Katsaus mobiilisovelluksien saavutettavuuteen

Turun yliopisto
Tietotekniikan laitos
TkK-tutkielma
Tietotekniikka
Kesäkuu 2025
Juho Torkkeli

TURUN YLIOPISTO
Tietotekniikan laitos

Juho Torkkeli: Katsaus mobiilisovelluksien saavutettavuuteen

TkK-tutkielma, 26 s.
Tietotekniikka
Kesäkuu 2025

Digipalvelut ovat siirtyneet vahvasti mobiiliin: suomalaiset hoitavat pankki-, terveys- ja vapaa-ajan asiat älypuhelimella lähes ajasta ja paikasta riippumatta. Samaan aikaan EU:n esteettömyysdirektiivi (European Accessibility Act) astuu voimaan kesäkuussa 2025 ja laajentaa saavutettavuusvaatimukset myös yksityisille digipalveluille. Tutkielma kartoittaa mobiilisovellusten saavutettavuuden nykytilaa ja tarkastelee, millaisin periaattein esteettömyys voidaan integroida ohjelmistokehityksen jatkuviin käytäntöihin. Kirjallisuuskatsaus kattaa viimeisen vuosikymmenen kansainvälisen tutkimuskirjallisuuden sekä WCAG-suosituksiin perustuvat standardit EU-sääntelykontekstissa.
Tulosten perusteella mobiilisovellusten saavutettavuus on edelleen hajanaisesti toteutunut ilmiö, jossa tekniset, prosessuaaliset ja organisatoriset tekijät limittyvät. Havaituissa ratkaisumalleissa korostuvat automatisoitu testaaminen, asiantuntija- ja käyttäjäpohjainen auditointi, koneoppimiseen pohjautuva semanttisen metadatan tuottaminen sekä niin sanottu shift-left-ajattelu, jossa saavutettavuus otetaan huomioon jo suunnitteluvaiheessa. Tutkielma esittää kerrostetun saavutettavuusekosysteemin, joka yhdistää nämä menetelmät ja tukee organisaatioita saavutettavuuden systemaattisessa kehittämisessä. Saavutettavuusekosysteemi tarjoaa teoreettisen perustan sovelluskehityksen prosessimuutoksille aikana, jolloin EU:n esteettömyysdirektiivin toimeenpano laajentaa saavutettavuusvelvoitteet myös yksityisiin digipalveluihin.

Asiasanat: saavutettavuus, mobiilisovellus, WCAG, esteettömyysdirektiivi

Sisällys

1 Johdanto

Viime vuosien aikana älypuhelimista on tullut keskeinen osa arkipäivää, mikä on lisännyt mobiilisovellusten käyttöä. Suomalaiset hoitavat pankkiasioita, terveydenhuoltoa, opiskelua ja vapaa-ajan palveluita mobiilisovelluksilla lähes riippumatta ajasta ja paikasta. Vuonna 2024 jo 96 % 16–64-vuotiaista suomalaisista käytti älypuhelinta päivittäin [1]. Sovellusten yleistyessä niiden kohderyhmät kasvavat ja käytettävyydessä tulisi ottaa huomioon eri ihmisryhmien tarpeet, jottei osa käyttäjistä jää palveluiden ulkopuolelle.

Maailman terveysjärjestön (WHO) World Report on Disability [2] raportoi jo vuonna 2011, että yli miljardi ihmistä – noin 15 % maailman väestöstä – elää jonkinasteisen vamman kanssa. WHO:n ja UNICEF:n tuoreempi raportti vuodelta 2022 (Global Report on Assistive Technology) [3] arvioi, että 2,5 miljardia ihmistä tarvitsee vähintään yhden apuvälineen tai avustavan teknologian, ja määrä voi kasvaa 3,5 miljardiin vuoteen 2050 mennessä. Nämä luvut osoittavat, että saavutettavat mobiilisovellukset eivät ole vain lisäominaisuus vaan välttämättömyys yhä suuremmalle käyttäjäkunnalle.

Euroopan tasolla saavutettavuusvaatimuksia yhdenmukaistetaan EU:n esteettömyysdirektiivillä (EAA, Direktiivi (EU 2019/882)). Direktiivi hyväksyttiin 17. huhtikuuta 2019, jäsenvaltioiden oli saatettava se osaksi kansallista lainsäädäntöä 28. kesäkuuta 2022 mennessä, ja sitä aletaan soveltaa 28. kesäkuuta 2025 alkaen [4]. EAA laajentaa saavutettavuusvelvoitteet julkisilta verkkopalveluilta (Di-

rektiivi 2016/2102/EU) myös moniin yksityisen sektorin digitaalisiin tuotteisiin ja palveluihin, esimerkiksi verkkokauppoihin, e-kirjoihin ja pankkisovelluksiin. Suomessa tämä tarkoittaa Digipalvelulain (306/2019) päivittämistä vastaamaan uusia vaatimuksia [5].

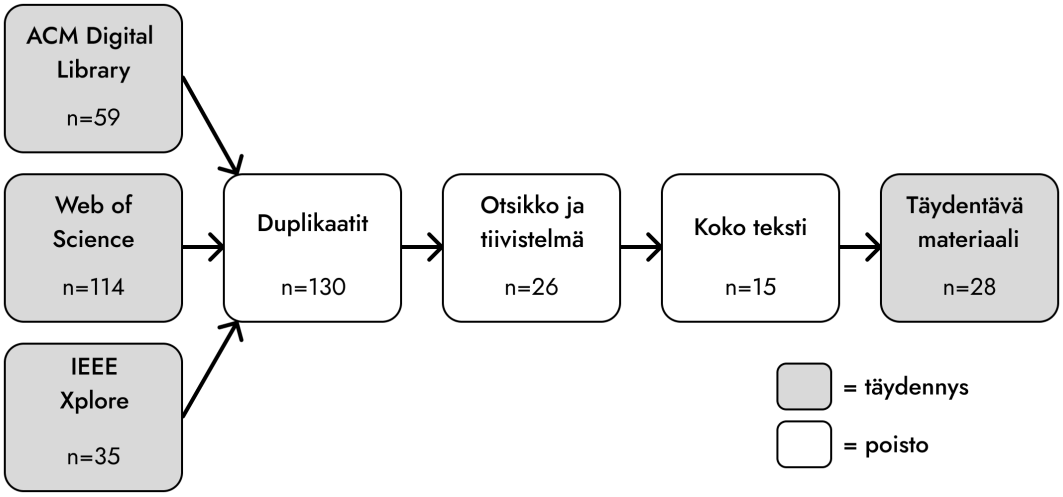
Tämä tutkielma keskittyy selvittämään, mikä on mobiilisovelluksien saavutettavuuden nykytila ja miten saavutettavuutta voidaan parantaa. Työssä tarkastellaan myös, mitä keinoja kirjallisuudessa on esitetty savutettavuuden parantamiseksi. Tutkielma pyrkii löytämään vastaukset seuraaviin tutkimuskysymyksiin:

TK1: Mikä on mobiilisovellusten saavutettavuuden nykytila?

TK2: Miten sovellusten saavutettavuutta voidaan parantaa?

Tutkielma toteutettiin kirjallisuuskatsauksena. Haku suoritettiin 5.4.2025 ja se kohdistettiin ACM Digital Library-, IEEE Xplore- sekä Web of Science -tietokantoihin. Haku rajattiin koskemaan maksimissaan 10 vuotta vanhoja julkaisuja. Kirjallisuuskatsauksen vaiheet on havainnollistettu kuvassa 1.1, lisäksi vaiheet on avattu tarkemmin omissa luvuissaan. Ennen aineiston karsimista päätettiin sisällyttämiskriteerit. Jokaisessa vaiheessa säilytettiin vain ne julkaisut, jotka

- ovat kirjoitettu suomeksi tai englanniksi
- ovat kirjoja, standardeja, tieteellisiä artikkeleita tai konferenssijulkaisuja
- ovat saatavilla ilmaiseksi verkossa tai Turun Yliopiston kirjaston kautta
- tuovat esille tai esittävät sovelluksien saavutettavuusongelmia tai ratkaisuja niihin
- käsittelevät asiaa tarpeeksi laajasti eli eivät keskity vain yhteen rajoitteeseen, maahan tai ominaisuuteen



