tasalaatuisuus, asennuksen ja päivittämisen toimivuus sekä taulukoiden näkyvyys. Testatut laitekokoonpanot voidaan jakaa osittain yhteensopiviin sekä täysin yhteensopiviin. Testaamalla osittain yhteensopivia laitekokoonpanoja testaustiimi voi varmistaa, että sovelluksen toiminnan kannalta kriittisimmät toiminnallisuudet toimivat vaikka laitekokoonpano ei kaikilta osin ole yhteensopiva. [40.]

Yhteensopivuustestaus testaa sovelluksen yhteensopivuutta joko eteenpäin tai taaksepäin. Yhteensopivuustestaus taaksepäin on tarkoitettu varmistamaan vanhemman laitteen yhteensopivuutta sovelluksen kanssa tasalaatuisen käyttökokemuksen takaamiseksi. Yhteensopivuustestaus eteenpäin keskittyy uuden sovellusversion yhteensopivuuden varmistamiseen. Yhteensopivuustestaus sekä eteenpäin että taaksepäin pitää sisällään eri parametreja, joita testataan erikseen, joita ovat kolmansien osapuolten sovellukset, laitteet, laitteiden komponentit, verkkoyhteydet, selaimet, sovelluksen vanhemmat versiot, käyttöjärjestelmät ja eri älylaitteet. [41.]

Mahdollisimman kattava yhteensopivuuden testaaminen laajentaa mobiilisovelluksen käyttäjäkuntaa, koska sovellus saadaan toimimaan mahdollisimman monella mobiililaitteella.

3.3.4 Turvallisuustestaus

Sovelluksen tietoturvan testaaminen toimii samalla tavoin kuin työpöytäsovelluksen turvallisuuden testaaminen. Tietoturva-aukot paikallistetaan ja niiden aiheuttama potentiaalinen uhka eliminoidaan. Turvallisuusstandardien noudattaminen on äärimmäisen tärkeää, etteivät asiakkaiden henkilötiedot, käyttäjätunnukset, salasanat tai maksutiedot joudu vääriin käsiin. Turvallisuustestaus suoritetaan kahdessa osassa, jotka ovat haavoittuvuuksien arviointi sekä penetraatiotestaus. Sovelluksen infrastruktuuri ja suojamekanismit arvioidaan haavoittuvuuksien varalta, jonka jälkeen löydettyjä haavoittuvuuksia pyritään hyödyntämään penetraatiotestauksen keinoin. Penetraatiotestaus tehdään löydettyjen haavoittuvuuksien pohjalta arvioiden niiden laajuutta ja vakavuutta. [42.]

Mobiilisovelluksen turvallisuusriskit alkavat jo sovelluskaupassa. Sovelluskauppa saattaa pitää kirjaa käyttäjäkohtaisista tiedoista, jotka voivat huonosti suojattuna johtaa tietovuotoon. Sovelluskaupan turvallisuusriskit ovat huomattavasti yleisempiä avoimeen lähdekoodiin pohjautuvassa Android sovellusten sovelluskaupassa verrattuna IOS-laitteiden sovelluskauppaan, jossa julkaistava sovellus menee tarkemman seulontaprosessin lävitse. Sovelluskauppaan liittyvien riskien lisäksi turvallisuustestauksessa kiinnitetään huomiota API-rajapintojen suojaustasoon tiedon liikkuessa sovelluksesta toiseen tämän pullonkaulan lävitse. [42.]

Mobiilisovelluksen heikosti hoidettu tietoturvan taso voi olla kohtalokas isku yrityksen maineelle ja usein myös yksi osasyy miksi potentiaalinen käyttäjä jättää sovelluksen lataamatta laitteelleen. Tästä syystä tietoturvatestaus on mobiilisovelluksen menestykselle välttämätön toimenpide varsinkin, jos sovellus käsittelee käyttäjille arkaluontoisia henkilötietoja. [43.]

