Miljömässig hållbarhet inom mjukvaruutveckling

En enfallsstudie om mjukvaruutövares uppfattningar på ett IT-företag i Sverige

SIMON PERSSON 2024
MVEM31 EXAMENSARBETE FÖR MASTEREXAMEN 30 HP
MILJÖVETENSKAP | LUNDS UNIVERSITET

Miljömässig hållbarhet inom mjukvaruutveckling

En enfallsstudie om mjukvaruutövares uppfattningar på ett IT-företag i Sverige

Simon Persson

2024

LUNDS UNIVERSITET

Simon Persson

MVEM31 Examensarbete för masterexamen 30 hp, Lunds universitet

Huvudhandledare: Andrius Plepys, Internationella miljöinstitutet (IIIEE), Lunds universitet

CEC - Centrum för miljö- och klimatvetenskap

Lunds universitet

Lund 2024

Abstract

The ICT sector's greenhouse gas emissions are comparable to those of the aviation industry, and the actors within the sector have a responsibility to reduce these in order to help combat global warming. A relatively new area of concern is the climate impact of software due to energy consumption. For example, AI has been acknowledged as an emissions-hungry technology because of the large computational power it requires. Over the past decade, studies have proposed tools and methods to address this issue. However, research shows that environmental sustainability within software development is largely overlooked in practice, partly due to a lack of awareness. Therefore, it is important to explore software practitioners' perceptions of the subject. By doing so, factors that may hinder or promote environmentally sustainable software practices can be identified and brought to light. These can then be considered in the creation of relevant interventions.

This study aims to examine how software practitioners at an IT company in Sweden perceive environmental sustainability in the context of software development. In order to do this, a qualitative single-case study is adopted. The views of seven participants are collected using semi-structured interviews and analyzed through thematic analysis.

The study concludes that the participants associate environmental sustainability within software development with both the climate impact of work processes and the software itself. They also recognize software's potential as a tool for promoting broader

climate efforts. Furthermore, participants demonstrate some knowledge of how the climate impact of software development can be reduced. Despite this, they also acknowledge that environmental sustainability is not actively addressed in their current workflows. Related to this, several perceived challenges and opportunities that could affect its adoption are highlighted, such as a lack of awareness and clear information. Finally, participants believe there are benefits to environmentally sustainable software practices, though their opinions on its broader importance differ. Some consider it important, while others are more skeptical.

Keywords: environmental sustainability, ICT sector, software development, software engineering, green software development, green software engineering, software practitioners, perceptions

Populärvetenskaplig sammanfattning

Växthusgasutsläpp från mänsklig aktivitet påskyndar den globala uppvärmningen och måste åtgärdas. Informations- och kommunikationstekniksektorn (IKT-sektorn), som tillhandahåller det digitala samhällets tekniker såsom datorer, smarttelefoner och internet, står för en mängd växthusgasutsläpp som kan jämföras med flygindustrins. Aktörerna inom sektorn har ett ansvar att minska dessa och ett relativt nytt problemområde är klimatpåverkan från mjukvara.

Mjukvaror såsom datorprogram är immateriella och påverkar inte klimatet på ett direkt sätt. De styr dock hur hårdvaran, det vill säga de fysiska delarna i datorn, förbrukar energi. På så sätt orsakar mjukvara växthusgasutsläpp indirekt genom energianvändning. I över tio år har forskningen utvecklat verktyg och metoder för att adressera såväl mjukvaran som mjukvaruutvecklingens klimatpåverkan, men studier visar att miljömässig hållbarhet inte beaktas i praktiken. En anledning till detta är bristande medvetenhet och kunskap om åtgärder. Det är viktigt att förstå hur aktörer som arbetar med att utveckla mjukvara – mjukvaruutövare – uppfattar miljömässigt hållbar mjukvaruutveckling. Detta för att kunna synliggöra och adressera såväl hämmande som främjande aspekter för praktisering. Sådana studier är dock begränsade i antal.