Kuva 1.1: Tutkimusprosessin vaiheet

Hakulause muodostettiin täydentämällä tutkimuskysymyksistä **TK1** ja **TK2** poimittuja avainsanoja synonyymeillä, sekä muilla aiheisiin liittyvillä termeillä. Näiden pohjalta muodostettiin seuraavat hakulausekkeet *(mobile app* AND accessi*)* sekä *(usab* OR barrier* OR challenge* OR issue* OR evaluat* OR experienc* OR implement* OR assess* OR analys* OR compliance)*. Ensimmäistä hakulausetta haettiin vain otsikoista ja toista hakulausetta kaikesta sisällöstä. Haku suoritettiin kolmeen eri tietokantaan, joista löytyi osittain samoja julkaisuja. Duplikaatit poistettiin niin, että jäljelle jäi vain yksi versio julkaisusta. Julkaisuja karsittiin otsikon, tiivistelmän ja koko tekstin mukaan niin, että poistettiin ne julkaisut, jotka eivät täyttäneet kaikkia sisällyttämiskriteereitä. Tässä vaiheessa poistettiin myös epätäydellisestä hakulausekkeesta johtuvat virheosumat sekä julkaisut, joiden rajaus ei vastaa tutkimuskysymyksiin **TK1** ja **TK2**. Lopuksi lähdeaineistoa täydennettiin yleisellä saavutettavuutta käsittelevillä teoksilla, saavutettavuuteen liittyvillä standardeilla sekä kehityksen kustannuksiin liittyvällä kirjallisuudella.

Tutkielma koostuu johdannon, taustaluvun ja johtopäätösten lisäksi kahdesta käsittelyluvusta. Luvussa 2 määritellään saavutettavuuteen liittyvät keskeiset kä-

sitteet ja esitellään saavutettavuusohjeiden yleinen rakenne. Luvussa 3 keskitytään saavutettavuuden nykytilaan ja kirjallisuuden tuntemiin saavutettavuusongelmiin sovelluksissa. Luku 4 käsittelee puolestaan kirjallisuuden tuntemia ratkaisuja saavutettavuusongelmien korjaamiseen. Lisäksi luvussa käsitellään tunnettujen ratkaisujen rajoitteita ja esitetään näitä yhdistävä sekä täydentävä ehdotelma uudeksi ratkaisuksi. Tutkielma päättyy johtopäätöksiin, joissa vastataan tutkimuskysymyksiin sekä pohditaan saavutettavuuden suuntaa tulevaisuudessa.

2 Saavutettavuus

Saavutettavuudella (engl. *accessibility*) tarkoitetaan digitaalisen tuotteen tai palvelun kykyä olla käytettävissä mahdollisimman laajalle käyttäjäjoukolle myös silloin, kun käyttäjillä on pysyviä, tilapäisiä tai tilannekohtaisia toimintarajoitteita. WHO:n ICF-viitekehys (engl. *International Classification of Functioning*, Disability and Health) muistuttaa, että toimintakyky muodostuu yksilön, tehtävän ja ympäristön vuorovaikutuksessa, eikä pelkästään henkilön ominaisuuksista [6]. Saavutettavuus eroaa käytettävyydestä (engl. *usability*): ISO 9241-11 määrittelee käytettävyyden tehokkuuden, tuloksellisuuden ja tyytyväisyyden näkökulmasta, kun taas saavutettavuus keskittyy esteiden poistamiseen, jotta palvelu olisi käytettävissä myös erilaisilla aisti-, motorisilla ja kognitiivisilla rajoitteilla [7].

2.1 Verkkosisällön saavutettavuusohjeet (WCAG)

Verkkosisällön saavutettavuusohjeet (engl. *Web Content Accessibility Guidelines*) on World Wide Web Consortiumin (W3C) kehittämä ja ylläpitämä ohjeistus, jonka tavoitteena on varmistaa, että mahdollisimman moni käyttäjä ja avustava teknologia pystyy hyödyntämään verkkosisältöä. Ohjeiston ensimmäinen versio julkaistiin jo vuonna 1999, ja uusin virallinen versio WCAG 2.2 hyväksyttiin lokakuussa 2023. W3C kehittää ohjeistusta jatkuvasti: ensimmäinen julkinen luonnos seuraavan sukupolven WCAG 3.0:sta julkaistiin tammikuussa 2021, ja uusin työluonnos on päivätty toukokuulle 2024 [8].

WCAG rakentuu tasoittain. Ylimpänä on neljä periaatetta (POUR) – havaittava (engl. *perceivable*), hallittava (engl. *operable*), ymmärrettävä (engl. *understandable*), toimintavarma (engl. *robust*). Periaatteet sisältävät yhteensä 13 ohjetta, joille on määritetty testattavia onnistumiskriteerejä [9]. Kriteerit on jaettu kolmeen hierarkkiseen vaatimustasoon:

- A: Vähimmäisvaatimus, kuten tekstivastineet ja näppäimistökäyttö,
- AA: Laajalle levinnyt tavoitetaso, esimerkiksi kontrastisuhde $\geq$4,5:1 ja sivuston kielen määrittely
- AAA: Korkein tavoitetaso, kuten laajennettu kontrasti ja viittomakielinen tulkkaus.

Tasot ovat kumulatiivisia: AA-vaatimusten täyttäminen edellyttää myös kaikkien A-kriteerien täyttämistä, ja AAA edellyttää sekä A- että AA-kriteerejä. WCAG tarjoaa näin selkeän ja mitattavan viitekehyksen, jonka avulla saavutettavuutta voidaan arvioida ja parantaa järjestelmällisesti [8]. Tässä tutkimuksessa aineistoa tarkastellaan WCAG 2.1 kautta, koska se on aineistossa laajimmin käytetty kriteeristö.

2.2 EU:n esteettömyysdirektiivi ja digipalvelulaki

Euroopan unionin esteettömyysdirektiivi eli European Accessibility Act (EAA) astui voimaan huhtikuussa 2019 ja velvoittaa jäsenvaltiot huolehtimaan siitä, että tietyt kuluttajille tarjottavat tuotteet ja palvelut ovat saavutettavia [4]. Jäsenmaiden tuli saattaa direktiivi osaksi kansallista lainsäädäntöään 28.6.2022 mennessä, ja vaatimusten soveltaminen alkaa pääosin 28.6.2025. Direktiivin tarkoituksena on edistää kaikkien – erityisesti vammaisten henkilöiden – mahdollisuuksia itsenäiseen digitaaliseen asiointiin YK:n vammaisyleissopimuksen tavoitteiden mukaisesti. EAA koskee muun muassa tele- ja viestintäpalveluja, liikkumiseen liittyviä verkkosivustoja

ja mobiilisovelluksia (kuten matkalippujen ostaminen ja reaaliaikaiset matkustustiedot), kuluttajapankkipalveluja, sähköisiä kirjoja ja niiden lukusovelluksia sekä verkkokauppaa. Mikäli tuote tai palvelu kuuluu direktiivin piiriin, sen on täytettävä määritellyt esteettömyysvaatimukset, ellei kyseessä ole mikroyritys, jolle myönnetään rajattu poikkeus.

Suomessa esteettömyysdirektiivin velvoitteet on sisällytetty lakiin digitaalisten palvelujen tarjoamisesta (306/2019, ns. digipalvelulaki), jonka muutokset tulivat voimaan 1.2.2023 [5]. Julkisen sektorin lisäksi laki laajentaa saavutettavuusvaatimukset yksityisiin toimijoihin aina, kun nämä tarjoavat direktiivin alaan kuuluvia digitaalisia tuotteita tai palveluja kuluttajille. Valvontaviranomaisena toimii Suomessa Liikenne- ja viestintävirasto Traficom, jonka tehtävänä on myös käsitellä käyttäjien saavutettavuuspalautteet ja valitukset.

Teknisenä viitekehyksenä sekä EU:ssa että Suomessa hyödynnetään standardia EN 301 549, joka esittää mitattavat vaatimukset tieto- ja viestintätekniikan tuotteille, palveluille ja järjestelmille [10]. Standardin noudattaminen luo niin sanotun olettaman vaatimustenmukaisuudesta; esimerkiksi verkkosivustojen ja mobiilisovellusten kohdalla se viittaa suoraan WCAG 2.1:n A- ja AA-tasojen onnistumiskriteereihin. EAA, kansallinen digipalvelulaki ja EN 301 549 muodostavat kokonaisuuden, joka ohjaa sekä julkisia että yksityisiä toimijoita suunnittelemaan ja hankkimaan saavutettavia digitaalisia palveluja.