3.3.5 Energiatehokkuuden testaaminen

Mobiilisovelluksen energiatehokkuutta voidaan mitata laitteistopohjaisilla- tai ohjelmistopohjaisilla työkaluilla. Laitteistopohjaiset sekä ohjelmistopohjaiset työkalut jakautuvat hienorakeisiin tai karkearakeisiin mittareihin. Hienorakeiset työkalut mittaavat yksittäisten komponenttien, kuten prosessorin, muistin, GPS-paikannuksen ja sensorien virrankulutusta. Karkearakeiset mittalaitteet mittaavat koko sovelluksen virrankulutusta valittujen toimintojen toteutuessa. Laitteistopohjaiset työkalut ovat kalliimpia ja vaikeampia käyttää, mutta niillä saa tarkemmat ja luotettavammat mittaustulokset. [44.]

Mobiilisovelluksen energiatehokkuuden testaaminen ja optimointi johtaa ekologisempaan sovellukseen ja parantaa käyttäjien käyttökokemusta vähentäen mobiililaitteen energian kokonaiskulutusta.

3.4 Tekoäly testauksessa

Tekoäly yhdistettynä olemassa oleviin automaatiotyökaluihin tuottaa uudenlaisen tehokkaamman tavan testata mobiilisovelluksia. Mobiililaitteiden monimuotoisuuden haasteita on entistä tehokkaampi ratkoa yhdistämällä tekoälysovellus pilvipohjaisiin testauskehyksiin. Pilvipohjaiset testaustyökalut paljastavat virheitä lähdekoodissa testaajan laatiman tekstisyötteen pohjalta. Koodin virheiden lisäksi automatisoitu tekoälytestaus pystyy testaamaan saavutettavuutta, tietoturvaa, suorituskykyä, kuormituskykyä, käyttöeleitä ja alustojen välistä toimintaa. Testitapausten kehittäminen on suoraviivaisempaa, koska testaajien tulee vain kuvailla tapaus tekstimuodossa sen sijaan että laatisivat jokaisen testin koodimuodossa erikseen. Tekoälyn käyttäminen testauksessa vapauttaa testaajille enemmän aikaa keskittyä tärkeimpiin kehityskohteisiin sovelluksessa, joka parantaa lopullisen mobiilisovelluksen lopputulosta. [5.]

Mobiililaitteiden käytettävyystestauksessa ei ole vielä tähän mennessä voinut luottaa automaatioon, koska ainoastaan oikeat testikäyttäjät ovat voineet

varmistaa, että visuaaliset elementit ja sovelluksen sujuva käyttö toteutuvat. Tähän löytyy ratkaisu visuaalisesta tekoälystä. Mobiililaitteiden testauksessa visuaalinen tekoäly voi tarkistaa käyttöliittymään kohdistuvat muutokset satojen virtuaalisten laitekokoonpanojen osalta, jolloin testauksen prosessi tehostuu. [45.]

4 Yhteenveto

Opinnäytetyön tavoitteena oli kartoittaa ja tutkia mobiilisovellusten testausmenetelmiä ja niiden vaikutuksia julkaistavaan tuotteeseen. Tutkimuksen tuloksena saatiin selville, että mobiilisovellusten testaamisessa hyödynnetään paljon samoja tekniikoita kuin työpöytäsovelluksien testaamisessa. Mobiililaitteiden monimuotoisuus aiheuttaa suurimman eron mobiilisovellusten ja työpöytäsovellusten testaamisessa. Mobiililaitteiden testaaminen eri laitekokoonpanoilla, jotka sisältävät variaatioita eri laitevalmistajien, käyttöjärjestelmien, ruudun kokojen, resoluutioiden, järjestelmäversioiden ja komponenttien välillä, on yksi suurimmista haasteista mobiilisovellusten testaamisessa. Laajojen laitekokoonpanojen kattavuus testitilanteessa voidaan lähes poikkeuksetta saavuttaa ainoastaan hyödyntämällä automatisoituja testausmenetelmiä. Tekoälyn hyödyntäminen osana mobiilisovelluksen testausprosessia osoittautui tehokkaaksi tavaksi minimoida manuaalisia työvaiheita.

