Automatiserade verktyg för tillgänglighetstestning av webbsidor: val, utvärdering och tillämpning för Region Skåne

CHRISTOFFER AXTEGEN
BACHELOR´S THESIS
DEPARTMENT OF ELECTRICAL AND INFORMATION TECHNOLOGY
FACULTY OF ENGINEERING | LTH | LUND UNIVERSITY

Examensarbete

Automatiserade verktyg för tillgänglighetstestning av webbsidor: val, utvärdering och tillämpning för Region Skåne

Av

Christoffer Axtegen

Department of Electrical and Information Technology
Faculty of Engineering, LTH, Lund University
SE-221 00 Lund, Sweden

Sammanfattning

Att ha en webb som är tillgänglig åt alla människor är viktigt i dagens digitala samhälle. För att designa tillgängliga webbplatser finns flera lagar, riktlinjer och rekommendationer som offentliga aktörer bör försöka efterleva. Men vad innebär dessa krav i praktiken?

Detta examensarbete, genomfört i samarbete med Region Skåne, syftar till att undersöka om automatiserade testverktyg kan användas för att kontrollera tillgängligheten på webbsidor. Ett effektivt testverktyg skulle kunna förenkla testningen av webbsidor i takt med att Region Skåne förbättrar tillgängligheten på sin webb.

För att undersöka detta har en litteraturstudie och intervjuer genomförts för att ge en insikt i behovet av tillgänglighet samt vilka regler som gäller för offentliga aktörer i Sverige. Sedan har fyra relevanta verktyg valts ut och jämförts för att se vilket verktyg som är det mest passande enligt kriterierna *kostnad, flexibilitet, användarbarhet, säkerhet* och *täckning*. Den kandidat som visades mest passande implementerades i ett JavaScript program för att analysera samtliga webbplatser på en domän.

Enligt de kriterier som undersöktes visades testverktyget Pa11y vara det mest lämpliga för användning hos en stor organisation som Region Skåne. Pa11y implementerades för att kunna användas med webcrawling för att generera rapporter om tillgänglighet på hela webbdomäner.

Examensarbetet visar att det är fullt möjligt att använda automatiserade testverktyg på intuitiva sätt för att ge organisationer flexibilitet i deras testprocess. Dessa automatiserade testverktyg är dock begränsade i sin förmåga att testa webbplatser och en stor del av testningen som behövs för att avklara de krav som finns, behövs fortfarande utföras manuellt. I slutändan är det bara människor som kan avgöra om något är tillgängligt eller inte.

Nyckelord: Tillgänglighet, WCAG, EN 301 549, DOS-lagen, AXE, WAVE, Pa11y, TAW, Automatiskt testning

Abstract

In today's digital society, accessible websites are more important than ever. To design websites that are accessible to everyone there are many laws, guidelines and recommendations that public organizations and authorities need to comply with. But how do you comply with these in Sweden, and how do you test your websites for issues with accessibility?

This thesis, in cooperation with Region Skåne aims to examine whether automated testing tools can be used to verify accessibility on websites. An effective testing tool would help Region Skåne in their testing process so that they can deliver more accessible websites.

With a literature study and a set of interviews, an examination of the need for accessibility and the relevant laws and standards have been conducted. Four different testing tools for accessibility were chosen and compared in accordance with five criteria: *cost*, *flexibility, useability, security* and *coverage*. The candidate that was most fitting was implemented in a JavaScript program to analyze all webpages in a domain.

According to the criteria, the testing tool Pa11y showed to be the most fitting tool for usage within a large organization, such as Region Skåne. Pa11y was implemented to be used in combination with webcrawling to generate reports on accessibility on full web domains.

This thesis proves that the usage of an automated testing tool can potentially help organizations in their testing process. These tools, though, are limited in their capabilities to test the full scope of functionality required by legislation. In the end a large proportion of the criteria need to be verified manually. In the end, only a human can know if something is accessible or not.

Keywords: Accessibility, WCAG, EN 301 549, DOS-lagen, Automated testning,

Förord

Jag vill rikta ett stort tack min handledare Carina Rosenqvist samt medarbetarna på Testcenter, Region Skåne för det varma välkomnandet och det stöd jag har fått under arbetets gång. Jag vill även tacka min handledare på LTH, Christin Swahn för hennes vägledning genom hela processen och för att hon sett till att jag hållit mig i fas under arbetet.

Ett stort tack till de personer som har intervjuats för detta examensarbete. Er input har varit en stor inspiration och vägledning i mitt arbete.

Innehållsförteckning

1. Inledning ... 8
1.1 Bakgrund ... 8
1.2 Syfte ... 9
1.3 Målformulering ... 9
1.4 Problemformulering ... 10
1.5 Motivering av examensarbetet ... 10
1.6 Avgränsningar ... 10
2. Teknisk bakgrund ... 12
2.1 Tillgänglighet och digital jämlikhet ... 12
2.1.1 Digital inkludering och jämlikhet ... 12
2.1.2 Hjälpmedel för personer med funktionsnedsättningar ... 13
2.1.3 Statistik ... 14
2.2 Lagar och riktlinjer för webbtillgänglighet ... 15
2.2.1 WCAG ... 15
2.2.2 EN 301 549 ... 17
2.2.3 DOS-lagen ... 18
2.2.4 Lagen om vissa produkter och tjänsters tillgänglighet ... 18
2.3 Automatiserade testverktyg för webbtillgänglighet ... 19
2.3.1 Axe-core ... 19
2.3.2 WAVE ... 20
2.3.3 TAW ... 21
2.3.4 Pa11y ... 21
3. Metod ... 23
3.1 Litteraturstudie ... 23
3.2 Intervjuer ... 24
3.3 Analys av testverktyg ... 25
3.4 Implementation ... 26
3.5 Källkritik ... 28
4. Resultat ... 31

4.1 Intervjuer 31
4.1.1 Tillgänglighet i praktiken 31
4.1.2 Inkludering 33
4.1.3 Lagar och riktlinjer 34
4.1.4 Automatiserad testning 34
4.2 Resultat från analys av testverktyg 35
4.2.1 Kostnad 36
4.2.2 Flexibilitet 37
4.2.3 Användarbarhet 40
4.2.4 Säkerhet 42
4.2.5 Täckning 44
4.2.6 Jämförelse av verktygens testresultat 46
4.3 Rekommendation av testverktyg 51
4.4 Implementation av testverktyg med webcrawling 52
4.4.1 Funktionalitet 52
4.4.2 Programstruktur och datastruktur 54
4.4.3 Begränsningar 55
5. Slutsats och diskussion 57
5.1 Utvärdering av problemformulering 57
5.2 Diskussion 58
5.2.1 Jämförelse med tidigare studier 58
5.2.2 Skillnader mellan testverktyg 59
5.2.3 Val av Pa11y 62
5.2.4 Användning hos Region Skåne 62
5.2.5 Reflektion kring implementation 63
5.2.6 Begränsningar 64
5.3 Etisk reflektion 64
5.4 Framtida utvecklingar 65
6. Terminologi 67
7. Källförteckning 68
7.1 Digitala källor 68
7.2 Fysiska källor 71

8. Appendix 72
8.1 Bilaga 1: Intervjuguide 72
8.2 Bilaga 2: Framgångskriterier med tester 74

1. Inledning

Detta avsnitt inleds med bakgrund till examensarbetet (1.1). I 1.2 introduceras projektets syfte och 1.3 och 1.4 omfattar arbetets målformulering respektive problemformulering. Avsnittet avslutas med 1.5, motivering av examensarbetet och 1.6 avgränsningar.

1.1 Bakgrund

Sedan 2018 har offentliga aktörer behövt tillgänglighetsanpassa sina webbplatser så att människor oavsett förutsättningar kan använda dem. Offentliga webbsidor måste uppfylla vissa krav, enligt en europeisk standard (EN 301 549 V3.2.1) som bygger på WCAG 2.1 vilket är en global standard för tillgänglighetsanpassning. Dessa standarder ställer bland annat krav på att all information på webbsidor ska vara tillgänglig för människor med synnedsättningar, att sidor ska vara enkla att navigera utan mus och att designen ska följa god praxis när det gäller färgval och textstorlek.

I Sverige är det Myndigheten för digital förvaltning (DIGG) som övervakar att offentliga aktörer uppfyller kraven på tillgänglighet. Om dessa krav inte uppfylls och återkommande brister konstateras i tillgänglighetsanpassningen, kan DIGG utfärda ett föreläggande om vite - ett penningbelopp som den offentliga aktören blir skyldig att betala.

Detta examensarbete görs i samarbete med Region Skåne som är Skåne läns regionkommun och ansvar för bland annat sjukvård och länstrafik inom 33 kommuner. Eftersom de är en offentlig aktör omfattas de av EN 301 549. Därmed måste alla Region Skånes webbsidor, inklusive de som hanteras av tredje part, uppfylla tillgänglighetskraven.

I Region Skånes organisationsstruktur är Testcenter ansvarigt för design och utförande av tester inom IT. Testcenter är en del av förvaltningen Digitalisering, IT och MT. Deras ansvarsområden gällande testning är till exempel journalsystem till sjukvården och liknande. Eftersom testdesign och testsupport ingår i deras uppdrag, är en fördjupad förståelse av testmetoder och automatisering av tillgänglighetstester av särskilt intresse.

Strukturen för testning av webbtillgänglighet är ett område som behöver vidareutvecklas. Det finns en efterfrågan på tydligare strategi för hur webbsidor kan testas effektivt och tillförlitligt redan under designprocessen. Att manuellt testa samtliga kriterier är tids- och resurskrävande, vilket skapar ett behov av automatiserade testmetoder för att effektivisera processen.

För testning av webbsidors tillgänglighet finns flera digitala verktyg som bland annat WAVE, Accessibility Insights och Axe. Verktyg som dessa skulle potentiellt kunna spara både tid och resurser om de på ett pålitligt sätt kan avgöra om kraven för tillgänglighetsanpassning hos en webbsida uppfylls eller ej. Dessa testverktyg är ofta specifikt utformade för att säkerställa efterlevnad WCAG:s riktlinjer och är därmed väl lämpade för att möta EU:s direktiv.

Dessa testverktyg kan potentiellt kombineras med andra program alternativt ett egenutvecklat skript för att generera fullständiga rapporter på en hel webbdomän. Processen innebär att ett verktyg först generar en rapport på en enskild sida, varefter ett program eller skript automatiskt navigerar till nästa sida för att upprepa processen, en process som ofta kallas ”web-crawling”. Genom att sammanställa resultaten i en övergripande rapport skulle detta kunna förenkla webbdesignprocessen och hjälpa designers att säkerställa att hela webbplatsen uppfyller tillgänglighetskraven.

1.2 Syfte

Syftet med detta examensarbete är att analysera verktyg för automatisk testning av tillgänglighet på webbsidor. Ett verktyg väljs ut, som kan passa för de ändamål Region Skånes Testcenter behöver. Användandet av verktyget ska underlätta designen av webbsidor så de klarar de krav på tillgänglighetsanpassning som finns.

1.3 Målformulering

Examensarbetets huvudsakliga mål är att undersöka hur tillgänglighetsanpassning av webbsidor kan automatiseras med de verktyg som finns. Ett verktyg ska väljas ut efter analys från ett antal potentiella verktyg. Om möjligt så ska “webcrawling”- en process att automatiskt samla webbinnehåll genom att följa länkar, användas.

Detta för att undersöka möjligheten att använda verktyget för att generera rapporter på en webbsidas samtliga undersidor.

1.4 Problemformulering

Detta examensarbete söker svar på följande problem:

1. Hur ser testning av webbsidors tillgänglighet ut för närvarande?
2. Vilka regelverk måste kunna hanteras av ett automatiskt testverktyg?
3. Vilka testverktyg är passande för automatisering av testprocessen gällande tillgänglighetsanpassning och hur definieras kriterierna för dessa verktyg?
4. Vilket testverktyg ska väljas för rekommendation till att användas av Region Skånes testcenter för automatisering av testprocessen?
5. Hur kan man använda det valda testverktyget i testprocessen på ett bra sätt?

1.5 Motivering av examensarbetet

Det som fick mig att vilja välja detta examensarbete är att det är ett projekt som ger mig möjlighet att använda kunskap jag erhållit under utbildning som programmering och ingenjörsmässigt tänkande. Jag tror även att det kommer att resultera i ett projekt som ger en faktisk användbar rapport, något som kommer att gynna Region Skånes förmåga att leverera webbsidor som är tillgängliga för alla människor framöver.

För Region Skåne kan ett lyckat arbete innebära att man får en överblick över hur testning av tillgänglighetsanpassning för webbsidor kan utvecklas. Potentiellt kan en passande strategi tas fram för att klargöra hur arbetet ska gå till för testare och webbdesigners för att webbsidor ska klara av de kriterier som specificeras av DOS-lagen.

1.6 Avgränsningar

Om det är möjligt att utforma en prototyp för web crawling som kan samarbeta med tillgänglighetsverktyget, kommer den att utvecklas som ett “proof-of-concept” – en förenklad prototyp snarare än en

färdig produkt avsedd för en slutanvändare. Det vill säga att det program eller skript som skapas, kommer prioritera funktionalitet över användarvänlighet och mer vara ett bevis på att det är möjligt snarare än en färdig produkt att användas till testning.

Undersökningen kommer att ske med verktyg avsedda för användning med operativsystemet Windows. Operativsystem som macOS eller Linux kommer ej att tas i åtanke.

För jämförelsen av testverktyg finns ingen budget avsedd för införskaffning. Detta innebär att bara gratisversioner kan användas vid testning av dessa verktyg.

2. Teknisk bakgrund

Detta kapitel inleds med en genomgång om varför tillgänglighetsanpassning är viktigt (2.1). Därefter presenteras relevanta regler och riktlinjer i avsnitt 2.2. Den tekniska bakgrunden avslutas med avsnitt 2.3 där de testverktyg som ska analyseras i den kommande analysdelen introduceras.

2.1 Tillgänglighet och digital jämlikhet

I detta avsnitt introduceras behovet av tillgänglighet på webbsidor. 2.1.1 diskuterar konsekvenser av bristande tillgänglighet. 2.1.2 introducerar olika alternativa sätt för användare med funktionsnedsättningar att använda webben och 2.1.3 behandlar den statistik som finns tillgänglig för hur många människor i Sverige som påverkas ifall tillgängligheten på en webbsida brister.

2.1.1 Digital inkludering och jämlikhet

Funktionsnedsättningar kan vara av olika slag, såsom visuella, auditiva, intellektuella eller motoriska. För att alla ska kunna använda digitala tjänster är det avgörande att webbsidor designas med tillgänglighet i åtanke. Bristande tillgänglighet i webbdesign leder till exkludering av vissa användare [4].

Personer med funktionsnedsättningar kanske inte har möjlighet att röra sig fritt och kan därför ha svårt att ta sig till ett bibliotek för att läsa. I stället kan de använda sig av internet att få tillgång till information och tjänster. En dåligt utformad webbdesign kan dock försvåra denna möjlighet. Internet erbjuder användare möjligheten att utföra uppgifter de kanske inte skulle kunna göra utan det [5].

Bra webbdesign med tillgänglighetsanpassning i åtanke gynnar inte bara människor med funktionsnedsättningar utan även andra grupper. Äldre människor, människor med svårigheter med språket, människor med en långsam uppkoppling eller äldre hårdvara och nya användare av internet är alla grupper som tjänar på tillgänglighetsanpassade webbsidor [5].

Det är viktigt att ta hänsyn till vilka människor som ska använda tekniken. För att en webbplats ska vara tillgänglig måste den designas med alla människor i åtanke, särskilt människor med funktionsnedsättningar. När produkter och tjänster utformas för vara inkluderande så kan alla människor använda dem på samma effektiva och tillfredställande sätt [6]. Genom att prioritera delaktighet i webbdesign når man den bredaste möjliga användargruppen.

2.1.2 Hjälpmedel för personer med funktionsnedsättningar

För olika funktionsnedsättningar finns det olika hjälpmedel för att kunna använda internet. Dessa hjälpmedel och teknologier ligger till grunden till varför webbsidor bör designas på ett specifikt sätt för att vara inkluderande.

HTML eller HyperText Markup Language, utgör grunden till hur webbsidor fungerar idag. Språket innehåller funktionalitet som gör det kompatibelt med olika hjälpmedel för navigering av webbsidor. HTML är utformat med semantiska element som anger webbsidans struktur. Taggar såsom <nav>, <main> och <article> definierar webbsidans layout, vilket kan användas av hjälpmedel som skärmläsare och pekhjälpmedel för att underlätta navigering [7].