2.3 Avustavat teknologiat mobiilialustoilla

Sekä Applen iOS-laitteet että Googlen Android-laitteet sisältävät nykyään laajan valikoiman sisäänrakennettuja avustavia teknologioita, joiden avulla laite on käytettävissä myös silloin, kun käyttäjällä on näköön, kuuloon, motoriikkaan tai kognitioon liittyviä rajoitteita. Avustavien ratkaisujen ydin on ruudunlukija – iOS:n VoiceOver ja Androidin TalkBack – joka muuntaa näytön sisällön puheeksi, tunnis-

taa eleohjauksen ja tukee pistekirjoitusnäyttöjä. Näiden lisäksi molemmat alustat tarjoavat perustoiminnot, kuten näytön suurennuksen, tekstin ja kontrastin säädöt, automaattiset tekstitykset sekä äänikomennoilla tapahtuvan laitteen ohjauksen. Nämä toiminnot on integroitu käyttöjärjestelmän asetuksiin asti: käyttäjä voi aktivoida ne heti laitteen käyttöönoton yhteydessä tai myöhemmin esteettömyysvalikosta. Jotta avustavat teknologiat toimivat oikein, tulee sovelluksissa olla tuki niille.

Applen ja Googlen ratkaisut on dokumentoitu kunkin yhtiön omissa kehittäjä- ja tukisivustoissa, ja ne tarjoavat työkalut WCAG-yhteensopivien sovellusten kehittämiselle. Abstraktilla tasolla malli on sama: ruudunlukija välittää semanttisen rakenteen äänipalautteena, visuaaliset säätimet mukauttavat näyttöä, ja vaihtoehtoiset syöttötavat (esim. kytkimet, puhe, silmän tai pään liike) mahdollistavat käyttöliittymän hallinnan ilman kosketusta. Näin mobiilialustat tarjoavat vakiotyökalut, joiden varaan sovelluskehittäjät voivat rakentaa saavutettavia ratkaisuja ilman erillisiä lisäohjelmia [11], [12].

3 Saavutettavuuden nykytila

Tässä luvussa tarkastellaan, minkälaisia saavutettavuusongelmia kirjallisuudessa on esitetty esiintyvän sovelluksissa. Saavutettavuusongelmat on luokiteltu neljään kategoriaan taulukon 3.1 mukaisesti. Tarkastelun tukena on käytetty WCAG 2.1 -standardin määrittelemiä kategorioita [9].

Taulukko 3.1: Lähdeaineistossa havaitut saavutettavuusongelmat

Lähde	1. Havaittava	2. Hallittava	3. Ymmärrettävä	4. Toimintavarma
Carvalho ym. 2018 [13]	x	x	x	
McKay ym. 2017 [14]	x	x	x	
Ross ym. 2017 [15]	x	x		
Oliveira ym. 2024 [16]	x	x	x	x
Yan ym. 2019 [17]	x	x	x	
Alajarmeh ym. 2022 [18]	x	x	x	x
Matos ym. 2023 [19]	x	x		
Eler ym. 2018 [20]	x	x		x

Taulukosta voidaan havaita, että mitä pidemmälle saavutettavuusperiaatteissa edetään, sitä vähemmän virheitä on käsitelty. Tätä saattaa osittain selittää se, että ohjeiden määrä laskee, mutta toisaalta myös se, että virheet muuttuvat teknisemmiksi, mitä pidemmälle mennään, jolloin niitä on vaikeampi havaita.

3.1 Havaittavuus

Havaittavuusperiaate edellyttää, että käyttöliittymän sisältö ja toiminnot ovat kaikkien käyttäjien aistittavissa [9]. Kuvakkeiden, painikkeiden ja kuvituskuvien tekstinvastineet muodostavat mobiilisovelluksissa merkittävän saavutettavuusaukon. Laajin empiirinen kartoitus osoitti, että 68 % suosittujen sovelluksien grafiikkaelementeistä oli vailla sisällön kuvaukseen tarkoitettua contentDescription-määritystä Androidilla tai sitä vastaavaa iOS-merkintää [18]. Puutteiden vaikutus konkretisoitui näkövammaisten laboratoriotestissä, jossa navigointiaika piteni keskimäärin 40 % verrattuna hyvin merkittyihin referenssisovelluksiin [14]. Samansuuntaisesta ongelmasta viestivät myös sovelluskauppojen käyttäjäarviot: tekstinvastineiden puutteet olivat kolmen yleisimmin raportoidun savutettavuusvalituksen joukossa [16]. Toimialakohtaista vaihtelua havaittiin: matkailusovelluksissa kuvagalleriat kärsivät pahiten puutteellisista tekstinvastineista, kun taas pankkisovelluksissa ongelma oli harvinaisempi [13], [15], [17]. Vaikka automattiset työkalut, pystyivät havainnoimaan lähes kolmanneksen tekstinvastineiden puutteista, 18 % virheistä jäi yhä tunnistamatta ilman manuaalista tarkistusta [20].

Toinen laajasti esille tullut asia on visuaalisen erottuvuuden ongelmat kuten kontrasti, fonttikoko ja elementtien värit – nämä muodostavat aineiston yleisimmän WCAG-poikkeaman, sillä 42-48 % käyttöliittymäelementeistä alitti AA-tason kontrastivaatimuksen eri otoksissa [17], [18]. Käytännön testissä virhemäärä kontrastivaatimuksen osalta kasvoi 60 prosenttiin, kirkkaassa valaistuksessa yhdistelmänä pieni fontti sekä heikko kontrasti [14]. Kontrastirikkomuksia oli helpompi havaita automaattitesteissä ja jopa 91 % virheistä löydettiin [20]. Palautteen perusteella 8 % käyttäjistä raportoi tumman tilan puutteesta [16].

Kuvien tekstinvastineet eivät ole ainut multimedian kuvaamiseen liittyvä ongelma, vaan ongelma on laajempi, sillä tekstitykset, ääniraidat ja transkriptiot ovat mobiiliympäristössä harvinaisia. Laajassa otoksessa vain 12 % sovelluksista tarjo-

si minkäänlaisen tekstityksen tai audiokuvauksen [18]. Kuulovammaiset opiskelijat suoriutuivat e-oppimistehtävissään 25 % heikommin sovelluksissa, joissa ei ollut tekstitystä saatavilla [19], ja lukivaikeuksista kärsiviä hyödyttävä hidastus–tekstitys -yhdistelmä osoitti vastaavasti virheiden kaksinkertaistumisen sen puuttuessa [14]. Vaikka aikaisemmin mainitut ongelmat olisivat kunnossa, rakenne ja semanttisen merkkaamisen puute rajoittaa näytönlukijan tehokasta käyttöä. Sovelluksista 77 % ei sisältänyt selkeää otsikkohierarkiaa tai sivujen päälohkoja ei ollut merkitty semanttisesti oikein [15], mikä mahdollistaisi ruudunlukijankäyttäjille mahdollisuuden siirtyä suoraan pääsisältöön tai navigaatioon. ARIA-roolit olivat kunnossa vain noin viidenneksessä tutkittuja elementtejä [13]. Lisäksi pakotettu pystyorientaatio rikkoi WCAG 1.3.4 -onnistumiskriteeriä kuudessa suositussa pankkisovelluksessa [20].

3.2 Hallittavavuus

Hallittavuusperiaate edellyttää, että käyttöliittymän osat ovat käytettävissä ja toimivat myös muilla tavoilla kuin hiirellä tai kosketuksella eli esimerkiksi näppäimistöllä. Lisäksi käyttäjän on voitava hallita käyttöliittymän ajoitusta, navigointia ja syöttötapoja [9]. Fokusjärjestyksen, eli käyttöliittymäelementtien valintajärjestyksen pirstoutuminen on mobiilialustoilla huomattavaa: näytönlukijan fokus eteni eri järjestyksessä kuin visuaalinen lukujärjestys 72 % tarkastelluista näkymistä [13]. Otsikkorakenteiden eli loogisen ja hierarkkisen otsikoinnin (WCAG 2.4.6) rikkomukset olivat kaksinkertaisesti yleisempiä kuin luvussa 3.1 tarkastellut vaihtoehtoisten tekstien puutteet (WCAG 1.1.1) [20]. Mediaohjainten puutteet (WCAG 2.2.2) näkyvät Rossin analyysissä, jossa 47 % video- ja äänikomponenteista ei tarjonnut pysäytys- ja toistotoimintoja [15].