Lähdeaineistona käytetyt blogikirjoitukset olivat ajoittain ristiriidassa keskenään eri testausmenetelmien käsitteistön osalta. Ristiriitaisten tietojen luotettavuuden takaamiseksi tietoa tuli hakea riittävän monesta lähteestä. Työssä onnistuttiin vertaamaan kriittisesti eri lähdeaineistojen tietoja keskenään luotettavan tiedon suodattamiseksi.

Sulava käyttöliittymä, suorituskyky, alhainen virrankulutus ja aukoton tietoturva ovat menestyksekkään sovelluksen laatustandardeja, joita on mahdoton saavuttaa ilman asiaan kuuluvia testausmenetelmiä. Mobiilisovelluksien maksavat asiakkaat poistavat sovelluksen hyvin herkästi, mikäli laatustandardit eivät ole kunnossa. Mobiilisovelluksen kehittäjät ja testaajat hyötyvät käyttämällä tässä työssä esiteltyjä testausmenetelmiä. Tämän työn jatkokehitys onnistuu tarkastelemalla mobiilisovelluksia käyttötarkoituksen valossa. Sovelluksen käyttötarkoituksen perusteella testausmenetelmät on mahdollista painottaa tarkemmin ja asettaa enemmän resursseja kriittisimpiin menetelmiin.

Lähteet

[1] Laborde, S., “50 landing page statistics to know in 2023,” 2023. Saatavilla: https://techreport.com/statistics/landing-page-statistics/. [Haettu 3.10.2023]

[2] Testsigma, ”Beginner’s guide to Mobile App Testing,” N.d. Saatavilla https://testsigma.com/mobile-testing. [Haettu 3.10.2023]

[3] UTOR, “Software Testing Strategies for Software Development: How to Make the Right Choice,” 2021. Saatavilla: https://u-tor.com/topic/software-testing-strategies-for-software-development. [Haettu 3.10.2023]

[4] Jabbarvand. R., Malek, S., "Advancing Energy Testing of Mobile Applications," 2017 IEEE/ACM 39th International Conference on Software Engineering Companion (ICSE-C), Buenos Aires, Argentina, 2017, pp. 491-492, doi: 10.1109/ICSE-C.2017.45. Saatavilla: https://ieeexplore.ieee.org/document/7965399 [Haettu 3.10.2023]

[5] ACCELQ, “How AI is Revolutionizing Mobile Test Automation,” 2023. Saatavilla: https://www.accelq.com/blog/how-ai-is-revolutionizing-mobile-test-automation/. [Haettu 3.10.2023]

[6] Bharati, N., ”How to test Native App vs Hybrid App vs Web App vs Progressive Web App (PWA)” 2022. Saatavilla: https://www.browserstack.com/guide/test-native-vs-hybrid-vs-web-vs-progressive-web-app [Haettu 3.10.2023]

[7] BrowserStack, “Testing for Fragmentation: Understanding Browser, OS and Device Fragmentation,” 2019. Saatavilla: https://www.browserstack.com/blog/understanding-browser-os-and-device-fragmentation/ [Haettu 3.10.2023]

[8] GeeksforGeeks, “Software Engineering | SDLC V-Model,” 2023. Saatavilla: https://www.geeksforgeeks.org/software-engineering-sdlc-v-model/. [Haettu 3.10.2023]

[9] Atlassian, “What is the Agile methodology?,” N.d. Saatavilla: https://www.atlassian.com/agile [Haettu 3.10.2023]

[10] Arguelles, C., ” Resolving the Hybrid vs. Native Apps Dilemma in 2023,” N.d. Saatavilla: https://appetiser.com.au/blog/hybrid-vs-native-apps/ [Haettu 3.10.2023]

[11] Thorpe, D., “Mobile Device Fragmentation – What it is and why should you care,” N.d. Saatavilla: https://indiespring.com/appinsights/mobile-device-fragmentation-what-it-is-and-why-should-you-care/ [Haettu 3.10.2023]