Skärmläsare är program som används för att läsa upp innehållet på en webbsida för att möjliggöra navigering för personer med allvarliga synnedsättningar eller blindhet. Dessa program fungerar genom att tolka webbsidans struktur och presenterar endast relevant information, såsom textinnehåll, länkar, alternativ text för bilder och sidtitlar. Användaren navigerar sedan på webbsidan med tangentbordsgenvägar eller andra alternativa hjälpmedel [9].

Det finns skärmläsare tillgängliga för alla operativsystem. För Windows är JAWS och NVDA två av de mest använda alternativen, och för Linux och MacOS finns program som ORCA respektive VoiceOver [7].

För personer med synnedsättningar finns fler alternativ än skärmläsare. Funktioner som zoomning, justering kontrast samt invertering och anpassning av färger är viktigt för många användare [7]. Dessa funktioner gör det möjligt för personer med exempelvis

nedsatt färgseende eller partiell synnedsättning att ta del av webbinnehåll.

Dagens webbsidor har blivit mer och mer komplexa. Vad som tidigare mest bestod av länkar och text är idag en mer interaktiv miljö som kräver mer avancerad navigering. Dessa webbsidor kan, om inte de implementeras rätt, försvåra navigation som tidigare gick att utföra med ett tangentbord.

Individer med allvarligare fysiska funktionsnedsättningar kan behöva andra alternativa inmatningssätt än tangentbord. En metod är Switch Access Scanning (SAS) där en användare navigerar genom att låta en navigeringszon systematiskt vandra över webbsidan. När zonen kommer till platsen användaren vill navigera till så trycker den på en knapp för att välja elementet [7]. En komplicerad webbsida med många element kan därför göra det svårt och tidskrävande att navigera.

2.1.3 Statistik

I Sverige finns det igen exakt data på hur många människor som lever med någon form av funktionsnedsättning. I stället har olika undersökningar gjorts för att estimera antalet, och beroende på undersökning så kan det estimeras att mellan 10%-30% av svenskar har någon form av funktionsnedsättning [1].

I en undersökning utförd av Statistiska centralbyrån (SCB) från 2023 (ULF), så frågades deltagande om de hade:

- Svårt att se, även om de använder glasögon eller linser
- Svårt att höra, även om de använder hörapparat
- Svårt att gå eller gå i trappor
- Svårt att komma ihåg eller koncentrera sig
- Svårt att ta hand om sig själva
- Svårt att kommunicera

Enligt denna definition så hade 9,8% av deltagande någon form av funktionsnedsättning. Om definitionen utvidgas och också omfattar de som svarat *ja* på frågan ”Har du någon fysisk, psykisk eller intellektuell funktionsnedsättning” så stiger andelen människor till 18,3 % [2][3].

I den nationella folkhälsoenkäten, en undersökning från Folkhälsomyndigheten så har andelen människor i Sverige med någon typ av funktionsnedsättning uppskattats vara drygt 20% [1].

Dessa siffror handlar om funktionsnedsättningar av olika typer och olika stor påverkan på individen, men det innebär att man kan uppskatta antalet människor i Sverige med någon form av funktionsnedsättning att vara mellan 1–2 miljoner.

2.2 Lagar och riktlinjer för webbtillgänglighet

I Sverige omfattas offentliga aktörer av den så kallade DOS-lagen när det gäller tillgänglighetsanpassning av digitala tjänster. DOS-lagen bygger på EU-direktivet 2016/2102, vilket hänvisar till den europeiska standarden EN 301 549. Denna standard i sin tur är baserad på de globala riktlinjerna för webbtillgänglighet, Web Content Accessibility Guidelines (WCAG), framtagna av World Wide Web Consortium (W3C). För att förstå DOS-lagen krävs därmed en förståelse av både EN 301 549 och WCAG.

2.2.1 WCAG

WCAG är en samling riktlinjer framtagna av W3C, en organisation grundades 1994 av Tim Berners-Lee med syfte att säkerställa att internet är öppet och tillgängligt för alla [10]. WCAG består av en samling kriterier indelade i tre olika nivåer: A, AA och AAA. Nivå A motsvarar den lägsta nivån, vilket innebär att de mest kritiska kriterierna finns på denna nivå. AA utgör grunden för tillgänglighetskraven i EN 301 549, och är därmed den nivå som måste uppfyllas för att tillgänglighetskraven i DOS-lagen ska uppnås [11].

WCAG 2.2 är den senaste versionen av riktlinjerna och publicerades den 5 oktober 2023. Webbsidor som klarar av version 2.2 är klarar även av de tidigare versionerna 2.0 och 2.1 då WCAG är designad att vara bakåtkompatibel [12].

Det finns totalt 13 riktlinjer i WCAG 2.2, vilka är organiserade i fyra övergripande principer som utgör grunden till webbtillgänglighet [13].

Dessa principer är att en webbsidas innehåll måste vara [14]:

- Möjlig att uppfatta (*Perceivable*) – Information och användargränssnitt måste presenteras på ett sätt som gör att de är tillgängliga till alla användare.
- Hanterbar (*Operable*) – Information och användargränssnitt måste gå att navigeras av alla användare utan att de krävs att utföra åtgärder de inte kan utföra.
- Begriplig (*Understandable*) – Allt innehåll och funktionalitet på webbsidan ska vara lättförståeligt av användare, även gällande hur navigering utförs.
- Robust (*Robust*) – Innehållet på webbsidan måste gå att interagera med på olika sätt, så den även stödjer alternativa navigeringssätt.

De 13 riktlinjerna sin tur innehåller olika framgångskriterier för att avgöra till vilken grad webbsidan uppfyller riktlinjerna. En riktlinje går därför att uppnås till A, AA eller AAA nivå.

Ett framgångskriterie exemplifieras enligt Exempel 1 nedan:

> Success Criterion 2.1.4 No Keyboard Trap
>
> (Level A)
>
> If keyboard focus can be moved to a component of the page using a keyboard interface, then focus can be moved away from that component using only a keyboard interface, and, if it requires more than an unmodified arrow or tab keys or other standard exit methods, the user is advised of the method for moving focus away.

Exempel 1. Framgångskriterie WCAG 2.2

Förklaring:
Framgångskriteriet 2.1.4 tillhör principen 2. Operable och riktlinje 2.1 ”Keyboard Accessible” i WCAG. Kriteriet innebär att om en användare kan navigera till ett visst element på en webbsida via tangentbordet, ska det vara möjligt att lämna detta element utan att behöva använda musen. Om det krävs mer än standardmetoder för

navigering med tangentbord, som piltangenterna eller tab-tangenter ska användaren tydligt informeras om metoden för att ta sig därifrån.

För närvarande är WCAG 2.2 den senaste versionen av WCAG, men att ta fram version 3.0 är ett pågående arbete hos W3C. Version 3.0 är för närvarande ett förslag, men förväntas så småningom bli släppt och bli W3C:s rekommendation för tillgängligt webbinnehåll. Den nya versionen kommer att introducera nya tester och kräva uppdateringar av webbsidor för att kunna säkerställa efterlevnad. W3C menar att webbinnehåll som idag efterlever WCAG 2.2 nivå A och AA förväntas klara det mesta av minimikrav för tillgänglighet som 3.0 kommer att ställa [28].

2.2.2 EN 301 549

Den 26:e oktober 2016 antogs EU-direktiv 2016/2102, vilket reglerar hur myndigheters webbplatser och mobila applikationer inom EU ska hantera tillgänglighetsanpassning inom EU. Artikel 12 i direktivet menar att offentliga aktörers webbplatser skall från och med den 23:e september 2019, följa lagstiftning som baseras på den europeiska standarden EN 301 549 [15].

EN 301 549 har utvecklats av European Telecommunications Standards Institute (ETSI) och specificerar tillgänglighetskrav för informations- och kommunikationsteknik (IKT). Syftet med denna standard är att säkerställa att de produkter och tjänster som omfattas är tillgängliga för personer med olika typer av funktionsnedsättningar [16].

Den första versionen av standarden publicerades 2014 och har sedan dess uppdaterats flera gånger. Den nuvarande versionen, V3.2.1, publicerades 2021. EN 301 549 V3.2.1 omfattar ett brett spektrum av IKT-produkter och tjänster, inte enbart webbsidor. Avsnitt 9 behandlar specifikt riktlinjer för webbtillgänglighet.

Likt WCAG delar avsnitt 9 in sina kriterier i fyra principer; *Perceivable, Operable, Understandable* och *Robust.* Totalt finns det 50 kriterier, vilka motsvarar WCAG 2.1 framgångskriterier på nivå A och AA. Standarden rekommenderar även att, när möjligt, nivå AAA också uppnås [16]. Standarden specificerar alltså version 2.1 av

WCAG och inte 2.2 som är den senaste versionen publicerad av W3C. Detta innebär att de tillkomna kraven i version 2.2 inte behöver uppfyllas för att en webbsida ska uppfylla EN 301 549 [20].

2.2.3 DOS-lagen

Lagen om tillgänglighet till digital offentlig service eller DOS-lagen (2018:1937) är Sveriges lagstiftning gällande tillgänglighetsanpassning av offentliga aktörers webbplatser. Denna lag omfattar bland annat statliga och kommunala myndigheter, kommun- och regionstyrelser samt en del privata aktörer inom sjukvård och utbildning [17].

Aktörer som bedriver viss verksamhet är undantagna från lagens tillämpningsområde, detta gäller bland annat vissa aktörer inom tv och radio, och aktörer vars service inte är kritisk för allmänheten eller är avsedd för personer med funktionsnedsättning [18].

Alla aktörer som omfattas av lagen ska följa de tillgänglighetskrav som specificeras i EN 301 549, i enlighet med EU direktiv 2016/2102 [18]. Myndigheten för Digital Förvaltning (DIGG) är den ansvariga tillsynsmyndigheten i Sverige och har i uppdrag att säkerställa att lagen följs [19]. Om brister gällande tillgänglighetsanpassning upptäcks hos en aktör efter en prövning eller anmälan så ska tillsynsmyndigheten informera aktören om detta och ge denne möjlighet att åtgärda bristerna. Om bristerna inte åtgärdas kan DIGG besluta om ett föreläggande. Vid upprepade överträdelser har myndigheten rätt att förelägga aktören vid vite [18].

2.2.4 Lagen om vissa produkter och tjänsters tillgänglighet

En annan svensk lagstiftning baserat på EU:s tillgänglighetsdirektiv är Lagen om vissa produkter och tjänsters tillgänglighet (LPTT). Denna lag träder i kraft den 28:e juni 2025 och ställer krav på tillgänglighet för ett antal digitala produkter och tjänster inom områden som bland annat elektronisk kommunikation, e-handel och banktjänster [49].

För att uppfylla de krav som ställs ska företag och organisationer uppfylla ett antal harmoniserade standarder där EN 301 549 är en av dem. Den huvudsakliga skillnaden mellan DOS-lagen och LPTT är att DOS-lagen specifikt omfattar digitala tjänster från offentliga aktörer

medan LPTT omfattar digitala produkter och tjänster som riktar sig till konsumenter [50].

För att avklara de krav som finns på webbplatser är det avsnitt 9 *Web* i EN 301 549 som är relevant. Det hänvisar till version 2.1 av WCAG nivå A och AA som krav som ska efterlevas [50].

2.3 Automatiserade testverktyg för webbtillgänglighet

Detta avsnitt ger en introduktion till de olika testverktyg som kommer att analyseras i avsnitt 3.3 Analys av testverktyg. De fyra verktyg som valts ut är Axe, Wave, Pa11y och TAW.

2.3.1 Axe-core

Axe-core är en testmotor för tillgänglighetstestning av webbsidor och stöds av Deque Systems. Axe-core är open-source enligt Mozilla Public License 2.0 (MPL-2.0) och kan användas kostnadsfritt [21] men Deque Systems erbjuder dock även egna betaltjänster som bygger på detta källkodsbibliotek.

Axe-core kan identifiera regelbrott enligt WCAG 2.0, 2.1 och 2.2 på A, AA och AAA-nivå och påstås automatiskt kunna upptäcka 57% av potentiella överträdelser av WCAG:s framgångskriterier. [21].

För att använda funktionaliteten i testmotorn kan axe-core integreras i egna skript eller program med JavaScript eller Python. Eftersom axe-core är ett open-source-projekt är det väldigt flexibelt att anpassas till olika testningsbehov. Det kan därför implementeras både som ett helautomatiserat testverktyg och som ett stöd för manuell granskning, beroende på användningsområde.

När axe-core analyserar en webbsida genereras en lista med resultat för de olika reglerna det kontrollerar. Varje objekt i listan representerar en identifierad regel och innehåller en beskrivning, en förklaring, en URL till mer detaljerad dokumentation samt taggar som identifierar vilken WCAG-regel som testats. Om en regel bryts på webbsidan markeras objektet som ett fel (*violation*), medan potentiella problem markeras som osäkra (*incomplete*) för manuell granskning [22].

Deque Systems står bakom axe-core-projektet och erbjuder betalversioner som bygger på open-source projektet. Dessa betalversioner erbjuder extra funktionalitet för automatisering av testningsprocessen samt användning via en GUI. Det finns två betalversioner av verktyget Axe DevTools: en Pro version och en Enterprise version.

Pro-versionen är ett webbläsartillägg som kan testa komponenter på webbsidor, har möjlighet för enkel delning av tillgänglighetsproblem på webbsidan utöver funktionaliteten som finns i axe-core för testning. Enterprise-versionen erbjuder utöver Pro-versionens funktionalitet CI/CD integration, möjlighet att göra testning via CLI (Command Line Interface), testning för Android och iOS, samt funktionalitet för tillägg av egna regler [29].

2.3.2 WAVE

WAVE är ett verktyg utvecklat av WebAIM för tillgänglighetstester på webbsidor. Det finns flera olika sätt att använda verktyget: för gratis användning finns möjligheten att analysera en webbsida på deras webbplats eller via tillägg för Firefox och Chrome. WebAIM erbjuder även betaltjänster då man kan få tillgång till WAVE API, där användaren antingen betalar för varje gång du använder API:n eller via ett köp av en licens [4].

WAVE analyserar en webbsida och visar vart problem och potentiella problem kan finnas med illustrativa ikoner som indikerar var på sidan problemet är, och vilket typ av problem det är samt hur problemen kan åtgärdas [23]. Detta innebär att verktygen är ett bra alternativ när en kombination av automatiserad testning av WCAG-regler samt mänsklig kontroll är av intresse.

Enligt WAVE:s egna dokumentation är verktyget utformat för att hjälpa utvecklare designa webbsidor med bättre tillgänglighet. WebAIM betonar dock att frånvaron av identifierade problem i verktyget inte garanterar att en webbsida är tillgänglig. Verktyget kan inte avgöra detta på egen hand, utan måste kompletteras av manuell granskning av utvecklare med erfarenhet inom tillgänglighetsanpassad webbdesign. WAVE ska därför ses som ett hjälpmedel för att förenkla delar av testprocessen.

WAVE kan inte testa all funktionalitet som krävs av WCAG, men kan hjälpa till att identifiera vissa typer av regelbrott som exempelvis avsaknad av alt-texter för bilder eller låg kontrastskillnad mellan bakgrund och text [24].

2.3.3 TAW

TAW är ett automatiskt verktyg för tillgänglighetstestning som utvecklas av Technological Center for Information and Communication (CTIC) i Asturias, Spanien. Verktyget stöder testning av WCAG 2.1 och är utformat för både användare med och utan erfarenhet av tillgänglighetsanpassning som vill bedöma en webbplats tillgänglighet [25][4].

Verktyget används genom att ange en URL på TAW:s webbplats, varefter en rapport med WCAG:s framgångskriterier genereras. Resultaten organiseras efter WCAG-nivå (A, AA, eller AAA) och indikerar vilka kriterier som inte uppfylls, potentiella problem samt vilka som inte kan testas automatiskt. En mer detaljerad rapport med information om vilken del av koden problemet motsvarar kan skickas via e-post. Enligt TAW:s användarvillkor är antalet analyser begränsat till ett maximum av 10 per timme per IP-adress [41].

TAW anger tydligt vilka kriterier som inte kan undersökas automatisk och rekommenderar manuell granskning för dessa områden. CTIC är också tydliga med att automatisk testning inte är en garanti på att en webbplats är tillgänglig.