Bluetooth-näppäimistöstä tai kytkinohjauksesta (engl. *switch access*) riippuvaiset käyttäjät kohtaavat merkittäviä esteitä. Suurimman kartoituksen mukaan 65 % painikkeista ei ollut fokusoitavissa fyysisellä näppäimistöllä [18]. Laboratoriossa kyt-

kinohjausryhmä suoriutui tehtävistään 30 % hitaammin kuin kontrolliryhmä, kun näppäintukea ei ollut implementoitu [14]. Myös automatisoidut auditoinnit paljastavat fokuksen hallinnan puutteita: 18 % Android- ja 15 % iOS-sovelluksista sisälsi ns. näppäimistöansan eli tilanteen, jossa käyttäjä jää jumiin elementtiin ilman mahdollisuutta poistua näppäimistöllä. (WCAG 2.1.2) [17]. Lisäksi vain 40 % tarkastelluista sovelluksista tarjosi käyttäjän määriteltävissä olevia pikanäppäimiä (WCAG 2.1.4), vaikka testihenkilöt pitivät niitä yhtenä tärkeimmistä tehokkuustekijöistä [18]. Huomattavaa oli myös, että erityisesti näytönlukijan käyttäjiä helpottava suoraan pääsisältöön ohjaava ohituslinkki löytyi vain 18 % mobiilisovelluksista [17], mikä hidastaa ruudunlukijakäyttäjien sisällönhakua. Tulokset osoittavat, että näppäimistöriippuvaisten käyttäjien esteet ovat sekä systemaattisia että alustariippumattomia.

Fokusjärjestyksen puutteet ja ongelmat eivät ole ainoita, joita aiheuttavat käyttäjille haasteita. Aineistossa korostuu kosketus- ja eleintaraktiopohjaisten käyttöliittymien suosion kasvu, joihin siirtyminen kaventaa näppäimistöpohjaisen ohjauksen sekä avustusteknologioiden tukea. Elepainotteiset käyttöliittymät jättävät usein keskeisiä toimintoja ilman näppäimistövastaavuutta [16], [18], [19]. Kahden tai useamman sormen eleille puuttui näppäin- tai valikkovastine 60 % sovelluksista (WCAG 2.5.1) [17], [18]. Lisäksi eleisiin kuten: pyyhkäisy, nipistys, vedä ja pudota tukeutuminen sulkee pois suuren joukon motorisesti rajoittuneita käyttäjiä. Matkailusovelluksissa peräti 44 % kriittisistä toiminnoista oli toteutettu ainoastaan eleillä [17]. Eleiden lisäksi anturipohjaiset komennot (WCAG 2.5.4) kuten laitteen kallistus, aiheuttavat ongelmia motorisesti rajoittuneille käyttäjille. Useissa pelisovelluksissa havaittiin, ettei vaihtoehtoista syöttötapaa anturipohjaisille komennoille ollut tarjolla [15]. Myös pelkästään kohteiden painaminen voi tuottaa ongelmia käyttäjille, sillä 44 % analysoiduista interaktiivisista elementeistä oli kooltaan alle suositellun 44 × 44 CSS-pikselin [16]. Tulokset viittaavat siihen, että eleiden ja kosketuskohteiden suunnittelu on merkittävä yksittäinen kehityskohde hallittavuuden parantamiseksi.

Huonoa hallittavuutta ja vaikeita interaktioita pahentaa sovellusten aikarajoitukset. Nämä osoittautuvat toistuvaksi saavutettavuusesteeksi. Viidessä pankkisovelluksessa automaattinen uloskirjautuminen käynnistyi alle 60 sekunnissa, rikkoen WCAG-vaatimuksen mahdollisuudesta jatkaa istuntoa ilman tietojen menetystä [18]. Näkövammaisten käyttäjien suoritusaika oli lisäksi keskimäärin kaksinkertainen verrattuna näkeviin, kun tehtävät katkesivat selityksettömiin aikakatkaisuihin. Vain 10 % palveluista salli aikarajan säätämisen [13], [19]. 17 % tutkimuksessa analysoiduista sovelluksista sisälsi mainoksia, jotka ylittivät WCAG:n 3 Hz vilkuntakynnyksen [17]. VR/AR-oppimisympäristöissä vastaavaa välkettä raportoitiin 26 % kohtauksista [19]. Välkkyvät mainosbannerit ja liian nopeat animaatiot voivat laukaista epileptisiä kohtauksia tai aiheuttaa pahoinvointia.

3.3 Ymmärrettävyys

Ymmärrettävyysperiaate edellyttää, että sisällön ja käyttöliittymän on oltava käyttäjälle ymmärrettäviä, mikä tarkoittaa selkeää kieltä, ennakoitavaa rakennetta ja apua syötteiden täyttämiseen. Aineistossa korostui kognitiivisen kuorman vähentämisen tarve. Pitkät, termipainotteiset ohjeet osoittautuivat erityisen ongelmallisiksi näkö- ja kognitiorajoitteisille käyttäjille [18]. Kielen eksplisiittisen määrittämisen havaittiin olevan puutteellista, sillä 41 % tutkittujen sovellusten pääsisällöstä oli ilman `localeConfig-/xml:lang`-merkintää (Android), jolloin näytönlukija turvautui tulkkauksessa varavaihtoehtoon [18]. Lisäksi useissa artikkeleissa kritisoitiin lyhenteiden ja termien selittämättä jättämistä. Esimerkiksi ”APR” ja ”BMI” jäivät talous- ja terveyssovelluksissa määrittelemättä, mikä rikkoo WCAG:n ohjetta 3.1.4 [18]. Ylimääräinen erikoissanasto myös lisäsi virhelukuja kenttätesteissä [16]. Luettavuusongelmat johtuvat yhtä lailla metatiedon puutteesta kuin vaikeasti hahmotettavasta sanastosta. Näytönlukija ja käyttäjä kohtaavat esteen jo ensimmäisellä ruudulla, ellei sisällön kieltä määritellä ja tekstejä yksinkertaisteta.

Käyttöliittymien ennakoitavuus rikkoutuu, kun fokus tai navigointielementit muuttuvat odottamatta. Carvalho, Dias, Reis ym. tutkimuksessa löydettiin 38 %:sta näkymiä automaattinen fokushyppy, jossa elementti vaihtuu ilman käyttäjän toimintaa [13]. Lisäksi alapalkin navigointijärjestys vaihtui ruudulta toiselle 71 % sovelluksista, mikä heikensi käyttäjien mentaalista karttaa sovelluksesta [18]. Yllättävät kontekstimuutokset, kuten ulkoisen linkin aukaiseminen ilman varoitusta, rikkoivat WCAG 3.2.2 -onnistumiskriteeriä. Alajarmeh osoitti, että ruudunlukijan fokus katosi selaimen ulkopuolelle siirryttäessä [18]. McKay havaitsi vastaavan ongelman hybridisovellusten WebView-kerroksissa [14].

Virheilmoitusten saavutettavuus osoittautui aineistossa heikoksi osa-alueeksi. Huomattiin, että 43 % testatuista lomakekentistä palautti virheen ainoastaan värikorostuksena [17], [18]. Virheilmoituksissa havaittiin erityisesti lomakkeiden yhteydessä puutteita: 57 % lomakkeista antoi vain geneerisen ilmoituksen virheellisestä syötteestä ilman kenttäkohtaista tarkennusta [18]. Oppimissovelluksen kirjautumistestissä kolme peräkkäistä epäonnistunutta yritystä johti 35 % käyttäjistä keskeyttämään prosessin [19]. Kuvapohjainen CAPTCHA ilman äänivaihtoehtoa esiintyi 9 % sovelluksista, estäen näytönlukijan käyttäjiä etenemästä [17].Vastaavasti Yan osoitti, että syötteen automaattinen täydentäminen, kuten osoitetiedon automaattinen ehdotus, parantaa sekä selaimen omia että ruudunlukijan virhe-ehdotuksia [17].