[12] GeeksforGeeks, “Unit Testing | Software Testing,” 2023. Saatavilla: https://www.geeksforgeeks.org/unit-testing-software-testing/ [Haettu 3.10.2023]

[13] GeeksforGeeks, “Software Engineering | Integration Testing,” 2023. Saatavilla: https://www.geeksforgeeks.org/software-engineering-integration-testing/ [Haettu 3.10.2023]

[14] GeeksforGeeks, “System Testing,” 2023. Saatavilla: https://www.geeksforgeeks.org/system-testing/ [Haettu 3.10.2023]

[15] GeeksforGeeks, “Acceptance Testing | Software Testing,” 2022. Saatavilla: https://www.geeksforgeeks.org/acceptance-testing-software-testing/?ref=gcse/ [Haettu 3.10.2023]

[16] Hamilton, T., ” What is Agile Testing? Process & Life Cycle,” 2023. Saatavilla: https://www.guru99.com/agile-testing-a-beginner-s-guide.html [Haettu 3.10.2023]

[17] Rajora, H., ” Guide to Designing an Effective Mobile App Testing Strategy,” 2021. Saatavilla: https://www.lambdatest.com/blog/mobile-app-testing-strategy/ [Haettu 3.10.2023]

[18] Statcounter, “Mobile Operating System Market Share Finland Sept 2022 - Sept 2023,”. Saatavilla: https://gs.statcounter.com/os-market-share/mobile/finland [Haettu 3.10.2023]

[19] Emizentech, “How to Find the Target Audience for A Mobile App?,” 2022. Saatavilla: https://www.emizentech.com/blog/how-to-find-the-target-audience-for-a-mobile-app.html [Haettu 3.10.2023]

[20] Statcounter, “Mobile & Tablet Android Version Market Share Finland Sept 2022 - Sept 2023,”. Saatavilla: https://gs.statcounter.com/android-version-market-share/mobile-tablet/finland [Haettu 3.10.2023]

[21] Mojsoski, B., ”How to Create a Device Matrix for Mobile Application Testing,” 2021. Saatavilla: https://www.testdevlab.com/blog/how-to-create-a-device-matrix-for-mobile-application-testing [Haettu 3.10.2023]

[22] Yadav, P., ” Mobile Testing Basics: Manual Vs Automated Testing,” 2021. Saatavilla: https://dzone.com/articles/mobile-testing-basics-manual-vs-automated-testing [Haettu 3.10.2023]

[23] Khomych, A., ” Mobile Game Marketing: 7 Best Strategies to Succeed in 2023,” 2023. Saatavilla: https://blog.getsocial.im/mobile-game-marketing-7-best-strategies-to-succeed/ [Haettu 3.10.2023]

[24] Tramontana, P., Amalfitano, D., Amatucci, N. et al. Automated functional testing of mobile applications: a systematic mapping study. Software Qual J 27, 149–201 (2019) Saatavilla: https://link.springer.com/article/10.1007/s11219-018-9418-6 [Haettu 3.10.2023]

[25] Bose, S., ”Functional Testing : A Detailed Guide,” 2023. Saatavilla: https://www.browserstack.com/guide/functional-testing [Haettu 3.10.2023]

[26] Nabil, M., ” What is unit testing in mobile development?,” 2022. Saatavilla: https://bitrise.io/blog/post/what-is-unit-testing-in-mobile-development [Haettu 3.10.2023]

[27] Cprime, ” Benefits of Automated Testing Over Manual Testing,” N.d. Saatavilla: https://www.cprime.com/resources/blog/benefits-of-automated-testing-over-manual-testing/ [Haettu 3.10.2023]

[28] Dayan, F., ” Regression Testing in Mobile Development,” 2020. Saatavilla: https://medium.com/trendyol-tech/regression-testing-in-mobile-development-c4895f262fb9 [Haettu 3.10.2023]