2.3.4 Pa11y

Pa11y är gratis och open-sourceverktyg för testning av webbtillgänglighet. Det kan användas direkt via kommandoraden för enstaka tester av webbsidor eller genom Node.js för mer flexibel och automatiserad testning. Pa11y är licensierat under GNU Lesser General Public License v3.0 (LGPL-3.0), vilket innebär att utvecklare kan använda och modifiera källkoden fritt, men det finns vissa krav om källkoden ändras och distribueras. För användning inom intern utveckling krävs inga licensförpliktelser [26][27].

Pa11y använder primärt HTML CodeSniffer för tillgänglighetstester

men kan också konfigureras för att använda både HTML_CodeSniffer och axe-core, vilket möjliggör kombinerade resultat [26].

För att testa en webbsida används dess URL antingen i ett kommandoradsverktyg eller genom ett skript eller program byggt med JavaScript. Det är även möjligt att testa lokala webbsidor genom att specificera en HTML-fil i stället för en URL. Resultatet från testkörningen är en lista med potentiella fel och information om dessa fel [26].

3. Metod

Detta kapitel berör hur examensarbetet utfördes. Arbetet delades upp i 4 delmoment: en litteraturstudie för att förstå området och för att välja ut kriterier till en analys (3.1), en samling intervjuer för att få en djupare uppfattning på behovet av webbtillgänglighet (3.2), en analys av utvalda automatiserade testverktyg (3.3), och en implementation av ett ”proof-of-concept” program för webcrawling (3.4).

Utöver detta genomfördes även en diskussion med en person på Region Skåne för att få en uppfattning av hur testning av webbtillgänglighet går till för närvarande hos Region Skåne. Resultatet av detta presenteras i 5. Diskussion och Slutsats.

3.1 Litteraturstudie

För den tekniska bakgrunden utfördes en litteraturstudie. Syftet med denna var att ge en grundläggande förståelse för området tillgänglighetstestning samt för de verktyg som används i denna rapport. Denna litteraturstudie ger en teoretisk grund för analysen, eftersom tillgänglighet är ett etablerat forskningsområde med omfattande dokumentation.

Den första delen av studien undersökte behovet av tillgänglighet i samhället sin helhet och specifikt på webben. Konsekvenser av att inte tillgodose alla användare med möjligheten att använda webbsidor undersöktes samt hur många människor i Sverige som skulle påverkas av bristande tillgänglighet i webbdesign.

I den andra delen av litteraturstudien undersöktes de olika standarder och lagstiftningar som berör offentliga aktörers webbsidor gällande tillgänglighet. Webbtillänglighetsstandarden WCAG, den europeiska standarden EN 301 549 samt Sveriges lagstiftning inom området, DOS-lagen visades vara de relevanta reglementena att undersöka.

Den sista delen av litteraturstudien var en studie av de fyra testverktyg som kommer att undersökas i analysdelen: axe-core, WAVE, TAW och Pa11y.

Denna litteraturstudie lade grunden för examensarbetarens urval av testverktyg samt de kriterier som används i analysdelen för att jämföra dem.

Källor för litteraturstudien är böcker och analyser som hittats med sökverktyget LUBsearch samt officiell dokumentation för regelverk och testverktyg.

3.2 Intervjuer

För att få en djupare förståelse av behovet av tillgänglighetstestning hos en offentlig aktör, genomfördes tre intervjuer med personer som har koppling till tillgänglighet på Region Skåne. Intervjuerna genomfördes online via Microsoft Teams och bestod av 11 frågor relaterade till tillgänglighet och tillgänglighetsanpassning (8.1 Bilaga 1: Intervjuguide).

Tre personer intervjuades för detta projekt:

Den första är arbetar som konsult inom tillgänglighet och har omfattande erfarenhet av hur man designar webbsidor med tillgänglighet i fokus. Personen har tidigare arbetat på DIGG och var med när de tog fram sitt arbetssätt gällande tillsyn av offentliga aktörers webbplatser.

Den andra personen arbetar som UX-designer hos Region Skåne och är involverad i hur Region Skåne arbetar med tillgänglighetsanpassning av interna system. Den tredje personen är fullstack-utvecklare hos Region Skåne men arbetar mest med frontend-utveckling.

Dessa personer valdes ut för att ge olika perspektiv på tillgänglighet: en expert på tillgänglighet och dess lagar och riktlinjer gällande webbdesign, en UX-designer som arbetar med interna system, och en utvecklare med praktisk erfarenhet av frontend-utveckling. Två av personerna arbetar hos Region Skåne och kan även ge en inblick i hur tillgänglighet tas i åtanke internt.

Intervjuerna spelades in och transkriberades med hjälp av inspelningsmjukvaran Google Recorder och de resulterande transkripten granskades manuellt för att säkerställa att innehållet

transkriberades korrekt. Resulterande transkript kodades för att hitta de intervjuades svar till intervjufrågorna.

3.3 Analys av testverktyg

För att kunna analysera och jämföra tillgänglighetstestverktygen i detta examensarbete har fem kriterier valts ut efter diskussion med handledare på Region Skåne. Dessa kriterier är utformade för att täcka olika aspekter av verktygens funktionalitet, användarbarhet och effektivitet. Varje kriterium består av flera olika delmoment som representerar de aspekter som bedöms inom respektive område. Nedan presenteras kriterierna tillsammans med deras definitioner:

1. **Kostnad**
   1.1. Är verktyget gratis eller kommersiellt?
   1.2. Om kommersiellt, vad är priset?
   1.3. Finns det potentiella dolda kostnader?
2. **Flexibilitet**
   2.1. På vilka sätt kan verktyget användas? (CLI, API, ...)
   2.2. Är det möjligt att lägga till egna regler till verktyget?
   2.3. Finns det stöd för automation av testprocessen?
3. **Användarbarhet**
   3.1. Krävs det en teknisk kompetens för att använda verktyget?
   3.2. Finns dokumentation och support?
4. **Säkerhet**
   4.1. Går det att använda verktyget lokalt?
   4.2. Kräver verktyget att information om webbsidan laddas upp?
5. **Täckning**
   5.1. Vilka av WCAG:s framgångskriterier täcks?
   5.2. Vilka nivåer (A, AA, AAA) och versioner (2.1, 2.2) av WCAG stöds?

Resultatet från analysen av hur verktygen uppfyller de fem kriterierna och dess aspekter presenteras i avsnitt 4.2, Resultat från jämförelse av testverktyg.

För att vidare undersöka kriterium 5, Täckning utfördes testkörningar med de fyra testverktygen på tre webbsidor:

1. Webbsida från W3C som visar hur en webbsida med tillgänglighetsbrister ser ut.

2. Startsida för utveckling inom Region Skåne.
3. Artikelsida med praktisk information om Region Skånes Universitetssjukhus.

Dessa tester av verktygen på de tre olika webbsidorna skedde under en kort period för att minimera risken att webbsidorna uppdateras vilket skulle kunna påverka testresultatet. Webbsidorna kan ha testats under olika dagar, men när en webbsida testades användes alla verktyg under samma dag.

För testning av verktygen specificerades version 2.1 av WCAG och nivå AA som de tester som jämfördes. Endast problem med direkt referens till WCAG har inkluderats i analysen för att säkerställa en rättvis jämförelse mellan verktygen. Exempelvis har Axe:s *best-practice-fel* exkluderats, liksom WAVE:s *alerts* som inte har en direkt koppling till ett WCAG-kriterium.

Resultatet från dessa tester redovisas i avsnitt 4.2.6, Jämförelse av verktygens testresultat. Där redovisas det totala antalet upphittade fel och potentiell fel, samt vilken typ av fel.

3.4 Implementation

Det första steget för att implementera ett skript eller program för att använda Pa11y på en hel webdomän var att planera upplägget för programmeringen.

Val av programmeringsspråk
Det valda programmeringsspråket för implementationen var JavaScript. För att kunna köra JavaScript-kod lokalt utan en webbläsare installerades även Node.js, en exekveringsmiljö för JavaScript utanför webbläsaren. JavaScript valdes främst då Pa11y är byggt i detta språk, vilket underlättade integrationen med verktyget. Övrig funktionalitet anpassades därefter till samma miljö.

Designprocess
Målet var att skapa ett enkelt men effektivt system för att testa tillgängligheten på alla webbsidor på en domän och sammanställa resultaten på ett överskådligt sätt.

För att åstadkomma detta delades processen in i tre huvudsakliga steg:

Steg 1 – Webcrawling och insamling av URL:er
Det första steget är att utföra webcrawling på en domän för att samla in alla unika URL:er. Dessa filtreras för att endast inkludera webbsidor av typen HTML, så adresser till bland annat PDF-dokument och Excelfiler utesluts. På detta sätt säkerställs att alla adresser i listan är webbsidor som Pa11y kan utföra tillgänglighetstestning på. Den resulterade listan lagras sedan i en .txt-fil med enbart unika URL:er.

Steg 2 – Pa11y-testning av URL:er
När listan med webbadresser har sammanställts används Pa11y för att köra tillgänglighetstester på varje unik webbsida med både HTML_CodeSniffer och AXE som testmotorer. Pa11y kan generera rapporter i JSON-format så varje webbsida får en egen resultatfil.

Steg 3 – Sammanställning av testresultat
När Pa11y-rapporter har skapats, itererar nästa del av programmet igenom alla testresultat och samlar ihop data om antal fel/varningar samt vilka unika typer som upphittades på domänen. En HTML-fil skapas sedan för att presentera resultatet.

Programmeringsprocess
Med detta som plan för hur programmet skulle fungera, programmerades först delen för webcrawling. När webcrawling-delen lyckades gå igenom alla webbsidor och lagra dem, implementerades Pa11y-testningen för att kunna få ut rapporter.

Slutligen implementerades den sista funktionaliteten, rapportgeneration för att kunna presentera resultatfilerna på ett behändigt sätt.

Efter diskussion med handledare på Region Skåne beslutades det att funktionalitet för att ge användare möjlighet att logga in innan webcrawling och testning även skulle implementeras. Detta för att vissa webbsidor kan vara låsta bakom inloggning och därför inte testas om inte en användare är inloggad på webbdomänen.

Struktur och funktionalitet i programmet
Programmet är delades in i fyra delar:

1. Main – Tar emot användarinput och tillkallar funktioner från de övriga delarna för att samla in adresser, köra Pa11y-testning och sammanställa resultat. Hanterar även login-funktionalitet om användaren vill logga in på en domän innan webcrawling/testning.

2. Webcrawling – Funktioner för att genomföra webcrawling på en domän och generera en lista av unika webbadresser.

3. Pa11y-testning – Funktioner för att köra Pa11y-tester på en webbsida och spara resultatfilen i JSON-format.

4. Rapportgeneration – Funktioner för att bearbeta och sammanställa testresultaten från JSON-filer i ett HTML-dokument.

Externa bibliotek
För att implementera den nödvändiga funktionaliteten användes dessa bibliotek och moduler:

Puppeteer – Användes för att kunna skapa ett webbläsarobjekt som kan användas både till webcrawling och testning med Pa11y. Nödvändigt då det kan spara login-data för sessionen så crawling och testning kan utföras med en inloggad användare.

File system – Användes för att kunna spara och läsa in data från olika filtyper på enheten.

Path – Användes för att kunna navigera i enhetens filsystem för att kunna hitta filer.

Readline – Användes för att kunna läsa in användarinput i terminal eller kommandoradstolken.

3.5 Källkritik

Detta avsnitt beskriver hur källor valts ut för det innehåll som har skrivits i examensarbetets rapport. Varje avsnitt i rapporten beskrivs separat då olika avsnitt kan ha betydliga skillnader i typer av källor. För 2.1 delades förklaringen av källkritik upp i tre delar då typen av källor var väldigt olika. För de andra kapitlen beskrivs valet av källor på kapitelnivå.

2.1.1 Digital inkludering och jämlikhet

Detta avsnitt har information tagen från källor som rapporter [4] och böcker [5][6] som handlar antingen om tillgänglighet generellt och hur bristande tillgänglighet i allmänhet har en negativ effekt på människor med funktionsnedsättningar eller specifikt om digital tillgänglighet. Dessa källor anses pålitliga då skribenterna är erkända akademiker och pålästa inom områdena tillgänglighet och webbtillgänglighet.

2.1.2 Hjälpmedel för personer med funktionsnedsättningar

Denna del tar information från ett kompendium [9] med olika författare som handlar om tillgängliga hjälpmedel för personer med funktionsnedsättningar, samt från ett kapitel om hjälpmedel från en bok om webbtillgänglighet [7]. Informationen från dessa källor är pålitlig då författarna av texterna informationen kommer ifrån är välkända akademiker inom området.

2.1.3 Statistik

All information i detta avsnitt är i form av statistik publicerad av svenska myndigheter. Undersökningarna som statistiken kommer ifrån är blanda annat från SCB och Folkhälsomyndigheten [1][2][3]. Denna statistik kan ses som en väldigt pålitlig källa då dessa myndigheter hålls till en hög standard gällande den information de publicerar.

2.2 Lagar och riktlinjer för webbtillgänglighet

Detta kapitel har bara direkta källor då det handlar om lagar och riktlinjer. All information är tagen direkt från respektive myndighet eller organisation så att informationen ska vara så exakt som möjligt. Information är tagen från bland annat DIGGs och PTS webbsidor [11][17][20][49][50], förordningar från Regeringskansliet [18][19] och officiell dokumentation för WCAG [10][12][13][14][28] och EN 301 549 [15][16].

2.3 Automatiserade testverktyg för webbtillgänglighet

Från detta kapitel är flera av källorna lite mindre pålitliga än andra i rapporten. Om möjligt så har information kommit från olika analyser eller rapporter [4][23] för att beskriva verktygens funktionalitet. Om det inte funnits tillräcklig information har verktygens egna webbsidor [24][25][41] och dokumentation [21][22][26][27] använts. Offentlig

dokumentation har ansetts mer pålitlig än information på deras webbsidor, så det har använts om möjligt.

4.1 Intervjuer

Svaren på intervjufrågorna är subjektiva. Det handlar mer om att få ett perspektiv på hur olika typer av människor ser på olika aspekter inom tillgänglighet som kan vara intressant för examensarbetet.

4.2 Resultat från analys av testverktyg

I resultatdelen för verktygen har, om möjligt, information tagits fram genom att testa funktionaliteten hos verktygen. Om detta inte varit möjligt som till exempel vid kriteriet för kostnad, har information från opartiska rapporter [4] och organisationer [46][47] eller verktygens egna webbsidor [29][32][34][37][39][40][45] eller offentliga dokumentation [22][26][30][31][33][35][36][38][42][43][44][48] använts som källor.

Officiell dokumentation från bland annat GitHub om hur verktygen uppfyller kriterium anses vara mer pålitlig än information på respektive verktygs hemsida. Detta då flertalet av verktygen har en kommersiell aspekt och verktygens förmåga kan potentiellt överdrivas i försäljningssyfte.

4. Resultat

Detta kapitel presenterar resultatet från de genomförda intervjuerna, analysen av de automatiserade testverktygen samt implementeringen av det valda testverktyget som ett proof-of-concept. Syftet är att redovisa insamlade data och insikter som ligger till grund för den efterföljande diskussionen och slutsatserna i rapporten.

I avsnitt 4.1 sammanställs resultatet från de tre intervjuer med experter inom tillgänglighet, webbutveckling och UX-design. I avsnitt 4.2 presenteras resultatet från en jämförelse av de fyra testverktygen: axe-core, WAVE, TAW och Pa11y. Avsnitt 4.3 presenterar examensarbetets rekommendation av testverktyg utifrån jämförelsen. Slutligen i avsnitt 4.4 så beskrivs resultatet av den implementation av ett testverktyg för automatisk testning hos Region Skåne.

4.1 Intervjuer

Detta avsnitt presenterar resultatet av de intervjuer som genomfördes för detta projekt. Resultatet redovisas i form av fyra teman som beskriver vad intervju-frågorna handlade om. Dessa intervjufrågor finns i avsnitt 8.1 Bilaga 1: Intervjuguide.

4.1.1 Tillgänglighet i praktiken

Samtliga intervjuade personer har observerat ett ökat fokus på digital tillgänglighet de senaste åren. Fler organisationer och utvecklare har börjat ta tillgänglighetsanpassning på större allvar, särskilt efter att det har blivit ett lagkrav. Dock varierar prioriteringen mellan olika organisationer.

UX-designern upplever att Region Skåne inte har prioriterat tillgänglighet lika högt som vissa andra offentliga aktörer. Denne menar att frågan bromsas av interna faktorer och anser att fler personer i organisationen bör engagera sig i digital tillgänglighet.