3.4 Toimintavarmuus

Toimintavarmuusperiaate velvoittaa, että käyttöliittymä ja sen koodi ovat teknisesti eheät ja semanttisesti kuvailtu niin, että avustavat teknologiat voivat tulkita ne luotettavasti. Osa toimintavarmuuden ohjeista tuntuu menevän osittain aiempien periaatteiden ohjeiden kanssa päällekkäin, joten voi olla vaikea tunnistaa kumman periaatteen alle virhe kuuluu. Teknisiä virheitä on havaittu erityisesti Android-asettelutiedostoissa; duplikaatti-ID:t tai sulkemattomat elementit esiintyivät 9 %:ssa

aineiston näytteistä, mikä aiheuttaa muun muassa fokuksen harhailua Androidin TalkBack-ruudunlukijajärjestelmässä [20]. Kaikista alakriteereistä (4.1.2: nimi, rooli, arvo) nousee aineistossa selkeimmin esiin. Alajarmeh raportoi sen kuuluvan mobiilin ”korkean esiintyvyyden” ongelmiin. Esimerkiksi ikonipainikkeista puuttuva ”accessible name” tai vastaava attribuutti estää niiden käytön ruudunlukuohjelmalla [18]. Saman puutteen takia toimimattomia painikkeita, jotka eivät reagoi käyttäjän toimintaan, esiintyy usein hybridisovelluksissa [16].

Usein myös sovelluksen tila ja virheet jäävät näytönlukijoilta piiloon. Latausprosessin edistymisen (engl. *progress bar*) päivittyminen jäi ilmoittamatta 68 % skenaarioista, mikä johti vääriin päätelmiin sovelluksen ”jumittumisesta”. Lisäksi Alajarmeh osoittaa, että dynaamisia ilmoituksia ei yleensä välitetä ruudunlukuohjelmalle; virheilmoitus saattaa vilahtaa näytöllä sekunnin ja kadota, jättäen ruudunlukijan hiljaiseksi [18]. Oliveira paikallistaa käyttäjäpalauteaineistossa toistuvia valituksia liian nopeasti katoavista ilmoituksista, mutta huomauttaa, ettei mikään analysoiduista työkaluista havaitse näitä ongelmia [16]. Sovelluskaupan palautteissa mitään ei tapahdu -tyyppiset viestipuutteet muodostivat 7 % saavutettavuusvalituksista [16].

4 Ratkaisut

Tässä luvussa tarkastellaan, minkälaisia ratkaisuja kirjallisuudessa on esitetty saavutettavuusongelmien ratkaisemiseksi. Ratkaisuilla viitataan kirjallisuudessa ehdotettuihin menetelmiin, työkaluihin ja toimintamalleihin, joilla mobiilisovellusten saavutettavuusongelmia pyritään ehkäisemään tai korjaamaan. Nämä löydökset on ryhmitelty taulukossa 4.1 neljään ylätason kategoriaan: (1) automaattisiin testausratkaisuihin, (2) manuaalisiin arviointi- ja käyttäjätestausmenetelmiin, (3) koneoppimista hyödyntäviin lähestymistapoihin sekä (4) prosesseja parantaviin käytäntöihin.

Taulukko 4.1: Lähdeaineiston ratkaisut

Lähde	Automaattinen testaus	Manuaalinen testaus	Koneoppimisen hyödyntäminen	Prosessien parantaminen
Eler ym. 2018 [20]	x	x		x
Kashif ym. 2022 [21]	x			x
Matos ym. 2023 [19]	x			x
Mehralian ym. 2024 [22]	x	x	x	x
Salehnamadi ym. 2022 [23]	x	x		
Salehnamadi ym. 2023 [24]				
Xin ym. 2023 [25]			x	
Zhang ym. 2021 [26]			x	
Zhong ym. 2025 [27]	x	x	x	

Huomionarvoista on, että ratkaisuihin painottuvat lähteet ovat keskimäärin tuoreempia kuin ongelmia kartoittavat tutkimukset (vrt. taulukko 3.1); tämä kuvastaa kentän nopeaa kehitystä ja erityisesti tekoälypohjaisten työkalujen kasvavaa roolia saavutettavuuden edistämisessä.

4.1 Automaattinen testaus

Automaattisen saavutettavuustestauksen tutkimuskenttä on selvästi kehittymässä pelkistä sääntöpohjaisista skannereista kohti dynaamisia vuorovaikutusanalyysejä ja tekoälyavusteisia korjausratkaisuja. Varhaisemmat menetelmät keskittyivät staattiseen koodin tai käyttöliittymäkuvien tarkastukseen, mutta viime vuosien työkalut kykenevät jo simuloimaan käyttäjän kulkua sovelluksessa ja tuottamaan konkreettisia korjausehdotuksia. Staattinen analyysi säilyttää silti arvonsa etenkin suunnitteluvaiheessa. Kashif ym. esittelivät Figma-liitännäisen, joka tarkastaa UI-mallit WCAG-sääntöjä vasten, mikä mahdollistaa virheiden löytämisen, jo prototyyppivaiheessa. Heidän laskelmiensa mukaan liitännäisen käyttö lyhensi kehitysaikaa 30 % [21].

Dynaamisen testauksen tarve nousi esiin useissa tutkimuksissa. MATE-järjestelmä ajaa sovelluksia emulaattorissa ja tarkkailee samalla visuaalisia saavutettavuussääntöjä; vertaillessa 73 sovellukseen se löysi yli 47 % enemmän virheitä kuin pelkkä staattinen analyysi [20]. Groundhog puolestaan käy sovelluksen käyttöliittymän läpi (engl. *crawl*) kahdesti – ruudunlukija päällä ja pois päältä – ja liittää raporttiinsa toistovideon, mikä vahvistaa havaintojen todennettavuutta ja vähentää väärien positiivisten aiheuttamaa ylimääräistä kuormaa kehittäjille[23]. Uusimpana ScreenAudit yhdistää heuristisen ruudunlukijasimulaation sekä staattiset tarkistukset ja raportoi keskimäärin 1,8-kertaisesti kriittisiä virheitä verrattuna perinteisiin skannereihin [27].

4.2 Manuaalinen testaus

Vaikka automaattiset skannerit paikantavat nopeasti selkeitä virheitä, manuaaliset asiantuntija-arvioinnit löytävät keskimäärin 40 % enemmän ongelmia, mutta vievät samalla noin kolminkertaisen ajan analysoitua ruutua kohden [20]. Useat tutkimukset korostavatkin manuaalisen testauksen ja automaation täydentävää luonnetta: automatisoitu analyysi toimii ”ensimmäisenä suodattimena”, jonka jälkeen ihminen varmistaa tulosten oikeellisuuden ja täydentää havaitut puutteet [22], [23].

Yhteistyön eduista nousi esille myös, että ruudunkaappausvideoiden ruutu-ruudulta-analyysi yhdessä asiantuntijan kanssa paljastaa elepolkuja, joita DOM-pohjaiset tarkastajat eivät tavoita, täten yhdistäen kuvantunnistuksen ja manuaalisen käytettävyystestauksen vahvuudet [27]. Aineisto nostaa esiin myös merkittävän mobiilisovelluksia koskettavan haasteen: WCAG-ohjeet eivät vielä kata kaikkia natiivien sovellusten eleitä tai orientaatiota, mikä pakottaa testaajat turvautumaan omiin, paikallisesti sovellettuihin menetelmiin [19]. Seurauksena on arvioinnin subjektiivistuminen ja tulosten heikko vertailtavuus organisaatioiden välillä. Standardien harmonisointi onkin tutkimusten mukaan kriittistä, jotta koko kenttä voi ottaa yhtenäisen askeleen eteenpäin.

4.3 Koneoppimisen hyödyntäminen

Sovellusten saavutettavuus on jo pitkään nojannut siihen, että kehittäjät lisäävät käyttöliittymäkomponentteihin vaihtoehtoisia tekstejä, selitteitä ja muita metatietoja käsin. Viime vuosina on kuitenkin nähty tapoja automatisoida tätä prosessia, sekä korjata virheitä, jo valmiissa sovelluksissa. Varhaisin kattava toteutus on näytöntunnistussovellus, joka tunnistaa käyttöliittymäelementit pelkistä ruutukaappauksista konvoluutioneuroverkon avulla ja rakentaa elementti-, ryhmä- ja tilatason hierarkian suoraan visuaalisesta datasta, sekä tuottaa tämän pohjalta niille semanttiset

tunnisteet VoiceOver-näytönlukijalle. Malli koulutettiin 77 637 ruudulla – kattaen yli neljä tuhatta iOS-sovellusta – ja alustavassa käyttäjäkokeessa yhdeksän ruudunlukijakäyttäjää pystyi navigoimaan aiemmin esteettömyydeltään vajavaisia sovelluksia [26]. Samankaltaista, mutta ajonaikaista otetta edustaa View Type Detector, joka analysoi Android-laitteen piirtopuskuria ja osaa erottaa semanttisesti samannäköiset, mutta toiminnaltaan erilaiset kontrollit, kuten nappi tai vipu. Menetelmällä on päästy jopa 84,5 %:n luokittelutarkkuuteen ja se havaitsee myös dynaamiset tilat, kuten kontrollit, jotka ovat poissa käytöstä (engl. *disabled*), ja toimii sovelluksissa, joiden lähdekoodiin ei ole pääsyä [25]. Metadatan automaattinen tuottaminen UI-komponenteille toimii erittäin hyvin silloin, kun sovellusten lähdekoodiin ei voida lisätä saavutettavuusmerkintöjä käsin – esimerkiksi suljetun lähdekoodin sovelluksissa tai laajasti kolmannen osapuolen kirjastoja hyödyntävissä sovelluksissa.