Genom att intervjua sju mjukvaruutövare på ett IT-företag i Sverige skapar denna studie inblick i uppfattningar på ett djupgående sätt. Resultaten visar att mjukvaruutövarna, även i denna studie, inte beaktar miljömässig hållbarhet inom

mjukvaruutvecklingen, men att de är medvetna om att både arbetsprocessen och användning av mjukvara kan leda till klimatpåverkan. Detta genom material- och energianvändning. De uppvisar även en viss förståelse för hur detta kan åtgärdas och diskuterar potentiella barriärer och möjligheter för att det ska ske. Ett exempel på en av de främsta barriärerna som lyfts är bristande medvetenhet om att mjukvaruutveckling kan ha en negativ påverkan på klimatet. För att överkomma detta föreslås bland annat tydlig information från företaget genom utbildning samt fysiska mötesplatser där expertkunskap kan erhållas. Andra centrala barriärer är bristande organisatoriskt stöd och ovilja att införa miljömässigt hållbara åtgärder. Studien belyser även hur mjukvaruutövarna ser på betydelsen av att beakta miljömässig hållbarhet i kontexten av mjukvaruutveckling, varpå det råder olika uppfattningar. Vissa anser att det är viktigt, medan andra är skeptiska till om det skulle ha en betydande påverkan sett till det generella klimatarbetet bortom företaget.

Sammanfattningsvis betonar studien vikten av att aktörer inom IKT-sektorn agerar för att hantera mjukvaruutvecklingens klimatpåverkan. Det finns en medvetenhet och viss förståelse för hur miljömässig hållbarhet och mjukvaruutveckling hänger ihop bland mjukvaruutövarna. Med relevanta insatser, grundande i mjukvaruutövarnas framhävda barriärer och möjligheter, kan både medvetenheten och förståelsen utökas och fördjupas för att leda till en minskning av IKT-sektorns växthusgasutsläpp.

Nyckelord: miljömässig hållbarhet, IKT-sektorn, mjukvaruutveckling, grön mjukvaruutveckling, mjukvaruutövare, uppfattningar

Innehållsförteckning

1. Inledning

I detta kapitel introduceras den problematik som studien grundar sig i. Därefter presenteras studiens syfte och frågeställning.

1.1. Introduktion

Den globala uppvärmningen är en av dagens största utmaningar; jorden blir varmare på grund av mänsklig aktivitet. Konsekvensen av detta är klimatförändringar som ökar risken för bland annat översvämningar och torka, vilket i sin tur hotar såväl människor som djurs existens (Förenta nationerna [FN], u.å.-a). En central orsak till problemet är utsläpp av växthusgaser som stänger in solens värme och höjer jordens temperatur (FN, u.å.-b). Det krävs drastiska åtgärder för att minska växthusgasutsläppen och en central del i det så kallade klimatarbetet anses vara IKT-sektorn (United Nations Framework Convention on Climate Change [UNFCCC], 2016; Europeiska kommissionen, u.å.). Majoriteten av energikällor ur ett globalt perspektiv domineras av fossila bränslen (Internationella energirådet, u.å.) och genom tekniska lösningar kan minskade växthusgasutsläpp uppnås genom exempelvis effektiv förvaltning och distribuering av energi (Ali & Choi, 2020). Trots detta har IKT-sektorn dock å andra sidan samtidigt en baksida; det är en av de snabbast växande sektorerna i termer av växthusgasutsläpp (Europeiska kommissionen, u.å.). Mer än halva jordens befolkning använder internet

och med en ökande efterfrågan från både företag och privatpersoner växer behovet av datacenter (Kumar et al., 2022). På grund av energianvändningen står dessa för ungefär 2 % av de globala växthusgasutsläppen, vilket kan jämföras med flygindustrin (UNFCCC, 2016). I sin helhet utgör IKT-sektorn upp till 4 % och de övriga växthusgasutsläppen härstammar från elektronikavfall (Europeiska kommissionen, u.å.). Avfallets växthusgasutsläpp uppstår från dess olika livscykelfaser och genom ökad efterfrågan på nya produkter och snabb innovation förkortas produkternas livslängd, vilket leder till att avfallet ackumuleras och växthusgasutsläppen ökar (Singh & Ogunseitan, 2022). Om rådande trend fortsätter i samma tempo befaras IKT-sektorns växthusgasutsläpp överstiga 14 % år 2040 (Belkhir & Elmeligi, 2018).