Utmaningar för organisationer

Både UX-designern och utvecklaren lyfter fram föråldrade interna system som en stor utmaning. Dessa system försvårar implementeringen av moderna tillgänglighetsanpassningar och skapar hinder i utvecklingsprocessen.

UX-designern och konsulten pekar på kompetensbrist som ett avgörande problem. Många utvecklare idag saknar kunskap om hur man skapar tillgängliga webbsidor, och en generell förståelse för tillgänglighet saknas ofta hos beslutsfattare i organisationer. De intervjuade menar att om kunskapen om tillgänglighet ökar inom organisationers ledningar, så kommer även att kompetensen hos utvecklare att förbättras.

UX-designern utvecklar genom att påstå att förståelse för tillgänglighet bör vara en del av en utvecklares verktygsbälte: ” […] du är inte fullstackutvecklare om du inte kan digital tillgänglighet”.

Konsulten påpekar även att vissa utvecklingsramverk används trots att de inte är byggda med tillgänglighet i åtanke. Detta kan leda till att tillgänglighet får en lägre prioritet under utvecklingsprocessen.

Viktiga faktorer för tillgänglig webbutveckling
UX-designern betonar vikten av att integrera tillgänglighet i hela utvecklingsprocessen snarare än att se det som en uppgift som ska utföras i slutskedet. Ett bra designsystem kan hjälpa utvecklare att implementera tillgänglighet från början.

Utvecklaren och konsulten framhåller att en enklare design ofta är bättre ur ett tillgänglighetsperspektiv. De menar att HTML i sin grundform är väl anpassat för tillgänglighet och att man som webbutvecklare bör ha god kunskap i HTML för att bygga tillgängliga webbsidor.

De varnar för att lägga till för mycket JavaScript och CSS, vilket kan försämra tillgängligheten på webbsidan. Vidare framhåller utvecklaren att ”om du börjar använda JavaScript eller CSS för att få ett element att bete sig på ett annat sätt, då förlorar du tillgängligheten”

Konsulten betonar vikten av att designa webbsidor med användaren i fokus. En utvecklare bör inte utgå från sin egen användarupplevelse när den skapar webbsidor utan att man i stället designar webbsidor med en bred användargrupp i åtanke, inklusive personer med funktionsnedsättningar. Genom att tillämpa principer för universell utformning kan man skapa en grundläggande design som fungerar för

alla och sedan bygga vidare med förbättringar för de som kan dra nytta av det utan att påverka den grundläggande konstruktionen.

Vanliga problem på svenska webbplatser
Ett vanligt återkommande problem på svenska myndigheter och organisationers webbplatser är bristfällig rubrikstruktur. Enligt konsulten och UX-designern skapar detta svårigheter för användare som navigerar med tangentbord och skärmläsare. Detta kan leda till att användaren har svårt att hitta relevant information eller att denne upplever frustration vid webbsurfing.

Felaktig hantering av alt-texter är ett annat vanligt problem. Konsulten förklarar att alt-texter inte endast handlar om att beskriva en bild, utan att de bör betraktas som en alternativ representation av informationen bilden förmedlar. En direkt beskrivning av just bildens innehåll är därför inte alltid den bästa lösningen.

Andra brister som nämns är dålig kontrast, avsaknad av stöd för mörkt läge och att vissa webbplatser inte går att navigera med endast tangentbord. Dessa faktorer kan göra webbplatser otillgängliga för många användare och bör åtgärdas.

4.1.2 Inkludering
Resultatet från en intervjufråga om konsekvenserna av bristande tillgänglighet visade att samtliga av de intervjuade ansåg att personer som inte kan använda en webbsida utsätts för en form av diskriminering. De får inte samma möjligheter att hitta information eller utnyttja digitala samhällstjänster på lika villkor.

Konsulten konstaterade att ”det är samma problem som när man utestänger människor i den fysiska världen […] man försvårar för människor som redan har tillräckligt med svårigheter på olika sätt”. Bristande tillgänglighet kan även vara ett demokratiproblem. Om en offentlig aktörs webbplats inte är tillgänglig kan vissa människor ha svårt att kontakta sin kommun, exempelvis för att påpeka brister eller göra en anmälan.

Tillgänglighet handlar dock inte bara om publika webbplatser, utan även om interna system. Om digitala verktyg på en arbetsplats inte är

anpassade kan personer med funktionsnedsättningar få sämre förutsättningar på arbetsmarknaden.

4.1.3 Lagar och riktlinjer

De intervjuade har märkt en generell förbättring av webbtillgänglighet sedan EU-direktivet infördes 2019. Förbättringen upplevs dock som ojämn – vissa aktörer har lyckats bättre än andra. UX-designern anser att Region Skåne fortfarande har mycket kvar att arbeta med och att vissa andra aktörer kanske ”sätter huvudet rakt ner i sanden”.

Konsulten menar att alla aktörer inte har samma förutsättningar att förbättra sin webbtillgänglighet. Stora kommuner och myndigheter har betydligt mer resurser att avsätta för detta arbete. Mindre kommuner kan däremot ha få anställda som ansvarar för webbsidorna, vilket kan förklara varför tillgänglighetsarbetet går långsammare på vissa håll.

Förbättringar i lagstiftning

I dagsläget omfattar lagkraven endas WCAG 2.1 på nivå A och AA. UX-designern anser att offentliga aktörer även borde omfattas av krav på att uppfylla nivå AAA. Dessa mål anses mer att vara något att sträva efter. Hen menar att om det inte blir ett lagkrav, kommer få aktörer att aktivt arbeta för att nå AAA-nivån.

Konsulten och utvecklaren påpekar att nuvarande framgångskriterier behöver utvidgas för att inkludera fler typer av funktionsnedsättningar. De anser att tillfälliga funktionsnedsättningar och kognitiva svårigheter, som stress, ångest och depression, ofta förbises trots att många människor påverkas av dem. De menar att dessa faktorer påverkar användbarheten av digitala tjänster och bör tas i åtanke vid design av webbsidor.

4.1.4 Automatiserad testning

Alla intervjuade var medvetna om att automatiserade testverktyg, som AXE och WAVE, existerar och vad de används till. Både utvecklaren och UX-designern anser att automatiserad testning kan vara en hjälp i utvecklingsprocessen. De använder i nuläget inte sådana verktyg dock.

Konsulten menar att automatiserad testning kan stötta utvecklare att designa tillgängligt, men påpekar att dessa verktyg inte kan ersätta

manuell testning. De täcker inte alla WCAG:s framgångskriterier. Till exempel kan WAVE upptäcka att en bild saknar alt-text, men det kan inte avgöra om texten är relevant och korrekt. Det måste en människa avgöra. Så länge användaren av testverktyget är medveten om dess begränsningar, kan det dock vara ett bra sätt att identifiera vissa problem.

UX-designern instämmer och påpekar att Region Skåne behöver mer manuell testning. Hen anser att det är värdefullt att låta personer med funktionsnedsättningar testa de digitala system som utvecklas. De kan då upptäcka saker som utvecklare kanske inte tänker på.

Integrering av testverktyg i testprocessen
Både utvecklaren och UX-designern konstaterar att de endast kan uttala sig om hur testprocessen går till i sina egna team. Utvecklaren har en egen testprocess då denne använder sig av testautomatiseringsramverket Playwright och verktyget Storybook för att testa de funktioner de implementerar. Detta för att upptäcka både design- och tillgänglighetsproblem. I UX-designerns fall hanteras testning av en specifik person i teamet. Hur tillgänglighetsfrågor upptäcks beror då på hur denna testperson arbetar.

Konsulten tror att vissa ”mogna” organisationer redan har börjat implementera kontinuerlig testning med automatiserade testverktyg. I sådana fall används verktygen för att kontrollera webbsidan vid varje ny funktion eller uppdatering. Om ett problem identifieras kan man då stoppa att brister introduceras på webbsidan. Denne påstår att det är nog många ”utvecklare använder typ Axe DevTools, WAVE och liknande, oftast på eget initiativ, snarare än att det är något organisationen påbjuder”.

4.2 Resultat från analys av testverktyg

I detta avsnitt presenteras resultatet av analysen av testverktygen axe-core, WAVE, TAW och Pa11y. Verktygen har jämförts enligt de kriterier som presenterades i 3.3 Analys. Verktygens prestation inom varje kriterium beskrivs i respektive avsnitt, ett resultat som sammanfattas med en tabell i avsnittets slut.

4.2.1 Kostnad

Kostnad har analyserats baserat på hur varje verktyg uppfyller underkriterium 1.1, 1.2 och 1.3. Resultatet för detta beskrivs i respektive avsnitts del och sammanfattas i tabell 1.

axe-core

Pro-versionen av Deque Systems Axe-DevTools kostar 45 dollar/månad/användare och priset på Enterprise-versionen behöver förhandlas med en säljare på Deque Systems och beror på storleken på företaget [29].

Det finns även ett gratis webbtillägg för testning, detta har begränsat funktionalitet. Det kan till exempel köra de automatiska testerna men saknar förmåga att spara resultatet från testkörningen.

axe-core som dessa verktyg baseras på, är ett open-sourceprojekt. Detta innebär att källkoden är gratis att använda, så länge projektets licensvillkor följs. Därmed är det möjligt att utveckla egna program eller skript skripts utan att använda Axe DevTools [22]. Ett egenutvecklat program skulle dock innebära att det behövs en utvecklare som underhåller programmet och ser till att det fungerar.

WAVE

WAVE erbjuder en gratisversion av sin tjänst i form av dess webbtillägg för Chrome, Firefox och Microsoft Edge. Det finns även två betalversioner av verktyget som ger tillgång till WAVE API:

Kreditbaserat system: Användaren köper krediter där priset per kredit sjunker vid större köp (exempelvis 1000 krediter för 0,03 $ per kredit). Antalet krediter per API-kallelse beror på analysens komplexitet [30].

Licensbaserat system: Ger möjlighet att använda API:et lokalt. Kostnaden beror på licensnivå: Basic (4000$/år), Expanded (8000$/år) och Unlimited (12 000$/år), där varje nivå avgör antalet installationer och API-anrop [31].

TAW

Användning av TAW:s webbaserade testning är gratis. De erbjuder en tjänst som heter TAW CMS som kan integreras för att testa webbsidor

innan de publiceras. Användning av detta innebär att ett Content Management System (CMS) behöver användas. Kostnaden för TAW CMS behöver förhandlas [32].

Pa11y
Samtliga av Pa11y:s versioner är gratis att använda förutsatt att användaren följer licensavtalet. I och med att det är open-source så måste potentiellt ett program eller skript byggas och underhållas, vilket medför underhållskostnader.

Tabell 1. Kriterium 1: Kostnad

Verktyg	1.1 Gratis eller kommersiellt	1.2 Pris	1.3 Dolda kostnader
axe-core/Axe DevTools	Gratis/Kommersiellt	- axe-core: gratis - Pro: 45$/månad/användare - Enterprise: förhandlingsbart	axe-core: underhåll
WAVE	Gratis/Kommersiellt	- Webbtillägg: gratis - API: 0,04$ - 0,025$ /kredit - Standalone: 4000$ - 12 000$ /år	
TAW	Gratis/Kommersiellt	- Webbsida: gratis - TAW CMS: förhandlingsbart	TAW CMS: behov av ett CMS
Pa11y	Gratis	- Pa11y: gratis - Pa11y CI: gratis	Underhåll

4.2.2 Flexibilitet
Verktygens flexibilitet analyserades enligt underkriterium 2.1, 2.2 och 2.3. Resultatet från detta presenteras i de olika verktygens avsnitt och sammanfattas i tabell 2.

axe-core
Axe DevTools erbjuder automatiserade tillgänglighetstester direkt i webbläsaren genom sitt webbläsartillägg. För mer anpassad testning kan axe-core användas, vilket möjliggör integration av tillgänglighetstester i egna skript eller program via API-anrop i JavaScript. Axe-core har även stöd för integration med testautomatiseringsramverket Robot Framework, ett

testautomatiseringsramverk för Python [33]. Detta ramverk används inom Region Skånes testcenter för testautomation. Axe DevTools har också funktionalitet för testning via CLI.

Både Axe DevTools och axe-core stödjer anpassning genom tillägg av egna regler [22]. För Axe DevTools är dock denna funktionalitet begränsad till betalversionen.

För att automatisera testprocessen med axe-core krävs det att användaren själv implementerar testflödet, exempelvis genom att skriva testskript i JavaScript eller genom att använda ramverk med stöd för axe-core.

WAVE
WAVE är främst tillgängligt som ett kostnadsfritt webbläsartillägg för Chrome, Firefox och Microsoft Edge [34]. För mer avancerad testning erbjuder WAVE API-tillgång för en kostnad. [4][30][31].

Det finns ingen dokumenterad information om att WAVE skulle stödja möjligheten att lägga till egna regler.

Automatisering av testprocessen med webbläsartillägget är begränsad, men genom API-åtkomst finns möjlighet till automatisering. Detta kräver dock att användaren själv utvecklar av skript eller program för att integrera API-funktionaliteten i testprocessen.

TAW
För kostnadsfri användning av TAW kan användaren skicka en förfrågan om att analysera en webbsida genom att ange dess URL på TAW:s webbsida. TAW CMS erbjuder möjligheten att analysera webbinnehåll för tillgänglighetsproblem inom ett Content Management System (CMS). Dessutom finns en nedladdningsbar version verktyget, men detta har endast stöd för WCAG 1.0 [32].

Det finns ingen dokumentation som indikerar att TAW Web eller TAW CMS skulle stödja att användaren lägger till egna regler för testning. Däremot TAW Standalone erbjuder dock denna funktionalitet [32].

Att automatisera testningen av webbtillänglighet med TAW är begränsat. Eftersom TAW Web kräver att användaren manuellt skickar en förfrågan för analys, blir det svårt att på ett tillförlitligt sätt automatisera testningen. TAW CMS påstås kunna användas för att testa webbsidor innan de publiceras, men det finns ingen tillgänglig information om stöd för testautomatisering.

Pa11y

Pa11y går att använda direkt i kommandoradtolken eller med Node.js, en run-time miljö för att köra JavaScript [26]. Det finns ingen dokumentation som indikerar att Pa11y stödjer tillägg av egna regler.

För automatiserad testning erbjuder Pa11y flera möjligheter. Användaren kan själv utveckla och underhålla skript och program för automation av tillgänglighetstestningen. Dessutom finns open-source-projekt som utvidgar funktionaliteten:

- Pa11y Dashboard - ett webbgränssnitt som automatiskt kan testa specificerade webbsidor, och kan visualisera testresultat över tid [35].
- Pa11y CI - ett kommandoradsverktyg för användning inom kontinuerlig integration (CI), vilket möjliggör bevakning av tillgänglighetsproblem vid kodändringar [36].

Tabell 2. Kriterium 2: Flexibilitet

Verktyg	2.1 Användningssätt	2.2 Stöd för egna regler	2.3 Stöd för automatisering av testprocessen
axe-core/Axe DevTools	- DevTools: CLI, webbläsartillägg - API	- Ja	- axe-core: egen implementation
WAVE	- Webbläsartillägg - API	- Nej	- API: egen implementation
TAW	- Webbsida - CMS - Program	- Ja (standalone)	- Nej
Pa11y	- CLI - Node.js	- Nej	- Egen implementation - CI - Webbgränssnitt

4.2.3 Användarbarhet

Hur verktygen uppfyller kriteriet för användarbarhet och dess underkriterium 3.1 och 3.2 presenteras i detta avsnitt. Resultatet sammanfattas i tabell 3.

axe-core

Eftersom axe-core kräver implementering av en utvecklare eller en tekniskt kunnig användare, behövs god teknisk kompetens för att använda verktyget. Användaren behöver skapa ett skript själv samt korrekt installera de bibliotek som behövs för användning av verktyget. Däremot kan en tekniskt kunnig användare utveckla ett skript eller program som andra, med mindre förståelse inom området kan använda. För att förstå rapporterna som verktyget genererar krävs en viss kompetens, eftersom de bygger på WCAG:s framgångskriterier och dess struktur.

För versionen av Axe DevTools som är ett webbläsartillägg krävs ingen speciell teknisk kompetens för att förstå hur verktyget används. I denna form så körs tillägget via inspektörfunktionen på webbläsaren och resultatet presenteras i detta fönster med information om potentiella tillgänglighetsproblem.