Tuorein kehitysaskel on ScreenAudit-sovellus, jossa tietokoneen näkö yhdistyy kielimalliin. UI-solmuja ryhmitellään korkean tason heuristiikalla, jonka avulla ennustetaan puuttuvia tekstivastineita. LLM-malli tuottaa selite-ehdotuksen sekä virheraportin. Kun nämä semanttiset täydennykset liitetään automaattisen raportin oheen, ruudunlukijan virheiden määrä väheni 31 % verrattuna sääntöpohjaiseen skanneriin 14 sovelluksen kokeessa [27]. Merkittävää on myös siirtymä pelkästä virheen havaitsemisesta kohti automatisoitua korjausta. Mehralian ym. esittivät FixAlly-agentti-arkkitehtuurin, joka liittää perinteisten skannereiden tuottamiin raportteihin LLM-pohjaisen korjausehdotuksen sekä rivitason selityksen. Heidän tutkimuksessaan 77 % ehdotuksista kääntyi ilman ongelmia ja 70 % hyväksyttiin sellaisenaan kehittäjien toimesta [22].

4.4 Prosessien parantaminen

Menettelytapojen ja roolien standardointi on osoittautunut keskeiseksi teemaksi. EU-tasoinen monivaiheinen arviointiprosessi (suunnittelu → automaattinen analyy-

si → manuaalinen auditointi → raportointi) lisäsi viranomaisille raportoitujen havaintojen vertailtavuutta, mutta mittarit ja dokumenttipohjat vaihtelivat yhä organisaatioittain [19]. WCAG-perusteinen ”kerrosmalli” helpotti suunnittelijoiden ja testaajien välistä kommunikaatiota, mutta hajallaan oleva ohjemateriaali hidasti käyttöönottoa [21]. Tulokset korostavat, että selvät vastuut ja yhtenäiset dokumenttipohjat ovat vähintään yhtä tärkeitä kuin itse tekniset työkalut. Prosessien parantamiseen tähtäävä tutkimus viestii selvästi, että saavutettavuus on vietävä ”vasemmalle” suunnittelutaulukkoon ja juurrutettava kehityksen automaattisiin putkiin, jotta virheiden havaitseminen ja korjaaminen nopeutuvat. Kashif ym. kuvaavat ”shift-left”-lähestymistavan, jossa Figma-liitännäinen tarkistaa WCAG-säännöt jo käyttöliittymän prototyyppivaiheessa; 85 % virheistä paikannettiin ennen ensimmäistä kehittäjän tekemää koodimuutosta versionhallintaan ja kehityssykli lyheni 30 % [21]. Dynaaminen regressiotestaus liittää saavutettavuuden osaksi jatkuvaa integraatiota. Eler ym. ehdottavat rakennetta, jossa emulaattoripohjaiset saavutettavuustestit ajetaan jokaisen koonnin yhteydessä; näin löydettiin 47 % enemmän virheitä kuin pelkällä staattisella analyysillä [20]. On siis tehokkuuden kannalta hyödyllistä, että saavutettavuusvirheet pyritään havaitsemaan viimeistään koodikatselmoinneissa ja CI/CD-putkissa.

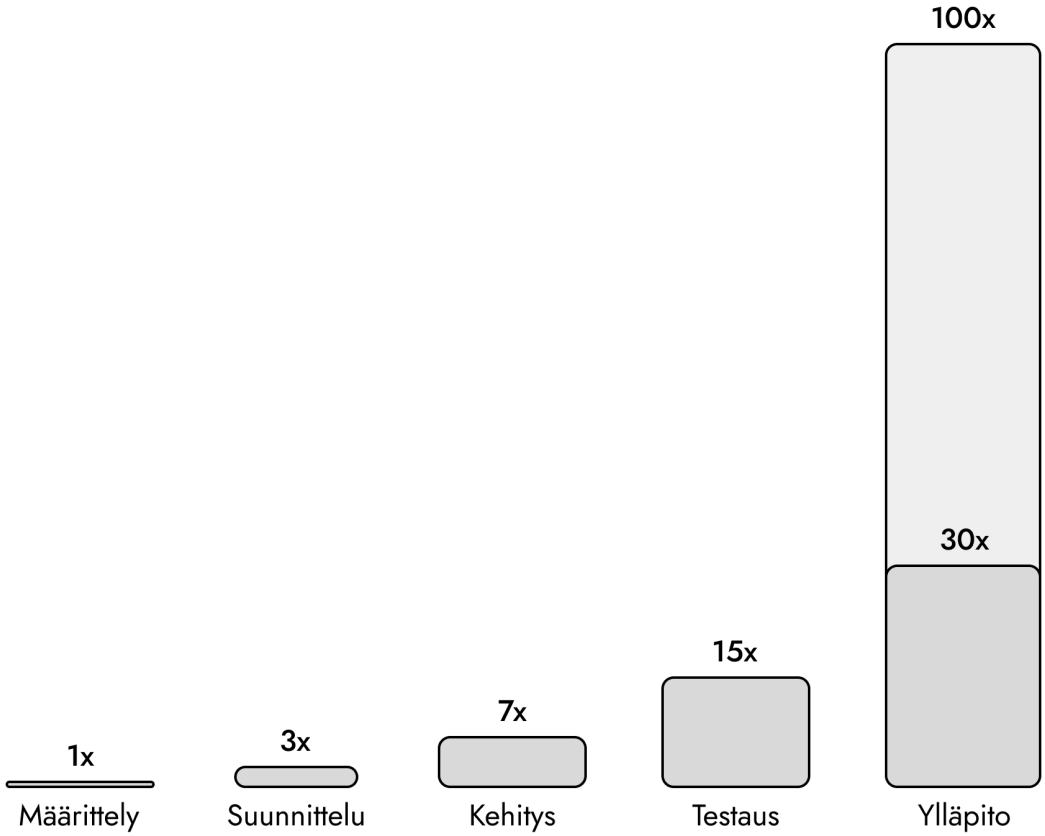
4.5 Tunnettujen ratkaisujen rajoitteet

Kirjallisuudessa on tunnistettu neljä keskeistä saavutettavuuden varmistamisen lähestymistapaa: sääntöpohjainen automaattitestaus, manuaaliset auditoinnit ja käyttäjätestit, tekoälypohjainen metadatan luonti sekä prosessien parannukset. Yksikään menetelmä ei kuitenkaan yksin takaa WCAG-yhteensopivuutta. Sääntöpohjaiset skannerit tuottavat nopean ja toistettavan analyysin, mutta kattavat vain rajallisen osan kriteereistä, kompastuvat dynaamiseen sisältöön ja räätälöityihin komponentteihin sekä aiheuttavat vääriä positiivisia ja negatiivisia havaintoja, mikä hei-

kentää luottamusta työkaluihin [19], [20], [22]. Ne eivät myöskään tunnista ääni- ja eleohjauksen, haptisen palautteen tai AR-sisällön erityispiirteitä [23]. Manuaaliset asiantuntija-auditoinnit ja käyttäjätestit tarjoavat syvällisempää tietoa, mutta ovat hitaita ja kalliita; siksi niitä käytetään harvoin ja vasta myöhäisessä kehitysvaiheessa [21], [24]. Tekoälyyn (AI, LLM) perustuva metadatan generointi lupailee vaihtoehtoistekstien ja semanttisten kuvausten automatisointia, mutta kärsii kontekstin katoamisesta, hallusinoiduista kuvauksista ja interaktiivisten sekä responsiivisten rakenteiden virhetulkinnasta [22], [25]–[27]. Kapeat harjoitusdatat synnyttävät kielellistä ja kulttuurista vinoumaa [24], ja pilvipohjainen kuvankäsittely nostaa mahdollisen huolen GDPR:n osalta [23], [25]. Prosessien parannusten tavoitteena on tuoda saavutettavuus varhaisempaan kehitysvaiheeseen, mutta ne törmäävät usein samoihin haasteisiin: KPI-mittareiden heikkoon laatuun, organisaation muutosvastarintaan sekä siihen, että CI/CD-putkiin liitetyt testit koetaan usein ylläpitokustannuksiltaan raskaiksi [20], [23], [27].