Aktörerna inom IKT-sektorn har ett ansvar att minska sin klimatpåverkan och ett relativt nytt problemområde är mjukvara (Chauhan & Saxena, 2013). Som abstrakt entitet har mjukvara ingen direkt påverkan på klimatet, men den styr hårdvarans operationer som i sin tur kräver elektricitet (Taina, 2010). Således orsakar mjukvara indirekta växthusgasutsläpp genom energianvändning (Sun et al., 2011; Capra et al., 2012; Erdélyi, 2013; D'Agostino et al., 2021; Ahmad Ibrahim et al., 2022). En problematisk aspekt i detta avseende är ineffektiv mjukvara, exempelvis lång och komplex kod. Detta kan orsaka mer hårdvaruintensiva operationer, vilket följaktligen kräver mer energi (Sun et al., 2011; Palomba et al., 2019). Ett mer konkret exempel är artificiell intelligens (AI). Forskare från University of Massachusetts Amherst identifierade i sin studie att processen att träna en enda AI-modell kan orsaka 626 000 pund koldioxidekvivalenter, vilket i studien jämfördes med växthusgasutsläppen från fem bilar under deras totala livslängd (MIT Technology Review, 2019). Vidare kan även så kallad ineffektiv mjukvara leda till tidigare utbyte av hårdvaran genom att mjukvaran gör den obrukbar (Naumann et al., 2011).

När denna studie genomfördes fanns inga vedertagna standarder för utveckling av "klimatsnäll" mjukvara, men i dess slutskede publicerades ISO/IEC 21031:2024. Denna ISO-standard specificerar hur mjukvara leder till klimatpåverkan samt konceptualiserar på vilket sätt detta kan förebyggas (Green Software Foundation [GSF], 2024). Specificeringen lyder:

> Mjukvara orsakar växthusgasutsläpp genom hårdvaran den körs på. Både genom energianvändning som dess fysiska hårdvara förbrukar och växthusgasutsläppen som är förknippade med tillverkningen av hårdvaran (GSF, u.å., egen översättning).

Standarden är fördelaktig gällande adressering av IKT-sektorns klimatpåverkan, men tidpunkten för publiceringen visar hur nytt ämnet är. I litteraturen har flera koncept och metoder tagits fram sedan år 2010 (Swacha, 2021). Samtidigt visar dock studier att mjukvarurelaterad klimatpåverkan inte beaktas i praktiken. En anledning till detta verkar vara att mjukvaruutövare har en snäv uppfattning om vad hållbarhet inom mjukvaruutveckling innebär, varpå fokuset tenderar att enbart beröra tekniska och ekonomiska aspekter (Groher & Weinreich, 2017). Enligt Karita et al. (2019) är det väsentligt att intresset inom akademin och praktiken är i linje för att mjukvara ska utvecklas med hänsyn till FN:s samtliga hållbarhetsdimensioner; ekonomisk, social samt miljömässig där adressering av klimatpåverkan ingår (FN, u.å.-c) Om mjukvaruutövare inte delar samma uppfattning eller ambition riskerar litteraturens framsteg att förbli i det dolda. Hur detta ter sig är dock enligt Karita et al. (2019) inte uppenbart då forskningen är begränsad. Studier som undersöker hur mjukvaruutövare uppfattar hållbarhet i kontexten av mjukvaruutveckling är relativt få, synnerligen gällande ett explicit fokus på den miljömässiga dimensionen. Genom att göra detta kan

mjukvaruutövares föreställningar om ämnet identifieras och synliggöras, exempelvis behov eller upplevda barriärer för praktisering av miljömässigt hållbara åtgärder inom mjukvaruutveckling. Dessa kan sedan tas i beaktning för att främja praktisk handling och därigenom minskad klimatpåverkan inom IKT-sektorn.