För axe-core finns god dokumentation på verktygets GitHub-sida. Där beskrivs hur verktyget fungerar, hur man installerar det och hur man kör tester [21]. Det finns även dokumentation på hur API:n fungerar i detalj, inklusive hur en användare kan specificera kommandon i JavaScript för att anpassa testresultatet, till exempel baserat på WCAG-nivå [22]. Deque Systems webbplats innehåller även omfattande dokumentation för betalversionerna av Axe DevTools, inklusive detaljerad information om webbtillägget [37]. Utöver detta finns även detaljerad dokumentation om de olika testregler axe-core använder vid testning av en webbsida [48].

WAVE

Användning av WAVE webläsartillägg för tillgänglighetstestning är en enkel process. En användare behöver bara installera tillägget till en av de webbläsare som stöds; Firefox, Chrome eller Microsoft Edge. Därefter kan användaren generera rapporter på webbsidor genom att klicka på en knapp i sin webbläsare alternativt genom att använda ett

kortkommando. Detta möjliggör tillgänglighetstestning även för oerfarna användare.

Presentationen av resultatet för tillgänglighetsanalysen i WAVE är också passande för nyare användare. Detta då det i stället för att enbart lista problem visualiserar vart på sidan problemen finns. Verktyget visar det med ikoner på webbsidan samt en förklaring av det potentiella problemet i ett fält vid sidan om webbsidan.

WAVE har en del inbyggd dokumentation för att hjälpa användare förstå verktygets grundläggande funktioner, men saknar omfattande teknisk beskrivning. Detta kan göra det svårt för nya användare att få en djupare förståelse för verktygets hela funktionalitet. API-versionen har viss dokumentation där nödvändiga parametrar för API-anrop beskrivs, men den är relativt kortfattad [38].

TAW

TAW kräver ingen specifik teknisk kompetens för att användas. Allt en användare behöver göra för att generera tillgänglighetsrapport är att använda testfunktionen på TAW:s webbsida. Resultatet redovisas på hemsidan och en detaljerad rapport går att få skickad via email.

Den detaljerade versionen av rapporten visar tydligt för användaren vilka typer av problem som har funnits, samt hur det relaterar till WCAG. Användaren kan även se vart i hemsidans kod problemen har upptäckts.

Omfattande teknisk dokumentation saknas hos TAW. Det finns en del kortfattad dokumentation på TAW:s webbsida, vilket speglar verktygets enklare och mer användarvänliga tillvägagångssätt för tillgänglighetstestning. Testrapporterna ger en del information om de olika testerna som körs och länkar till W3Cs dokumentation för hur testning kan fungera [47].

Pa11y

Pa11y behöver en viss teknisk kompetens för att använda. Att det är open-source innebär att användaren behöver installera nödvändiga komponenter själv. För användning i kommandoradstolken måste Pa11y installeras med Node Package Manager (npm) och för ytterligare funktionalitet med JavaScript krävs även Node.js. Detta gör

det svårare att använda för en användare som är obekant med dessa steg.

För att förstå resultaten så krävs även en viss förståelse för hur WCAG fungerar. Detta då tillgänglighetsproblem som upptäcks på en webbsida visar vilket framgångskriterie som misslyckas samt information om vart problemet är. Detta innebär att användaren behöver en viss teknisk kompetens för att förstå denna information.

Det finns god dokumentation om Pa11y på dess GitHub-sida. Där finns information om installationsprocessen, hur testningen går till och hur en användare kan konfigurera Pa11y för att kunna anpassa testningen [35]. På Pa11ys hemsida finns det guider för Pa11ys funktionalitet [39]. Det finns även god dokumentation för hur testmotorn HTML_CodeSniffer fungerar på dess webbsida. Där kan information om de tester som körs hittas, samt om vad som inte omfattas av testningen [46].

Tabell 3. Kriterium 3: Användarbarhet

Verktyg	3.1 Teknisk kompetens	3.2 Dokumentation
axe-core/Axe DevTools	- Kunskap inom programmering - Förståelse av WCAG-kriterier	- GitHub - Dokumentation om tester
WAVE	- Nej	- Information i testresultat - Dokumentation om API-version
TAW	- Nej	- Information i testresultat
Pa11y	- Kunskap inom programmering - Förståelse av WCAG-kriterier	- GitHub - Guider - Dokumentation om tester

4.2.4 Säkerhet

Verktygens säkerhet analyserades enligt underkriterierna 4.1 och 4.2 och presenteras i detta avsnitt. Resultatet för respektive verktyg beskrivs i dess avsnitt och sammanfattas sedan i tabell 4.

axe-core
Genom att installera axe-core på användarens egna system går det att använda verktyget på lokal maskin. För webbläsartillägget Axe DevTools Browser Extension finns det också möjlighet att använda det i ett ”offline”-läge med begränsad funktionalitet [40]. Denna funktionalitet är dock begränsad till betalversionen.

För användning med axe-core eller i offline-läget för webbläsartillägget delas ingen information om webbsidorna som analyseras med Deque Systems. Detta då en lokal instans av verktyget körs.

WAVE
Verktyget WAVE kan användas lokalt med API-versionerna, men dessa kräver kommunikation med WAVE:s servrar via API-anrop för att genomföra analyser. Enligt WAVE:s dokumentation, körs webbläsartillägget direkt i webbläsaren utan att skicka någon information till WAVE:s servrar [34].

Vid användning av API-versionerna så skickas endast webbsidans URL till WAVE:s servrar för analys. Ingen ytterligare information om sidans innehåll eller användardata skickas [38]. För webbläsartillägget sker all analys lokalt, vilket innebär att ingen information om webbsidan skickas till WAVE [34].

TAW
TAW Standalone kan användas lokalt men täcker endast kriterier för WCAG 1.0. Därför saknar den förmågan att analysera kriterier relevanta för att uppfylla kraven enligt DOS-lagen. TAW Web kan däremot inte användas lokalt, eftersom användaren behöver begära tillgänglighetsanalyser via TAW:s webbsida. Vid analys via TAW Web skickas endast webbsidans URL till TAW:s servrar, och ingen ytterligare information om webbsidans innehåll behövs.

Pa11y
Pa11y kan användas lokalt via kommandoradstolken alternativt via eget skript i JavaScript på egen maskin. Då Pa11y-paketet installeras lokalt så behövs ingen information skickas till några externa webbservrar, så ingen data delas med Pa11y.

Tabell 4. Kriterium 4: Säkerhet

Verktyg	4.1 Lokal användning	4.2 Delning av information
axe-core/Axe DevTools	- Ja	- Nej
WAVE	- Ja	- Webbläsartillägg: Nej - API: URL
TAW	- Nej	- URL
Pa11y	- Ja	- Nej

4.2.5 Täckning

Detta avsnitt presenterar resultatet från analysen av kriterium 5 Täckning och dess underkriterier 5.1 och 5.2. Resultatet redovisas i respektive verktygsdel och sammanfattas i en tabell i slutet på avsnittet (tabell 5). I bilaga 2 (8.2) detaljeras vilka kriterier som har tester för varje verktyg.

Notera att täckning är ett mycket komplext kriterium. De framgångskriterier som verktygen analyserar har ofta ytterligare krav som inte går att testas med automatiserade verktyg utan måste undersökas manuellt. Det enda resultat verktygen visar är om webbsidan klarar verktygets specifika test.

Till exempel så kan verktygen hitta avsaknad av alt-text till bilder på webbsidor, men de kan inte avgöra om existerande alt-text är korrekt enligt WCAG:s specifikationer. Avsaknad av alt-text bryter mot framgångskriterium 1.1.1 i WCAG, något som även inkorrekt alt-text gör. Att ett verktyg har ett testfall för ett av WCAG:s framgångskriterium innebär inte att framgångskriteriet testas i den omfattning det bör.

Axe

Axe har regler för WCAG versionerna 2.0, 2.1 och 2.2 för nivåerna A och AA. De rekommenderar användning av deras största och nyaste regeluppsättning vilken omfattar version 2.1 på nivå AA [42]. WCAG-nivå AAA stöds på version 2.0 [22]. Utöver WCAG-regler så kan även en del ”best practice”-regler användas för en bättre användarupplevelse [43].

De olika reglerna som axe testar en webbsida för, kan motsvara en eller flera WCAG framgångskriterier. Regeln ”area-alt” till exempel

täcker ett behov som ställs av framgångskriterium 2.4.4 samt 4.1.2. Vissa framgångskriterier har även flera regler som behövs uppfyllas. Totalt så nämner axe-cores API-dokumentation att 23 unika framgångskriterier testas för WCAG 2.1 nivå A och AA [43] främst inom kategorierna *Perceivable* och *Operable*.

WAVE
Verktyget WAVE testar webbsidor mot WCAG version 2.1 för nivåerna A och AA [44]. Eftersom WCAG version 2.1 är bakåtkompatibelt, omfattar det även de krav som ställs i tidigare versioner. Enligt WAVE:s dokumentation testar verktyget 22 av WCAG:s framgångskriterier. [44].

Det är viktigt att notera att vissa tester inte kan avgöra om ett framgångskriterium inte uppfylls, utan snarare uppmanar användaren att själv undersöka om funktionaliteten uppfyller kriteriet. Detta passar väl WAVE:s mer visuella presentation av resultaten på webbsidan.

TAW
TAW har tester för WCAG version 2.1 på alla tre nivåer; A, AA och AAA [45]. Verktyget saknar dock dokumentation om sina testfall och hur de täcker WCAG:s framgångskriterier. Enligt TAW:s rapporter, som genereras efter analys, så testas minst 40 olika kriterier i någon omfattning, varav majoriteten inom kategorierna *Perceivable*, *Operable* och *Understandable*.

TAW erbjuder ingen dokumentation om vilka framgångskriterier som har testfall i sin dokumentation. Flera av de tester som finns med i rapporterna verkar dock inte köras i de tester som utförts utan markeras som ”*unable to perform automatic review*”. Dessa är markerade i 8.2 Framgångskriterier med tester.

Pa11y
Pa11y testar mot WCAG 2.1 på alla tre nivåer (A, AA, AAA) och använder sig av HTML_CodeSniffer för sin regeluppsättning [26]. HTML_CodeSniffer har regler för testning av delar av 22 unika framgångskriterier på nivåerna A, AA och AAA [46].

Tabell 5. Kriterium 5: Täckning

Verktyg	5.1 Andel framgångskriterier med tester	5.2 Nivåer och versioner av WCAG
axe-core/Axe DevTools	- 23 framgångskriterier	A, AA 2.0, 2.1, 2.2
WAVE	- 22 framgångskriterier	A, AA 2.1
TAW	- 40* framgångskriterier (se avsnitt om TAWs täckning)	A, AA, AAA 2.1
Pa11y	- 22 framgångskriterier	A, AA, AAA 2.1

4.2.6 Jämförelse av verktygens testresultat

Detta avsnitt presenterar resultatet från testningen av tre webbsidor med tillgänglighetsverktygen axe, WAVE, TAW och Pa11y. Resultatet sammanfattas i tabeller 6, 7 och 8. I tabell 9 redovisas de olika typer av fel som identifierades av verktygen.

Jämförelse av fel och varningar på webbsidor

Fel är tester där verktyget har tydligt identifierat att ett WCAG-kriterium inte uppfylls för en funktion på en webbsida, medan varningar indikerar potentiella problem som kräver manuell granskning.

Tabell 6 visar resultatet från den första testade webbsidan, en exempelwebbsida från W3C med bristande tillgänglighet. Som förväntat identifierade samtliga verktyg ett stort antal fel:

- Axe rapporterade 44 fel
- WAVE rapporterade 39 fel
- TAW rapporterade 59 fel
- Pa11y rapporterade 41 fel

Antalet identifierade varningar skiljde dock avseendevärt mellan verktygen:

- Axe identifierade 2 varningar
- WAVE identifierade 32 varningar
- TAW identifierade 114 varningar
- Pa11y identifierade 65 varningar

Tabell 6: Kvantitativ jämförelse av testresultat från webbsida 1, CityLights – Testsida

Verktyg	Antal fel	Antal varningar
AXE	44	2
WAVE	39	32
TAW	59	114
Pa11y	41	65

Tabell 7 visar resultatet från testningen av den första webbsidan från Region Skåne, en webbsida med praktisk information om Skånes Universitetssjukhus. Här hittades som förväntat betydligt mindre fel än hos webbsida 1:

- Axe identifierade 0 fel
- WAVE identifierade 6 fel
- TAW identifierade 32 fel
- Pa11y identifierade 0 fel

Antalet varningar skiljer sig återigen stort mellan verktygen:

- Axe identifierade 4 varningar
- WAVE identifierade 6 varningar
- TAW identifierade 171 varningar
- Pa11y identifierade 14 varningar

Tabell 7: Kvantitativ jämförelse av testresultat från webbsida 2, Skånes Universitetssjukhus – Praktisk Information

Verktyg	Antal fel	Antal varningar
AXE	0	4
WAVE	6	6
TAW	32	171

Pa11y	0	14

Slutligen i tabell 8 visas resultatet från testningen webbsida 3 – Utveckling i Region Skåne. Här identifierades totalt minst antal fel:

- Axe identifierade 1 fel
- WAVE identifierade 1 fel
- TAW identifierade 5 fel
- Pa11y identifierade 1 fel

Antalet varningar hos de olika verktygen varierar, TAW är återigen det verktyg med flest varningar:

- Axe identifierade 5 varningar
- WAVE identifierade 0 varningar
- TAW identifierade 51 varningar
- Pa11y identifierade 6 varningar

Tabell 8: Kvantitativ jämförelse av testresultat från webbsida 3, Region Skåne – Utveckling

Verktyg	Antal fel	Antal varningar
AXE	1	5
WAVE	1	0
TAW	5	51
Pa11y	1	6

Kvalitativ undersökning av fel på webbsida 1

Upptäckta fel på webbsida 1 hos de olika verktygen undersöktes för att förstå hur verktygen skiljer i typ av hittade fel och felklassificiering. Detta presenteras för felens tillhörande framgångskriterium och sammanfattas i tabell 9.

Framgångskriterium 1.1.1 Non-text Content (A)
Allt innehåll som inte är text ska alternativt kunna förmedlas med text. En del undantag finns om innehållet i fråga till exempel är en CAPTCHA, om innehållet är helt i dekorationssyfte. Samtliga verktyg upptäckte flertalet bilder utan alt-text och markerade det som ett fel på

webbsidan. WAVE delade ytterligare upp bilder utan alt-text till en ytterligare kategori – Spacer image without alt-text. Dessa är bilder som används i dekorativt syfte, men kanske ändå bör ha alt-text.

Framgångskriterium 1.3.1 Info and Relationships (A)
Information, struktur och relationer som förmedlas genom presentation kan programmässigt bestämmas eller vara tillgängligt i textform. Detta innebär att webbsidans struktur inte enbart får förmedlas genom utseende, utan måste tydligt anges i koden med relevanta etiketter.

För fel gällande framgångskriterium 1.3.1, så var TAW det verktyg som genererade flest fel. Både TAW och Pa11y identifierade en så kallad layout-tabell för formulär på webbsidan utan etiketter. Vidare identifierade både TAW och WAVE att användning av ett layout-table för formulär i sig kan vara ett fel då det generellt inte anses vara ett tillgängligt sätt att designa webbsidor. WAVE identifierade det som en varning snarare än ett direkt fel dock.

TAW identifierade också användning av taggar direkt i HTML-koden för att ändra utseendet på webbsidan som ett fel. Något de andra verktygen inte identifierade som fel.

Framgångskriterium 1.4.3 Contrast (Minimum) (A)
All visuell presentation av text och bilder av text måste ha ett kontrastförhållande på minst 4.5:1. Undantag finns om texten är tillräckligt stor, om texten är tillfällig och inte har i syfte att förmedla information eller en del av en logotyp.

Axe, WAVE och Pa11y identifierade att kontrasten på webbsidan var otillräcklig för att klara av framgångskriterium 1.4.3. TAW identifierade inte detta som ett problem, utan markerade det som något för en testare att manuellt undersöka.

Pa11y delade även upp denna typ av problem på bekräftade kontrastfel och potentiella kontrastfel. De som den markerade som bekräftade var kontrastskillnader mellan text och webbsidans bakgrund, medan de potentiella kontrastfelen var kontrast mellan bakgrundsbilder och text vilket verktyget inte kunde avgöra om de var ett problem.