4.6 Käytettävyys osana ohjelmistokehitystä

On houkuttelevaa ajatella, että sovelluksista voisi tehdä automaattisesti saavutettavia, mutta todellisuudessa kestävä ratkaisu edellyttää monikerroksista lähestymistapaa, jossa tekniset työkalut, prosessit ja ihmiset muodostavat toisiaan vahvistavan kokonaisuuden. Tarve tällaiselle järjestelmälle ei ole vain eettinen tai lakien muodostama tarve: ohjelmistovirheiden korjauskustannukset kasvavat eksponentiaalisesti kehityksen myöhäisissä vaiheissa – NIST:n mukaan kertaluokasta toiseen, kun vika siirtyy määrittelystä tuotantoon [28]. Kuvassa 4.1 on havainnollistettu tätä eroa vaiheiden välillä. Käytettävyyden integroimista osaksi ohjelmiston tuotantoprosessia voidaan siis perustella yksinkertaisella kustannus-hyöty-logiikkalla: mitä aikaisemmin saavutettavuus otetaan työnkulkuun, sitä vähemmän se maksaa. Pe-



Kuva 4.1: Ongelmien havaitsemiseen ja korjaamiseen vaadittava työmäärä [28]

rustuen katsauksessa esiintyneisiin ratkaisumalleihin, alla esitellään viisiportainen ehdotelma saavutettavuusekosysteemistä:

Osaaminen: Saavutettavuus juurtuu organisaatioon, kun WCAG-periaatteet, digipalvelulain velvoitteet ja mobiilialustojen avustavat teknologiat sisällytetään sekä oppilaitosten opetussuunnitelmiin että työntekijöiden perehdytysohjelmiin. Tällöin peruskäsitteet ja käytännöt muuttuvat rutiininomaisiksi, mikä lyhentää virheiden korjaussyklejä ja parantaa kehitysvaiheiden läpinäkyvyyttä.

Suunnittelu: Shift-left-ajattelussa saavutettavuus otetaan huomioon jo käyttöliittymäluonnoksissa. Suunnittelutyökalujen liitännäiset ja lähdekoodin automaattinen tyylitarkastus havaitsevat kontrasti-, semantiikka- ja navigointivirheitä ennen ensimmäistä koodimuutosta versionhallintaan, mikä on osoittautunut tehokkaaksi keinoksi ehkäistä virheitä varhaisessa vaiheessa.

Automaatio: Staattinen analyysi täydentyy dynaamisilla emulaattoritesteillä, jotka suoritetaan jokaisen koonnin yhteydessä. Yhdistelmä tunnistaa huomattavasti enemmän vuorovaikutusongelmia kuin pelkkä sääntöpohjainen skannaus, ja tekoälyavusteiset työkalut voivat ehdottaa välittömiä korjauksia kriittisille löydöille.

Manuaalinen validointi: Asiantuntija-auditoinnit ja käyttäjätestit täydentävät automaatiota tuomalla esiin ele-, orientaatio- ja ilmoitusvirheitä, joita koneet eivät tavoita. Empiiriset tutkimukset osoittavat, että manuaalinen tarkastus löytää keskimäärin noin 40 % enemmän ongelmia, vaikka se vaatii enemmän resursseja analysoitua ruutua kohden.

Jatkuva oppiminen: Virhe- ja palautetiedot kootaan yhteiseen tietokantaan, jonka analytiikka tuottaa tiimille säännöllisiä parannusehdotuksia. KPI-mittaristoa tarkastellaan kvartaaleittain, ja saavutettavuusdialogi ylläpitää tietämystä standardien ja lainsäädännön muutoksista. Prosessi jalostaa näin itseään ja varmistaa, että saavutettavuus pysyy keskeisenä laatukriteerinä.

Saavutettavuuden kokonaisvaltainen ekosysteemi ei nojaa yhteen työkaluun, vaan se rakentuu saumattomasta yhteistyöstä, jossa yhdistyvät ihmisten osaaminen ja kerrostettu automaatio. Ekosysteemin arvo konkretisoituu sekä loppukäyttäjälle esteettömänä käyttökokemuksena että organisaatiolle pienempinä korjauskustannuksina. Kun koko tiimi tuntee saavutettavuusstandardit, automaatio poimii triviaalit virheet ja tekoälysovellukset täydentävät koodissa olevat aukot. Näin manuaa-

listen testien osuus voi keskittyä siihen, missä ihminen on korvaamaton: todellisen käyttökontekstin analysointiin ja ymmärtämiseen.

5 Johtopäätökset

Tämän tutkielman tutkimuskysymykset olivat *"Mikä on mobiilisovellusten saavutettavuuden nykytila?"* (**TK1**) ja *"Miten sovellusten saavutettavuutta voidaan parantaa?"* (**TK2**). **TK1**:n osalta analyysi osoitti, että puutteet ovat edelleen systemaattisia ja kohdistuvat kaikkiin WCAG:n POUR-periaatteisiin: noin kaksi kolmannesta kuvakkeista jäi ilman tekstivastinetta, ja lähes puolet käyttöliittymäelementeistä alitti AA-tason kontrastivaatimuksen. Lisäksi näppäimistötuen ja fokusjärjestyksen heikkoudet hankaloittivat avustavan teknologian käyttöä merkittävästi. **TK2**:n näkökulmasta tutkimus toi esiin neljä toisiaan täydentävää kehittämislinjaa. Ensimmäiseksi automaattinen testaus tunnistaa nopeasti toistuvat virheet ja soveltuu integroitavaksi CI/CD-putkiin. Toiseksi manuaalinen testaus, erityisesti asiantuntija-auditoinnit ja avustavaa teknologiaa hyödyntävät käyttäjätestit, täydentävät automaation katvealueita. Kolmanneksi koneoppiminen tarjoaa lupaavia työkaluja esimerkiksi semanttisen rakenteen täydennykseen. Neljänneksi prosessien jatkuva kehittäminen siirtää virheiden ehkäisyn yhä varhaisempaan suunnittelu- ja toteutusvaiheeseen, juuri kuten shift-left-ajattelu edellyttää.

Näiden havaintojen perusteella tutkielma hahmottelee luvussa 4.6 monikerroksisen ekosysteemin, jossa saavutettavuus varmistetaan yhtä aikaa ihmisten osaamisen, teknologian ja prosessien avulla. Hahmoteltu kerrosrakenteinen saavutettavuusekosysteemi on vielä konseptuaalinen, mutta tarjoaa johdonmukaisen idean, jonka vaikuttavuus on syytä todentaa empiirisesti erilaisissa kehitysympäristöissä.

Lähivuosina muutospaine kasvaa: Euroopan esteettömyysdirektiivi (engl. *European Accessibility Act*) laajentaa saavutettavuusvelvoitteet yksityisen sektorin digipalveluihin 28.6.2025 alkaen. Samalla ele-, lisätty todellisuus- ja AI-pohjaisten rajapintojen standardointi on vasta aluillaan, mikä avaa jatkotutkimukselle runsaasti tilaa.

Yhteenvetona mobiilisovellusten saavutettavuus on yhä puutteellista, mutta korjauskeinot kehittyvät nopeasti. Kestävin polku kohti esteettömiä sovelluksia on integroida saavutettavuus varhaiseen suunnitteluun ja ylläpitää sitä kerrostetulla automaatiolla, asiantuntija-arvioinnilla ja jatkuvalla oppimisella. Näin saavutettavuus muuttuu projektin loppuvaiheen tarkistuslistasta pysyväksi laatutekijäksi, joka hyödyttää sekä käyttäjiä että organisaatioiden kustannusrakennetta.

Lähdeluettelo

[1] Liikenne- ja viestintävirasto Traficom. ”Viestintäpalvelujen kuluttajatutkimus 2024”. (4. helmikuuta 2025), url: https://www.traficom.fi/fi/julkaisut/viestintapalvelujen-kuluttajatutkimus-2024 (viitattu 21. 06. 2025).