[29] Testsigma, ” The Complete Guide to Regression Testing,” N.d . Saatavilla: https://testsigma.com/regression-testing/regression-testing-in-agile [Haettu 3.10.2023]

[30] Etestinghub, ” V-Model in Software Testing,” N.d. Saatavilla: https://etestinghub.com/v_model.php#2 [Haettu 3.10.2023]

[31] Felice, S., “What is Sanity Testing with examples,” 2022. Saatavilla: https://www.browserstack.com/guide/sanity-testing [Haettu 3.10.2023]

[32] Webomates, ” Smoke Testing vs Sanity Testing : A comparison,” 2020. Saatavilla: https://www.webomates.com/blog/software-testing/smoke-testing-vs-sanity-testing-a-comparison/ [Haettu 3.10.2023]

[33] Lamdatest, ” Functional Integration Testing Tutorial: A Comprehensive Guide With Examples And Best Practices,” 2023. Saatavilla: https://www.lambdatest.com/learning-hub/functional-integration-testing [Haettu 3.10.2023]

[34] Bose, S., ”Types of Mobile Testing,” 2023. Saatavilla: https://www.browserstack.com/guide/mobile-testing-types [Haettu 3.10.2023]

[35] Kalinin, K., ” Mobile App User Testing: Methods, Tools, Best Practice,” 2020. Saatavilla: https://topflightapps.com/ideas/mobile-usability-testing/ [Haettu 3.10.2023]

[36] Hrzenjak, B., ” How to use the beta testing phase for better app marketing,” 2023. Saatavilla: https://www.shakebugs.com/blog/beta-testing-in-marketing/ [Haettu 3.10.2023]

[37] Chatterjee, S., “What is Android Performance Testing?,” 2022. Saatavilla: https://www.browserstack.com/guide/android-performance-testing [Haettu 3.10.2023]

[38] Vtestcorp, ”The Ultimate Guide to Load Testing for Mobile Applications,” 2020. Saatavilla: https://www.vtestcorp.com/blog/load-testing-for-mobile-applications/ [Haettu 3.10.2023]

[39] SoftwareTestingHelp, “Stress Testing Guide For Beginners,” 2023. Saatavilla: https://www.softwaretestinghelp.com/stress-testing/ [Haettu 3.10.2023]

[40] Ghosh, A., ”Understanding Mobile Compatibility Testing: Why your application needs it,” 2022. Saatavilla: https://www.headspin.io/blog/understanding-mobile-compatibility-testing-why-your-application-needs-it [Haettu 3.10.2023]

[41] Chatterjee, S., ”What is Compatibility Testing? (Examples Included),” 2023. Saatavilla: https://www.browserstack.com/guide/compatibility-testing [Haettu 3.10.2023]

[42] Tagade, K., ” How to Perform Mobile Application Security Testing,” 2023. Saatavilla: https://www.getastra.com/blog/mobile/mobile-application-security-testing/ [Haettu 3.10.2023]

[43] Weetech solution, “Importance of Mobile App Security and Why it’s Essential to Your Business,” 2022. Saatavilla: https://www.weetechsolution.com/blog/importance-of-mobile-app-security [Haettu 3.10.2023]

[44] Khan, M., Abbas, S., Lee, S., Abbas, A., Measuring power consumption in mobile devices for energy sustainable app development: A comparative study and challenges, Sustainable Computing: Informatics and Systems, Volume 31, 2021, 100589, ISSN 2210-5379, Saatavilla: https://doi.org/10.1016/j.suscom.2021.100589. [Haettu 3.10.2023]

[45] VentureBeat, ”How visual AI can solve the challenge of native mobile app testing,” 2023. Saatavilla: https://venturebeat.com/ai/how-visual-ai-can-solve-the-challenge-of-native-mobile-app-testing/ [Haettu 3.10.2023]

[46] Karnes, K., C., ” Why Users Uninstall Apps: 28% of People Feel Spammed [Survey]” N.d. Saatavilla: https://clevertap.com/blog/uninstall-apps/ [Haettu 3.10.2023]