Framgångskriterium 2.4.4 Link Purpose (In Context) (A)
Syftet med varje länk på webbsidan kan tydligt uppfattas med informationen i länk-texten eller med länk-texten och information i andra element som till exempel en header. Alla verktyg lyckades identifiera flertalet länkar utan tillgänglig text som beskriver dess syfte.

Framgångskriterium 3.1.1 Language of Page (A)
Språket på en webbsida ska kunna identifieras programmässigt. Detta innebär att webbsidans språk måste anges i HTML-kodens metadata. Samtliga testverktyg upptäckte att webbsidans språk saknades i dess metadata.

Framgångskriterium 4.1.2 Name, Role, Value (A)
Alla UI-element på en webbsida måste gå att programmässigt identifiera. Detta inkluderar bland annat knappar, formulärfält och länkar. Samtliga verktyg identifierade samma select-element på webbsidan som saknade ett namn som kunde läsas av programmässigt. WAVE identifierade detta som en varning snarare än ett direkt fel.

Tabell 9: Upptäckta fel med testverktyg

Typ av fel	WCAG-Kriterium	AXE	WAVE	TAW	Pa11y
Färgkontrast	1.4.3	Upptäckt	Upptäckt	Inte upptäckt	Upptäckt
Sidspråk	3.1.1	Upptäckt	Upptäckt	Upptäckt	Upptäckt
Bilder utan alt-text	1.1.1	Upptäckt	Upptäckt	Upptäckt	Upptäckt
Bilder för uppdelning av webbsida utan alt-text	1.1.1	Inte upptäckt	Upptäckt	Inte upptäckt	Inte upptäckt
Avsaknad av namn i select-element	4.1.2	Upptäckt	Upptäckt, men identifierat som problem	Upptäckt	Upptäckt
Länkar måste ha beskrivande namn	2.4.4	Upptäckt	Upptäckt	Upptäckt	Upptäckt

Formulär saknar HTML-etikett	1.3.1	Inte upptäckt	Inte upptäckt	Upptäckt	Upptäckt
Användning av layout-table för formulär	1.3.1	Inte upptäckt	Upptäckt, men identifierat som problem	Upptäckt	Inte upptäckt
Användning av taggar för att ändra utseende i HTML	1.3.1	Inte upptäckt	Inte upptäckt	Upptäckt	Inte upptäckt

4.3 Rekommendation av testverktyg

Utifrån den jämförelse som gjorts baserat de kriterier som har satts upp för detta examensarbete bedöms Pa11y vara det mest lämpade verktyget. Verktyget är det enda av de utvärderade verktygen som har helt öppen källkod och saknar kommersiella begränsningar. Det kan köras direkt via kommandoradstolken för snabb testning av en webbsida, men också integreras i en automatiserad testmiljö via Node.js, vilket gör det flexibelt och användbart i en större testprocess.

Verktyget kräver en viss teknisk kunskap vid installation, till skillnad från WAVE eller TAW som är mer användarvänliga. Dock finns det god dokumentation och tydliga instruktioner för hur miljön ska sättas upp. En viktig fördel är att Pa11y kan användas helt lokalt, utan att behöva skicka data till externa servrar, vilket gör det till ett säkert alternativ ur ett sekretessperspektiv.

Pa11y testar mot flera framgångskriterier på nivåerna A, AA och AAA för version 2.1 av WCAG, vilket är den version som nuvarande lagstiftning kräver att offentliga aktörer följer. En ytterligare fördel är att det utöver HTML_CodeSniffer även finns stöd för att köra tester med axe-core. Därmed ökar tillförlitligheten i resultaten då två olika testmotorer används.

I testningen av webbsidor angav Pa11y ett antal fel som var i linje med de andra verktygen, det gav även varningar på saker det inte kunde säkerställa som direkta fel. Det upptäckte inte två av de unika

felen i tabell 9, men hade över lag bland det bästa resultatet då det lyckades upptäcka fel som bland annat färgkontrast, avsaknad av alt-texter och att formulär saknar relevanta HTML-etiketter.

Pa11y som verktyg är passande för en stor organisation som Region Skåne då det har stora möjligheter att anpassas till flera ändamål. Det kan potentiellt köras i en testmiljö, antingen hos utvecklare eller hos testpersonal och på så sätt vara en övergripande kontroll innan webbsidor publiceras.

Då verktyget går att användas i egna skript och program är det flexibelt att köras på olika sätt. En användare kanske bara vill kolla en specifik webbsida som de har gjort ändringar på. Om organisationen vill ha en övergripande förståelse för hur webbtillgängligheten ser ut idag hos en webbdomän kan det, med lite arbete, implementera användbara skript eller program för det ändamålet.

4.4 Implementation av testverktyg med webcrawling

Detta kapitel beskriver resultatet av implementationen av Pa11y för användning på en hel webbdomän. 4.4.1 beskriver det resulterande programmets funktionalitet, 4.4.2 beskriver programmets struktur och datastruktur, 4.4.3 beskriver de begränsningar som finns i programmet.

4.4.1 Funktionalitet

Proof-of-concept programmet som detta examenarbete har resulterat med är ett program som tar en användares input i kommandoradstolken i form av en URL och y/n för två andra parametrar till main.js. Den första av dem är en parameter som avgör om användaren vill logga in på domänen innan den kör testningen och den andra är en parameter som bestämmer om en hel domän ska testas eller bara den angivna URL:en.

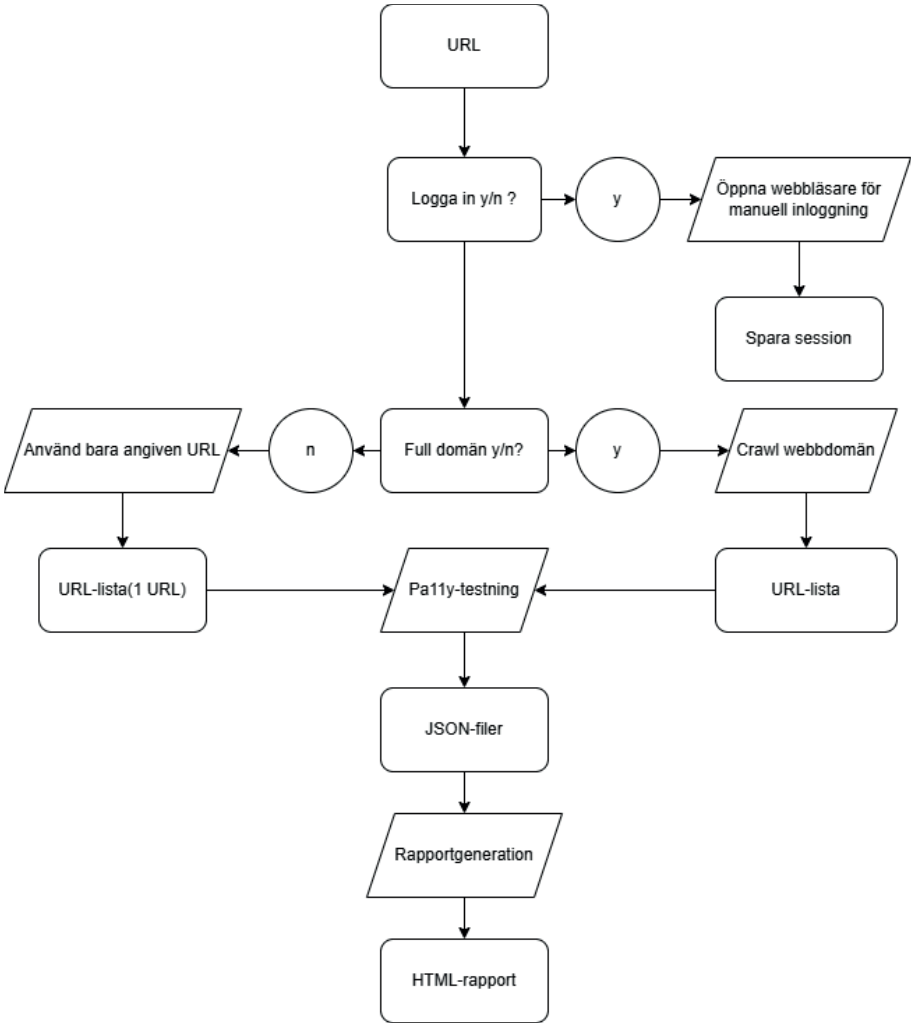
Om användaren vill undersöka en hel domän används funktioner för att utföra rekursiv webcrawling i puppeteerCrawl.js. Dessa funktioner läser av HTML-koden på den angivna webbsidan, hittar alla länkar till webbadresser som tillhör samma domän, och lägger till dem till en lista.

Funktionen gör sedan detsamma för nästa adress i listan och fortsätter tills den har undersökt alla unika webbadresser. Denna lista med unika URL:er sparas till en .txt-fil i testkörningens specifika mapp i resultatmappen. Om användaren valde att inte undersöka hela domänen läggs bara den specificerade adressen till i listan och ingen webcrawling sker.

Därefter används en funktion från pa11yAnalyze.js för att utföra analysen på alla adresser i listan. Där finns inställningar för vilken data som ska loggas om testresultatet samt vilka testmotorer som används, i detta fall både Pa11y och axe. Testresultatet för varje webbsida sparas i JSON-format i en unik fil för varje unik webbadress i testkörningens resultatmapp.

Nästa steg av programmet använder funktioner från generateReport.js. Detta steg ansvarar för att sammanfatta testkörningens resultat i en HTML-fil. Varje JSON-fil från testmappen analyseras och det totala antalet fel och varningar från testkörningen samt antalet fel och varningar per webbsida redovisas. Alla unika som har upptäckts under testkörningen presenteras i en lista med totalt hur många gånger de upptäcktes.

Programmet kan på så sätt genomföra automatiserad tillgänglighetstestning på stora domäner på 100+ HTML-sidor och sammanställa en rapport på webbdomänens tillgänglighetshälsa. En illustration av programmets flöde presenteras i figur 1.



Figur 1: Flödesschema för program

4.4.2 Programstruktur och datastruktur

Programmet är modulärt och består totalt fyra moduler: main.js, puppeteerCrawl.js, pa11yAnalyze.js och generateReport.js. Programmet körs genom att tillkalla main.js via Node.js och de övriga modulerna innehåller funktioner för att genomföra sin specifika uppgift.

Dessa filer ligger i projektmappen pa11y-webcrawl som går att laddas ner från projektets GitHub-sida. När programmet används för första

gången genereras en mapp i projektmappen för resultatfiler. I denna placeras varje unik testkörning i en unik mapp, med den angivna URL:en, datum och tid som namn. I testkörningens mapp hamnar resultatfilerna för varje URL i JSON-format, en lista med testade webbadresser i .txt-format samt testkörningens rapport i HTML-format.

Programmets filstruktur är alltså enligt figur 2:

```
↳ pa11y-webcrawl
       - main.js
       - puppeteerCrawl.js
       - pa11yAnalyze.js
       - generateReport.js
       ↳ resultat
             ↳ https_www_url_com_datum_tid_domain/page
             - https_www_url_com_datum.json
             - https_www_url_path_com_datum.json
             - urls.txt
             - rapport.html
```

Figur 2: Programmets filstruktur

Utöver dessa filer finns även metadatafiler som .gitignore, README.md och package.json, vilket har uteslutits från programmets visualisering för tydlighetens skull.

4.4.3 Begränsningar

Programmet är ett proof-of-concept och av väldigt simpel konstruktion för att demonstrera potentiella användningsområden hos pa11y som ett automatiserat testverktyg för tillgänglighet. Utöver de begränsningar som finns hos pa11y som testverktyg, finns även begränsningar hos programmets funktionalitet:

Programmet är inte användarvänligt – Det går att använda i kommandoradstolken, men förväntar sig exakta inputs av användaren. Programmet vill ha URL i formaten https://www.url.com eller http://www.url.com. Om användaren inte använder något av dessa format ger programmet ett felmeddelande och uppmanar användaren att ange en korrekt URL.

Ej headless-mode vid inloggning – Om en användare vill logga in på en domän öppnas puppeteer-webbläsaren i ett fönster i stället för i headless mode – Detta fönster måste förbli öppet under hela webcrawl- och testprocessen för att programmet ska fungera.

Programmet visar inte på vilken sida upphittade fel är i rapportfilen – En sammanfattning av hur många gånger varje unikt fel har upptäckts visas, men användaren måste själv undersöka JSON-filerna för att förstå vilka fel som upptäckts på varje webbsida.

Resultatfiler kan bli många och oorganiserade – För stora webbdomäner genereras många resultatfiler. Detta kan göra det svårt att manuellt gå igenom dem.

Programmet verifierar inte om inloggningen lyckades – Det förlitar helt på att användaren lyckades logga in.

Ingen automatisk felgruppering – Felen i rapporten presenteras i en lista utan att sorteras på något sätt.

Det finns ingen garanti att dynamiskt innehåll laddas – Dynamiskt JavaScript-innehåll som introducerar ny HTML-kod på webbsidor kan missas. Detta kan innebära att vissa webbadresser kan undgå att analyseras.

5. Slutsats och diskussion

Detta kapitel introducerar examensarbetes slutsats och diskussion. Avsnitt 5.1 är arbetets slutsats och beskriver huruvida dess problemformulering besvarades. Avsnitt 5.2 diskuterar bland annat arbetets relation till tidigare studier, skillnader mellan testverktyg och arbetets begränsningar.

Avsnitt 5.3 är en diskussion kring etiska dilemman och examensarbetets samhällsnytta. 5.4 diskuterar vidare utvecklingsmöjligheter.

5.1 Utvärdering av problemformulering

Resultatet av detta examensarbete är en analys och jämförelse av 4 olika automatiserade testverktyg för webbtillgänglighet, samt ett ”proof-of-concept”-program för webcrawling och analys med testverktyget Pa11y.

De problemformuleringar (1.4) som detta examensarbete utgick ifrån besvarades på ett tillfredsställande sätt. En förståelse för de regelverk (problem 2) som gäller för offentliga aktörer i Sverige beskrivs i arbetets inledande litteraturstudie. Hur DOS-lagen relaterar till EN 301 549 och WCAG 2.1 sammanfattas i kapitel 2.2. Att utgå från WCAG 2.1 nivå A och AA som gällande kriterier underlättande att hitta relevanta testverktyg då endast testverktyg med regler för detta togs i åtanke i urvalet.

Fyra verktyg valdes ut som kandidater till slutgiltig implementation: Axe, WAVE, TAW och Pa11y. Dessa analyserades efter kriterier som fastställdes efter litteraturstudien, intervjuer och diskussion med handledare på Region Skåne (problem 3), i ett försök att spegla de behov på tillgänglighetstestning organisationen har.

I slutändan valdes Pa11y ut av examensarbetaren som det mest passande verktyget (problem 4), då det presterade bra utifrån de kriterier som bestämts och har utmärkta möjligheter att skräddarsys för flera olika ändamål. Det kan användas både för snabb testning av enskilda sidor eller för omfattande testning av flera sidor samtidigt. Det som kan begränsa anpassningen av Pa11y är den tid och resurser

som behövs för att genomföra en implementation som specifikt passar Region Skånes behov.

Utvecklare och testpersonal inom Region Skåne skulle kunna använda Pa11y i olika skeden av utvecklingsprocessen (problem 5). En utvecklare kan exempelvis använda verktyget i samband med publicering eller uppdatering av webbsidor, för att snabbt identifiera större fel. Om testansvaret i stället ligger hos särskild testpersonal, kan de använda Pa11y för att undersöka om baskriterier uppfylls tillsammans med den nödvändiga manuella testningen.

Då ”proof-of-concept” programmet har demonstrerat att webcrawling och analys av webbsidor med Pa11y är möjlig finns även möjlighet för Region Skåne att kunna testa hela domäner på en gång efter tillgänglighetsproblem. Detta kan vara användbart för att undersöka det stora antal webbsidor hos Region Skåne som redan är offentliga.

Det första problemet som introducerades i problemformuleringen (problem 1) – Hur testning av webbtillgänglighet ser ut för närvarande, diskuteras i avsnitt 5.2.4 *Användning hos Region Skåne*.

5.2 Diskussion

Detta diskussionsavsnitt består av 5.2.1 Jämförelse med tidigare studier, 5.2.2 Skillnader mellan testverktyg, 5.2.3 Val av Pa11y, 5.2.4 Användning hos Region Skåne, 5.2.5 Reflektion kring implementation och 5.2.6 Begränsningar.