[2] World Health Organization. ”World Report on Disability”. (2011), url: https://www.who.int/teams/noncommunicable-diseases/sensory-functions-disability-and-rehabilitation/world-report-on-disability (viitattu 21. 06. 2025).

[3] World Health Organization ja UNICEF. ”Global Report on Assistive Technology”. (2022), url: https://www.who.int/publications/i/item/9789240049451 (viitattu 21. 06. 2025).

[4] European Parliament. ”EAA Directive”. (2019), url: https://eur-lex.europa.eu/legal-content/EN-FI/TXT/?uri=CELEX:32019L0882 (viitattu 16. 06. 2025).

[5] Suomen eduskunta. ”Laki digitaalisten palvelujen tarjoamisesta 306/2019”. (20. maaliskuuta 2019), url: https://finlex.fi/fi/lainsaadanto/saadoskokoelma/2019/306 (viitattu 21. 06. 2025).

[6] World Health Organization. ”International Classification of Functioning, Disability and Health (ICF)”. (2001), url: https://www.who.int/standards/classifications/international-classification-of-functioning-disability-and-health (viitattu 21. 06. 2025).

[7] International Organization for Standardization. ”Ergonomics of human-system interaction — Part 11: Usability: Definitions and concepts”, International Organization for Standardization. (2018), url: https://www.iso.org/standard/63500.html (viitattu 21. 06. 2025).

[8] World Wide Web Consortium. ”Standards-guidelines”. (6. toukokuuta 2025), url: https://www.w3.org/WAI/standards-guidelines/wcag/ (viitattu 21. 06. 2025).

[9] World Wide Web Consortium. ”Web Content Accessibility Guidelines (WCAG) 2.1 ”. (22. marraskuuta 2019), url: https://www.w3.org/Translations/WCAG21-fi (viitattu 16. 06. 2025).

[10] European Telecommunications Standards Institute. ”Accessibility requirements for ICT products and services, EN 301 549”. (maaliskuu 2021), url: https://www.etsi.org/deliver/etsi_en/301500_301599/301549/03.02.01_60/en_301549v030201p.pdf (viitattu 16. 06. 2025).

[11] Apple Inc. ”Accessibility Features”. (ei julkaisupäivää), url: https://www.apple.com/accessibility/features/ (viitattu 21. 06. 2025).

[12] Google LLC. ”Android Accessibility Overview”. (ei julkaisupäivää), url: https://support.google.com/accessibility/android/answer/6006564 (viitattu 21. 06. 2025).

[13] M. C. N. Carvalho, F. S. Dias, A. G. S. Reis ja A. P. Freire, ”Accessibility and usability problems encountered on websites and applications in mobile devices by blind and normal-vision users”, teoksessa *Proceedings of the 33rd Annual ACM Symposium on Applied Computing*, sarja SAC ’18, Pau, France: Association for Computing Machinery, 2018, s. 2022–2029, ISBN: 9781450351911. DOI: 10.1145/3167132.3167349.

[14] M. McKay, "Accessibility Challenges of Hybrid Mobile Applications", teoksessa *Universal Access in Human–Computer Interaction. Design and Development Approaches and Methods*, M. Antona ja C. Stephanidis, toim., Cham: Springer International Publishing, 2017, s. 193–208, ISBN: 978-3-319-58706-6.

[15] A. S. Ross, X. Zhang, J. Fogarty ja J. O. Wobbrock, "Epidemiology as a Framework for Large-Scale Mobile Application Accessibility Assessment", teoksessa *Proceedings of the 19th International ACM SIGACCESS Conference on Computers and Accessibility*, sarja ASSETS '17, Baltimore, Maryland, USA: Association for Computing Machinery, 2017, s. 2–11, ISBN: 9781450349260. DOI: 10.1145/3132525.3132547.

[16] A. D. A. Oliveira ja M. M. Eler, "Exploring Accessibility of Mobile Applications Through User Feedback: Insights from App Reviews in a Systematic Literature Review", teoksessa *Proceedings of the XXIII Brazilian Symposium on Human Factors in Computing Systems*, sarja IHC '24, New York, NY, USA: Association for Computing Machinery, 2024, ISBN: 9798400712241. DOI: 10.1145/3702038.3702094.

[17] S. Yan ja P. G. Ramachandran, "The Current Status of Accessibility in Mobile Apps", *ACM Trans. Access. Comput.*, vol. 12, nro 1, helmikuu 2019, ISSN: 1936-7228. DOI: 10.1145/3300176.

[18] N. Alajarmeh, "The extent of mobile accessibility coverage in WCAG 2.1: sufficiency of success criteria and appropriateness of relevant conformance levels pertaining to accessibility problems encountered by users who are visually impaired", *Universal Access in the Information Society*, vol. 21, nro 2, s. 507–532, 1. kesäkuuta 2022, ISSN: 1615-5297. DOI: 10.1007/s10209-020-00785-w.

[19] M. Matos, L. Seixas Pereira ja C. Duarte, "Evaluation of the Accessibility of Mobile Applications: Current Approaches and Challenges", teoksessa *HCI*

International 2023 – Late Breaking Papers, Q. Gao, J. Zhou, V. G. Duffy, M. Antona ja C. Stephanidis, toim., Cham: Springer Nature Switzerland, 2023, s. 352–371, ISBN: 978-3-031-48041-6.

[20] M. M. Eler, J. M. Rojas, Y. Ge ja G. Fraser, "Automated Accessibility Testing of Mobile Apps", teoksessa *2018 IEEE 11th International Conference on Software Testing, Verification and Validation (ICST)*, 2018, s. 116–126. DOI: 10.1109/ICST.2018.00021.

[21] S. Kashif, M. Ahmad, M. Shahid, M. A. Habib ja K. Rizwan, "Development of An Automated Accessibility Evaluation Plugin Tool for Mobile Applications", teoksessa *2022 3rd International Conference on Innovations in Computer Science & Software Engineering (ICONICS)*, joulukuu 2022, s. 1–10. DOI: 10.1109/ICONICS56716.2022.10100578.

[22] F. Mehralian, T. Barik, J. Nichols ja A. Swearngin. "Automated Code Fix Suggestions for Accessibility Issues in Mobile Apps". (2024), url: https://arxiv.org/pdf/2408.03827.

[23] N. Salehnamadi, F. Mehralian ja S. Malek, "Groundhog: An Automated Accessibility Crawler for Mobile Apps", teoksessa *Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering*, sarja ASE '22, Rochester, MI, USA: Association for Computing Machinery, 2023, ISBN: 9781450394758. DOI: 10.1145/3551349.3556905.

[24] N. Salehnamadi, Z. He ja S. Malek, "Assistive-Technology Aided Manual Accessibility Testing in Mobile Apps, Powered by Record-and-Replay", teoksessa *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*, sarja CHI '23, Hamburg, Germany: Association for Computing Machinery, 2023, ISBN: 9781450394215. DOI: 10.1145/3544548.3580679.

[25] T. Xin, J. Zhu, L. Wang ja X. Qin, ”Screen Recognition: Creating Accessibility Metadata for Mobile Applications using View Type Detection”, teoksessa *2023 9th International Conference on Computer and Communications (ICCC)*, ISSN: 2837-7109, joulukuu 2023, s. 1787–1793. DOI: 10.1109/ICCC59590.2023.10507590.

[26] X. Zhang, L. de Greef, A. Swearngin ym., ”Screen Recognition: Creating Accessibility Metadata for Mobile Applications from Pixels”, teoksessa *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, sarja CHI ’21, event-place: Yokohama, Japan, New York, NY, USA: Association for Computing Machinery, 2021, ISBN: 978-1-4503-8096-6. DOI: 10.1145/3411764.3445186.

[27] M. Zhong, R. Chen, X. Chen, J. Fogarty ja J. O. Wobbrock, ”ScreenAudit: Detecting Screen Reader Accessibility Errors in Mobile Apps Using Large Language Models”, sarja CHI ’25, New York, NY, USA: Association for Computing Machinery, 2025, ISBN: 979-8-4007-1394-1. DOI: 10.1145/3706598.3713797.

[28] National Institute of Standards & Technology. ”The Economic Impacts of Inadequate Infrastructure for Software Testing”. (toukokuu 2022), url: https://www.nist.gov/system/files/documents/director/planning/report02-3.pdf (viitattu 21. 06. 2025).