5.2.1 Jämförelse med tidigare studier

Det finns redan ett antal studier som jämför testresultat från olika testverktyg på olika webbsidor. Många av dessa är dock av mer kvantitativ typ och jämför endast antal identifierade fel mellan verktygen. Sådana jämförelser kan vara svåra att tolka, eftersom verktygen skiljer sig åt i hur de klassificerar och rapporterar fel.

I detta examensarbete har, utöver en kvantitativ jämförelse, en kvalitativ analys av testresultat genomförts, med fokus på att undersöka vilka typer av fel som har upptäckts. Detta ger en djupare förståelse av verktygens funktionalitet.

Denna skillnad i klassificering är något som uppenbarade sig i detta examensarbetes analys. Axe, WAVE och Pa11y var väldigt lika varandra i vilka fel de hittade, men kunde ofta ge olika resultat på om felet var allvarligt eller ej. TAW identifierade mycket fel och potentiella fel, men många av dessa berodde på att verktyget inte kunde avgöra om det faktiskt rörde sig om fel.

För automatiserade testverktyg är det viktigt att kunna identifiera faktiska fel. Mer information än så kan enkelt bli överflödig och göra analysen mer ofokuserad. Att veta vilka typer av fel verktyget kan hitta och vilka typer av fel en människa manuellt måste hitta är viktigt. TAW är mer fokuserat på att användas på enskilda webbsidor och kan därför i stället vara ett hjälpmedel för den manuella testningen.

Pa11y ger klara besked om vad som är uppenbart fel och vad som är en varning. Användaren kan dessutom anpassa resultatet genom att välja att enbart visa faktiska fel. Det är även möjligt att använda testmotorn Axe i kombination med HTML_CodeSniffer, vilket ökar chansen att fler fel upptäcks.

5.2.2 Skillnader mellan testverktyg

De automatiserade testverktyg som undersöktes i analysdelen av examensarbetet skiljde mycket i hur de uppfyllde de kriterium som ställdes. Axe och Pa11y som båda har en open-source-aspekt var de verktyg som genast visades vara bättre för en organisation för att på eget sätt implementera automatiserad testning av webbtillgänglighet.

De kriterium som undersöktes i analysen var *Kostnad, Flexibilitet, Användarbarhet, Säkerhet* och *Täckning* och definieras tillsammans med deras underkriterier i avsnitt 3.3. Skillnaderna mellan de olika verktygen var stora:

Kriterium 1, Kostnad hade väldigt olika resultat för testverktygen. Pa11y är helt open-source utan någon kommersiell aspekt, vilket gynnar verktyget i prestation enligt kriteriet. Även Axe går att använda gratis i sin open-sourceform men har en viss kommersiell aspekt då Deque Systems som backar källkoden är ett företag som erbjuder betaltjänster baserade på kodbasen. Detta kan upplevas som en potentiell begränsning då företaget skulle kunna begränsa

användarbarheten av verktyget för att uppmana företag och organisationer att använda betalversionerna.

WAVE:s gratisversion i webbläsaren är väldigt begränsad gällande nödvändig funktionalitet för att kunna skräddarsys för en stor organisations ändamål. Hade verktyget presterat betydligt bättre än de andra kandidaterna i andra kriterier och i testresultatet hade kanske dess begränsningar gällande kostnad inte varit en lika stor nackdel.

Även TAW:s gratisversion är begränsad för att kunna användas för storskalig testning, men är fullt duglig för att testa enskilda webbsidor. TAW CMS kan potentiellt erbjuda en mer flexibel lösning, men då priset för användning är oklart och dokumentation otillräcklig kan det inte bekräftas. Organisationen bakom TAW, CTIC kontaktades enligt kontaktformulär för information om detta men ingen respons kom.

Även kriterium 2, Flexibilitet gynnar open-sourceverktygen. Axe och Pa11y visades vara väldigt flexibla gällande hur de går att använda och hur man på egen hand kan implementera testning. Även WAVE har i dess API-versioner möjlighet att vara flexibelt nog att användas av en större organisation. Det är då kostnader för licens eller per API-anrop som dock begränsar huruvida denna flexibilitet går att uppnås.

TAW är väldigt begränsat enligt kriterium 2 då det inte erbjuder någon form av egen implementation. Dess standalone-version har bara tester för WCAG 1.0, vilket inte gör den versionen speciellt önskvärd.

Verktygens användarbarhet (kriterium 3) handlade om huruvida verktygen kräver någon särskild teknisk kompetens för att användas och om de har god dokumentation. WAVE som webbläsartillägg och TAW var de verktyg som var enklast att direkt använda. Axe och Pa11y behöver en viss anpassning till passande skript för att användas av nya användare. Proceduren att börja använda verktygen var dock relativt enkel om man följde dokumentationen på GitHub. Det är något både Pa11y och Axe har gott om, detaljerad dokumentation. Både för setup och verktygens användningsmöjligheter.

För kriterium 4, Säkerhet var att ge en organisation möjlighet att testa lokalt och med att så lite data som möjligt går till externa servrar något som önskades. Pa11y och Axe uppfyllde båda underkriterierna,

medan WAVE går att använda lokalt, men den version som tar betalt per API-anrop behöver skicka data till en extern server för att användas. Licensversionen däremot går att användas helt lokalt utan extern kommunikation. TAW är inte ett passande verktyg i säkerhetsaspekt då det inte går att använda lokalt och behöver kommunikation med en extern server för att användas.

Verktygen har tester för mellan 20–28 framgångskriterier på minst AA-nivå för WCAG 2.1, vilket är nödvändigt för efterlevnad av DOS-lagens krav. Verktygens tester undersöker dock inte fullt ut dessa framgångskriterier utan många har ändå ett stort behov av mänsklig verifiering. Dessa verktyg bör alltså användas för att snabbt undersöka tillgängligheten på webbsidor för att ge en inblick i om det finns tydliga brister.

Även om en webbsida inte har några direkta fel enligt de automatiserade testverktygen måste ändå en människa kontrollera om andra framgångskriterier följs. För att kunna åstadkomma detta krävs att utvecklare eller testpersoner i en organisation har den nödvändiga kompetensen inom WCAG:s framgångskriterier.

Axe, WAVE och Pa11y visade relativt lika testresultat i den kvantitativa jämförelsen. WAVE visade lite fler fel och varningar per sida. TAW var det verktyg som stod ut och visade ett mycket större antal fel och varningar på varje webbsida som undersöktes. En stor orsak till detta är att TAW ofta ger varningar för saker den inte kan bekräfta som fel eller inte, samt ofta flaggar samma HTML-element med flera olika varningar.

De andra verktygen är mer sparsamma med sådan typ av flaggning, något som passar bättre för en automatiserad testprocess. Ett stort antal fel och varningar blir ett överflöd av data som gör processen att verifiera vad som är fel svårare.

I den kvalitativa jämförelsen av testresultat ser vi typen av fel dessa automatiserade verktyg kan upptäcka. Det är endast tydliga fel som går att utläsas från webbsidans HTML-kod. Exempelvis dålig färgkontrast, avsaknad av sidspråk och bilder som saknar alt-texter är brister verktygen uppmanar användaren att åtgärda. Denna typ av fel är relativt enkla att åtgärda och en duktig webbutvecklare borde

undvika att begå sådana misstag. Om utvecklaren har fått någon form av utbildning om lagkrav så bör de veta om vad som krävs av dem.

5.2.3 Val av Pa11y

Valet av Pa11y som testverktyg verkade självklart enligt de kriterier som tagits fram. Det har inga kostnadsaspekter och utvecklas aktivt enligt verktygets GitHub-sida. Det var relativt enkelt att sätta upp på en dator för att kunna använda för enkel testning av webbsidor i kommandoradstolken.

Även mer komplicerad användning av verktyget visades inte vara allt för komplicerad då implementationen av det i kombination med webcrawling fungerade bra. I handen på en erfaren utvecklare går det nog att användas på flertalet olika sätt och dess förmåga att kunna använda axe-core utöver HTML_CodeSniffer gjorde så att det finns ytterligare säkerhet i att så många fel som möjligt upptäcks.

5.2.4 Användning hos Region Skåne

Detta examenarbete utfördes i samarbete med testcenter på Region Skåne. Det var därför viktigt att arbetet gav någon form av nytta åt organisationen. I intervjuer med utvecklare på Region Skåne och diskussion med personal som arbetade med en utvärdering av organisationens förmåga att testa för tillgänglighet var det uppenbart att det inte fanns en organiserad taktik för verifiering.

I varje projekt inom organisationen ansvarar för närvarande tjänst- och informationsägare över hur testningen går till. Det innebär att testningen sker på väldigt olika sätt och med varierande resultat. Testcenter, som hanterar bland annat teststrategi och testning kan stödja olika projekt, genom att erbjuda resurser för testning, något som bara sker på begäran från projektansvariga.

Region Skåne arbetar för närvarande med att ta fram en plan för hur man kan förbättra tillgänglighetstestningen internt. Resultatet av detta förväntas vara en mer enhetlig och standardiserad strategi som bör leda till en förbättrad tillgänglighet i de tjänster som erbjuds av organisationen.

Användning av automatiserade testverktyg som till exempel Pa11y är en bra resurs för organisationen i dess testprocess. Verktyget i dess grundform kan användas av utvecklare för att snabbt testa enskilda webbsidor i arbetsprocessen eller i en mer komplicerad implementering användas av testpersonal för att undersöka nutida tillgänglighetsnivå hos organisationens webbdomäner.

Examensarbetets implementation av testning och webcrawling kan användas för att ge en slags hälsorapport kring en domäns tillgänglighet. Om flera fel hittas på någon webbplats innebär det nog att det finns andra fel som behövs undersökas och åtgärdas manuellt.

5.2.5 Reflektion kring implementation

Implementationen av det valda testverktyget Pa11y i kombination med webcrawling visade sig vara mindre komplext än förväntat. Trots begränsad erfarenhet av JavaScript kunde en fungerande lösning snabbt utvecklas, mycket tack vare den omfattande och tydliga dokumentationen från Pa11y:s officiella webbplats och GitHub.

Det var en större utmaning att implementera en tillförlitlig webcrawler. I nuvarande implementation finns det potentiell risk att missa vissa webbsidor i crawlprocessen om det finns mycket dynamiskt innehåll. Funktioner för att vänta på att innehåll laddas och för att automatisk scrollning har integrerats, men eftersom webcrawlern är utformad för att hantera generella domäner är det svårt att säkerställa full täckning.

Vid anpassning för en specifik domän hade det varit möjligt att ta hänsyn till specifika HTML-element på domänen för att hitta specifika ID:n för knappar för att läsa in dynamiskt innehåll.

Eftersom syftet var att implementera ett proof-of-concept program för att demonstrera hur ett automatiserat verktyg kan användas, bedöms resultatet som tillfredställande. Med detta arbete som grund, med små anpassningar, kan mer erfarna utvecklare inom Region Skåne vidareutveckla programmet för att passa organisationens ändamål.

5.2.6 Begränsningar

Utöver de begränsningar som finns hos proof-of-concept implementationen som beskrivs i avsnitt 4.4.3 har examensarbetet över lag en del begränsningar.

Examensarbetet utfördes utan budget för införskaffning av automatiserade testverktyg. Detta kan potentiellt begränsa vissa verktygs förmåga att prestera enligt de kriterier som satts. Att inte använda betalversioner av vissa verktyg bör dock inte ha påverkat verktygens prestation gällande antal upphittade fel och varningar i avsnitt 4.2.6.

Förmågan att testa dessa verktygs prestation gällande kriterierna *Flexibilitet* och *Användarbarhet* kan dock ha påverkats. Detta har dock tagits i åtanke vid val av slutkandidat till implementation.

De verktyg som undersöktes representerar endast en mindre del av det totala antalet tillgängliga automatiserade testverktyg för webbtillgänglighet. En studie av ett större antal verktyg hade potentiellt inneburit fler potentiella slutkandidater. Examensarbetet var begränsat i tid och ett mindre antal verktyg valdes ut för att ge en mer fokuserad analys.

5.3 Etisk reflektion

Detta examensarbete har lett till en fördjupad förståelse och uppskattning för hur komplext det är att designa webbplatser som fungerar för alla. Ju mer information som granskats om testverktyg och relevant lagstiftning, desto tydligare har det blivit att webbtillgänglighet handlar om mer än att bara uppfylla juridiska krav. Det handlar om att möjliggöra jämlikt deltagande i samhället.

Därför är det viktigt att organisationer inte bara ser de lagkrav som finns som en slags checklista att komma förbi i arbetsprocessen. I stället bör myndigheter, organisationer och företag sträva efter att göra mer än vad som krävs. Genom att utvidga förståelsen om området hos personal och ledning ökar även drivet att leverera tjänster som är så tillgängliga som möjligt.

Förhoppningen är att denna rapport kan inspirera någon organisation att börja använda automatiserade testverktyg för att underlätta testningen av webbtillgänglighet. Men även att denna rapport uppmuntrar organisationer att internt öka förståelsen om varför digital tillgänglighet är viktig och nödvändig i samhället.

Implementationen av testverktyget Pa11y beskrivs övergripande i rapporten. Kodexempel och källkod inkluderas inte, eftersom implementation menas att överlåtas till Testcenter på Region Skåne efter arbetets avslut. Tanken är att Testcenter eventuellt kommer att använda lösningen i sin framtida testverksamhet. Ansvar för vidareutveckling och delning av kodbasen ligger därför hos Region Skåne, vilket är orsaken till att detaljerad kod ej redovisas i rapporten.

5.4 Framtida utvecklingar

I framtiden kommer vi att se ett utökat behov av tillgänglighetstestning för webbsidor. *Lagen om vissa produkter och tjänsters tillgänglighet* träder i kraft den 28:e juni 2025 och kommer att innebära att privata företag och organisationer också kommer att behöva tillgodose de krav som finns på tillgängliga webbsidor.

Testverktyg som de som har undersökts i detta examensarbete kommer att vara en bra resurs att använda för ägare av webbplatser för att snabbt undersöka om man klarar vissa minimikrav. En kompetensutveckling inom området digital tillgänglighet kommer att behövas och många gamla webbsidor kommer behövas anpassas till nuvarande lagstiftning.

Mer omfattade analys av automatiserade testverktyg behövs. Detta examensarbete gör ett försök att kvalitativt undersöka vilka typer av fel olika verktyg hittar. Denna undersökning är dock begränsad till sitt tillvägagångssätt av att jämföra resultat på olika webbsidor. Det skulle vara mer intressant att noggrant jämföra de testregler som finns hos verktygen för att få en detaljerad uppskattning av exakt vilka delar som testas hos varje verktyg. På så sätt kan man se exakt vilka aspekter hos en webbplats varje verktyg kan undersöka.

Detta examensarbetes implementation av ett testverktyg med webcrawling är väldigt simpel. Om en organisation är seriös om att testa domäner på ett liknande sätt finns det många förbättringssätt.

Det första är att undersöka sina egna domäner och standardisera dem så att det enkelt går att implementera metoder för att kunna ladda dynamiskt innehåll via kod. På så sätt kan man garantera att ett liknande program kan ladda alla HTML-element. Det bör garantera att hela domänen blir undersökt och att inget missas.

Dessutom är bättre visualisering av testresultat något som kan utvecklas. Den genererade rapporten i detta examensarbetes implementation är väldigt enkel. Det är möjligt att förbättra den så användaren får mer organiserad information om de olika fel som finns på webbdomänen.

6. Terminologi

Detta avsnitt introducerar och förklarar några begrepp som används i denna rapport.

- CLI – Förkortning av Command Line Interface, på svenska: kommandoradstolk.
- CMS – Förkortning av Content Management System, på svenska: innehållshanteringssystem. Ett utvecklingssystem för webbsidor där användaren designar webbsidor i ett användargränssnitt i stället för i kod.
- CI/CD – Förkortning av Continuous Integration och Continuous Delivery. Ett arbetssätt inom mjukvaruutveckling, där kod automatiskt testas.
- DOS-Lagen – Sveriges lagstiftning om digital tillgänglighet för offentliga aktörer.
- EN 301 549 – EU-standard för tillgänglighet.
- ETSI – Förkortning av European Telecommuncations Standards Institute. Organisationen som har tagit fram standarden EN 301 549.
- Headless-mode – Läge i Puppeteer där webbläsaren körs i bakgrunden utan ett synligt fönster.
- IKT – Förkortning av Informations- och telekommunikationsteknik.
- W3C – Förkortning av World Wide Web Consortium. Organisationen som har tagit fram WCAG.
- WCAG – Förkortning av Web Content Accessibility Guidelines. En internationell standard för webbtillgänglighet.
- Webcrawling – Processen att läsa in länkar på en webbsida för att kunna navigera till flera webbsidor.

7. Källförteckning

De källor som har använts till detta examensarbete specificeras i 7.1 Digitala källor och 7.2 Fysiska källor.

7.1 Digitala källor

[1] Myndigheten för delaktighet ”Statistik om personer med funktionsnedsättning” MFD, [Online]. Tillgänglig: https://www.mfd.se/uppfoljning-och-statistik/statistik-om-personer-med-funktionsnedsattning/ [Hämtad: 3 feb. 2025]
[2] Statistikdatabasen ”Undersökningarna av levnadsförhållanden. Funktionsnedsättning. Andel personer i procent efter indikator, redovisningsgrupp, kön och åldersintervall” SCB, [Online]. Tillgänglig: https://www.statistikdatabasen.scb.se/pxweb/sv/ssd/START__LE__LE0101__LE0101H/LE01012021H02/table/tableViewLayout1/ [Hämtad: 3 feb. 2025]
[3] SCB, ”Hälsa 2022” Levnadsförhållanden 2023:4, Stockholm, Sverige. Tillgänglig: https://www.scb.se/contentassets/d07415fd4b284056acd7027e96ccd041/le0101_2022a01_br_lebr2304.pdf [Hämtad: 3 feb. 2025]
[10] World Wide Web Consortium (W3C), ”About us”, [Online]. Tillgänglig: https://www.w3.org/about/ [Hämtad: 8 feb. 2025]
[11] Myndigheten för digital förvaltning (DIGG), ”Det här är EN 301 549 och WCAG”, [Online]. Tillgänglig: https://www.digg.se/webbriktlinjer/lagar-och-krav/det-har-ar-en-301-549-och-wcag [Hämtad: 8 feb. 2025]
[12] World Wide Web Consortium (W3C) ”WCAG 2 Overview”, [Online]. Tillgänglig: https://www.w3.org/WAI/standards-guidelines/wcag/ [Hämtad: 8 feb. 2025]
[13] World Wide Web Consortium (W3C) ”Web Content Accessibility Guidelines (WCAG) 2.2”, [Online]. Tillgänglig: https://www.w3.org/TR/WCAG22/ [Hämtad: 8 feb. 2025]
[14] World Wide Web Consortium (W3C) ”Introduction to Understanding WCAG 2.2”, [Online]. Tillgänglig: https://www.w3.org/WAI/WCAG22/Understanding/intro#understanding-the-four-principles-of-accessibility [Hämtad: 8 feb. 2025]
[15] European Parliament and Council of the European Union, Directive (EU) 2016/2102 *on the accessibility of the websites and mobile applications of public sector bodies*, Official Journal of the European Union, L 327/1, dec. 2, 2016 [Online] Tillgänglig: https://eur-lex.europa.eu/eli/dir/2016/2102/oj/eng [Hämtad: 9 feb. 2025]

[16] European Telecommunications Standards Institute (ETSI), EN 301 549 V3.2.1 (2021–03): *Accessibility Requirements for ICT products and services,* ETSI, Mar 2021. [Online] Tillgänglig: https://www.etsi.org/deliver/etsi_en/301500_301599/301549/03.02.01_60/en_301549v030201p.pdf [Hämtad: 9 feb. 2025]
[17] Myndigheten för digital förvaltning (DIGG), ”Om lagen om tillgänglighet till digital offentlig service”, [Online] Tillgänglig: https://www.digg.se/analys-och-uppfoljning/lagen-om-tillganglighet-till-digital-offentlig-service-dos-lagen/om-lagen [Hämtad: 10 feb. 2025]
[18] Regeringskansliet, ”Lag (2018:1937) om tillgänglighet till digital offentlig service”, [Online] Tillgänglig: https://rkrattsbaser.gov.se/sfst?bet=2018:1937 [Hämtad: 10 feb. 2025]
[19] Regeringskansliet, ”Förordning (2018:1938) om tillgänglighet till digital offentlig service” [Online] Tillgänglig: https://rkrattsbaser.gov.se/sfst?bet=2018:1938 [Hämtad 10 feb. 2025]
[20] Myndigheten för digital förvaltning (DIGG), ”WCAG har uppdaterats – vad innebär det för dig som offentlig aktör?”, [Online] Tillgänglig: https://www.digg.se/om-oss/nyheter/digital-tillganglighet/nyheter/2023-10-12-wcag-har-uppdaterats---vad-innebar-det-for-dig-som-offentlig-aktor [Hämtad 25 feb. 2025]
[21] Deque Systems, ”axe-core”, GitHub [Online] Tillgänglig: https://github.com/dequelabs/axe-core [Hämtad: 25 feb. 2025]
[22] Deque Systems, ”axe-core API documentation”, GitHub [Online] Tillgänglig: https://github.com/dequelabs/axe-core/blob/develop/doc/API.md [Hämtad: 26 feb. 2025]
[24] WebAIM, ”WAVE Help”, [Online] Tillgänglig: https://wave.webaim.org/help [Hämtad: 28 feb. 2025]
[25] CTIC, ”TAW index”, [Online] Tillgänglig: https://www.tawdis.net/index [Hämtad: 1 mar. 2025]
[26] Pa11y ”Pa11y”, GitHub [Online] Tillgänglig: https://github.com/pa11y/pa11y?tab=readme-ov-file [Hämtad: 2 mar. 2025]
[27] Pa11y ”License”, GitHub [Online] Tillgänglig: https://github.com/pa11y/pa11y/blob/main/LICENSE [Hämtad: 2 mar. 2025]
[28] World Wide Web Consortium (W3C) ”Web Content Accessibility Guidelines (WCAG) 3.0” [Online] Tillgänglig: https://www.w3.org/TR/wcag-3.0/ [Hämtad: 2 mar. 2025]
[29] Deque Systems ”Plans” [Online] Tillgänglig: https://axe.deque.com/axe-devtools/plans?utm_medium=referral&utm_source=deque.com&utm_campaign=axe [Hämtad 13 mar. 2025]
[30] WebAIM ”WAVE API” [Online] Tillgänglig: https://wave.webaim.org/api/ [Hämtad: 13 mar. 2025]
[31] WebAIM ”WAVE Stand-alone API and Testing Engine [Online] Tillgänglig: https://wave.webaim.org/standalone [Hämtad: 13 mar. 2025]

[32] CTIC ”TAW Tools” [Online] Tillgänglig: https://www.tawdis.net/proj?lang=en [Hämtad: 13 mar. 2025]
[33] Deque Systems ”axe-devtools-robot API Reference” [Online] Tillgänglig: https://docs.deque.com/devtools-for-web/4/en/py-api-robot [Hämtad: 14 mar. 2025]
[34] WebAIM ”WAVE Browser Extensions” [Online] Tillgänglig: https://wave.webaim.org/extension/ [Hämtad 18 mar. 2025]
[35] Pa11y ”Pa11y Dashboard”, GitHub [Online] Tillänglig: https://github.com/pa11y/pa11y-dashboard [Hämtad: 18 mar. 2025]
[36] Pa11y ”Pa11y CI [Online] Tillgänglig: https://github.com/pa11y/pa11y-ci [Hämtad: 18 mar. 2025]
[37] Deque Systems ”Welcome to axe DevTools for Web” [Online] Tillgänglig: https://docs.deque.com/devtools-for-web/4/en/welcome-axe-devtools [Hämtad: 19 mar. 2025]
[38] WebAIM ”WAVE API (version 3.1) Details” [Online] Tillgänglig: https://wave.webaim.org/api/details [Hämtad: 19 mar. 2025]
[39] Pa11y ”Pa11y Tutorials” [Online] Tillgänglig: https://pa11y.org/tutorials/ [Hämtad: 19 mar. 2025]
[40] Deque Systems ”Offline Mode” [Online] Tillgänglig: https://docs.deque.com/devtools-for-web/4/en/devtools-offline [Hämtad: 22 mar. 2025]
[41] CTIC ”Terms of use” [Online] Tillgänglig: https://www.tawdis.net/condiciones [Hämtad: 23 mar. 2025]
[42] Deque Systems ”Rulesets” [Online] Tillgänglig: https://docs.deque.com/devtools-for-web/4/en/rulesets [Hämtad: 28 mar. 2025]
[43] Deque Systems ”Rule Descriptions” GitHub [Onlibe] Tillgänglig: https://docs.deque.com/devtools-for-web/4/en/rulesets [Hämtad: 28 mar. 2025]
[44] WebAim ”Wave Documentation” [Online] Tillgänglig: https://wave.webaim.org/api/docs?format=html [Hämtad: 28 mar. 2025]
[45] CTIC ”Index” [Online] Tillgänglig: https://www.tawdis.net/index [Hämtad: 28 mar. 2025]
[46] Squizlabs, ”HTML_CodeSniffer, WCAG 2.1 Standard: Summary” [Online] Tillgänglig: https://squizlabs.github.io/HTML_CodeSniffer/Standards/WCAG2/ [Hämtad: 28 mar. 2025]
[47] W3C. ”Introduction to Techniques for WCAG 2.0” [Online] Tillgänglig: https://www.w3.org/TR/2008/NOTE-WCAG20-TECHS-20081211/intro.html [Hämtad: 21 apr. 2025]
[48] Deque Systems, ”List of Axe HTML versions” [Online] Tillgänglig: https://dequeuniversity.com/rules/axe/html [Hämtad: 21 apr. 2025]

[49] Post- och telestyrelsen (PTS) ”Lagen om vissa produkter och tjänsters tillgänglighet” [Online] Tillgänglig: https://pts.se/digital-inkludering/lagen-om-vissa-produkters-och-tjansters-tillganglighet/ [Hämtad: 9 maj. 2025]
[50] Post- och telestyrelsen (PTS), ”Frågor och svar om tillgänglighetslagen” [Online] Tillgänglig: https://pts.se/digital-inkludering/lagen-om-vissa-produkters-och-tjansters-tillganglighet/vanliga-fragor-och-svar-om-tillganglighetslagen/ [Hämtad: 9 maj. 2025]

7.2 Fysiska källor

[4] J. Dias, D. Carvalho, A. Reis, J. Barroso and T. Rocha, "Identifying Concerns On Automatic Tools For Accessibility Assessment: A Comparative Case Study On Two Portuguese Universities' Websites*," 2022 5th International Conference on Information and Computer Technologies (ICICT)*, New York, NY, USA, 2022, pp. 1–6
[5] S. Lawton Henry, ”Understanding Web Accessibility” in *Web Accessibility – Web Standards and Regulatory Compliance,* Springer-Verlag: New York, 2006, pp. 2-51
[6] C.C. Shelly,” Inclusion*”* in *Web Acessibility – A Foundation for Research*, Y. Yesilada, S. Harper Eds. Springer: London 2019, pp 301–307
[7] H. Nicolau, K. Montague,” Assistive Technologies” in *Web Accessibility – A Foundation for Research,* Y. Yesilada, S. Harper Eds. Springer-Verlag: London, 2019, pp 317–331
[8] A. Kirkpatrick,” Overview of Accessible Technologies” in *Web Accessibility – Web Standards and Regulatory Compliance*, Springer-Verlag: New York, 2006, pp 86–100
[9] J. Thatcher,” Assistive Technology: Screen Readers and Browsers” in *Web Accessibility – Web Standards and Regulatory Compliance,* Springer-Verlag: New York, 2006, pp 104–124
[23] S. U. Dongaonkar, R. S. Vadali, C. Dhutadmal ”Accessibility Analyzer: Tool for New Adaptations in Government Web Applications to Improbe Accessibility”, *2017 International Conference on Computing, Communication, Control and Automation (ICCUBEA),* Pune, India, 2017, pp 1-5

8. Appendix

8.1 Bilaga 1: Intervjuguide

Denna intervjuguide användes till intervjudelen av projektet. Samtliga intervjupersoner fick samma frågor och intervjuerna genomfördes via Microsoft Teams. Alla personer som intervjuades fick frågorna i förhand så de var förberedda på intervjuns innehåll.

Bakgrund

1. Kan du berätta lite om din bakgrund och erfarenhet inom just tillgänglighetsanpassning?

Tillgänglighet i praktiken

2. Hur ser du på utvecklingen av digital tillgänglighet i Sverige och EU de senaste åren och framöver?
3. Vilka ser du som de största utmaningarna med att göra digitala tjänster tillgängliga idag?
4. Vilka faktorer är viktigast att tänka på vid design och utveckling av webbsidor för att de ska vara tillgänglighetsanpassade?
5. Finns det några vanliga problem du ofta ser på svenska myndigheter, organisationer och företags webbplatser gällande just deras tillgänglighet?

Inkludering

6. Hur påverkar bristande tillgänglighet olika användargrupper med funktionsnedsättningar i samhället?

Lagar och riktlinjer

7. Hur väl efterlevs en 301 549 och WCAG idag på svenska webbplatser? Har det skett en förbättring de senaste åren?
8. Vad tycker du om nuvarande lagstiftning? Är den tillräcklig eller finns det områden där den borde stärkas?

Automatiserad testning

9. Vad tycker du om automatiserad testning med hjälp av testverktyg som AXe och Wave, inom just tillgänglighetsanpassning av webbsidor?
10. Hur brukar företag och myndigheter integrera automatiserad testning i sina arbetsprocesser.

Avslutning

11. Om du fick ge en organisation tre konkreta råd för att förbättra sin digitala tillgänglighet, vilka skulle det vara?

8.2 Bilaga 2: Framgångskriterier med tester

Framgångskriterium	Axe	TAW	WAVE	Pa11y
1.1.1 (A)	X	X	X	X
1.2.1 (A)			X	
1.2.2 (A)	X		X	
1.2.3 (A)			X	
1.2.4 (AA)				
1.2.5 (AA)			X	
1.2.6 (AAA)				
1.2.7 (AAA)				
1.2.8 (AAA)				
1.2.9 (AAA)				
1.3.1 (A)	X	X	X	X
1.3.2 (A)		X	X	
1.3.3 (A)		X*		
1.3.4 (AA)				
1.3.5 (AA)	X			X
1.3.6 (AAA)				
1.4.1 (A)	X	X*		X
1.4.2 (A)	X		X	
1.4.3 (AA)	X	X*	X	X
1.4.4 (AA)	X			
1.4.5 (AA)		X*		
1.4.6 (AAA)	X	X*		X
1.4.7 (AAA)				
1.4.8 (AAA)		X*		
1.4.9 (AAA)		X*		
1.4.10 (AA)				
1.4.11 (AA)				
1.4.12 (AA)	X			
1.4.13 (AA)				
2.1.1 (A)	X	X	X	X
2.1.2 (A)		X*	X	X
2.1.3 (AAA)	X			
2.1.4 (A)				
2.2.1 (A)	X	X*	X	X
2.2.2 (A)	X	X*	X	X
2.2.3 (AAA)		X*		
2.2.4 (AAA)				
2.2.5 (AAA)		X*		
2.2.6 (AAA)				
2.3.1 (A)		X		

2.3.2 (AAA)		X		
2.3.3 (AAA)				
2.4.1 (A)	X	X	X	X
2.4.2 (A)	X	X	X	X
2.4.3 (A)		X	X	
2.4.4 (A)	X	X	X	
2.4.5 (AA)		X*		
2.4.6 (AA)		X	X	
2.4.7 (AA)		X*		
2.4.8 (AAA)		X*		X
2.4.9 (AAA)	X	X		
2.4.10 (AAA)		X		
2.5.1 (A)				
2.5.2 (A)				
2.5.3 (A)				X
2.5.4 (A)				X
2.5.5 (AAA)				
2.5.6 (AAA)				
3.1.1 (A)	X	X	X	X
3.1.2 (AA)	X	X*		X
3.1.3 (AAA)		X*		
3.1.4 (AAA)		X*		
3.1.5 (AAA)		X*		
3.1.6 (AAA)		X*		X
3.2.1 (A)		X*		
3.2.2 (A)		X*	X	X
3.2.3 (AA)		X*		
3.2.4 (AA)		X*		
3.2.5 (AAA)	X	X		X
3.3.1 (A)				
3.3.2 (A)	X		X	
3.3.3 (AA)				
3.3.4 (AA)				
3.3.5 (AAA)				
3.3.6 (AAA)				
4.1.1 (A)				X
4.1.2 (A)	X	X	X	X
4.1.3 (AA)				

Printed by Tryckeriet i E-huset, Lund 2025

Series of Bachelor`s theses
Department of Electrical and Information Technology
LU/LTH-EIT 2025-1062
http://www.eit.lth.se