1.2. Syfte och frågeställningar

Syftet med denna studie är att undersöka hur mjukvaruutövare på ett IT-företag betraktar miljömässig hållbarhet i kontexten av mjukvaruutveckling. För att göra detta ämnar studien att besvara följande frågeställning:

- Vilka centrala uppfattningar uttrycker mjukvaruutövare på ett IT-företag om miljömässig hållbarhet i kontexten av mjukvaruutveckling?

2. Litteraturöversikt

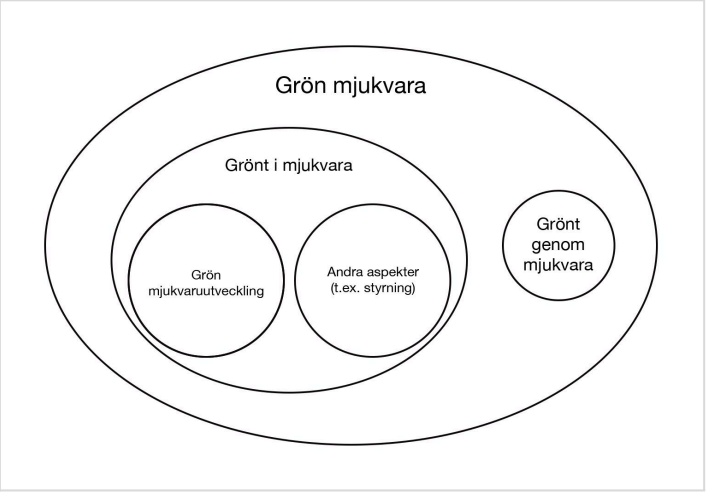
I detta kapitel presenteras en redogörelse för vad miljömässig hållbarhet innebär i kontexten av mjukvaruutveckling. Först beskrivs definitioner från litteraturen, följt av litteraturens framställning av praktiska åtgärder. Avslutningsvis sammanställs tidigare forskning om mjukvaruutövares uppfattningar i avseendet ur olika perspektiv. Litteraturöversikten syftar till att sätta studiens frågeställning i kontext och fungera som en analysram för resultat och diskussion.

2.1. Grön mjukvaruutveckling

I litteraturen har flera koncept tagits fram för att minimera klimatpåverkan från såväl mjukvaruutvecklingen som mjukvaran (se figur 1). Ett av dem är "Green software engineering" (GSWE) som kan översättas till "grön mjukvaruutveckling". Att praktisera GSWE innebär att ha inkorporerade strategier i mjukvaruutvecklingslivscykeln, det vill säga utvecklingsfasen, som ämnar att minska källor till växthusgasutsläpp från både arbetsprocessen och mjukvarans användningsfas (Calero & Piattini, 2017). Två centrala strategiområden för detta är (1) användning av "gröna" utvecklingsmodeller och (2) tekniska åtgärder för att energieffektivisera mjukvaran (Sriraman & Raghunathan, 2023). Enligt Naumann et al. (2011) är det viktigt att adressera mer än bara de tekniska aspekterna inom mjukvaruutveckling.

Detta eftersom mjukvaruutövare då har möjlighet att påverka mjukvarans slutliga form och klimatpåverkan genom sina olika ansvarsområden.

Vidare finns ytterligare koncept som syftar till att konceptualisera vad så kallad "Green software" (GSW) eller "grön mjukvara" innebär, huvudsakligen två: (1) "Green in software" (GISW) eller "grönt i mjukvara" och (2) "Green by software" (GBSW) eller "grönt genom mjukvara". GISW fokuserar på mjukvaruutvecklingen och mjukvarans klimatpåverkan genom exempelvis energieffektivisering; GSWE ingår i detta koncept. Inom GISW omfattas även andra aspekter såsom styrning (Calero & Piattini, 2017). Detta avser exempelvis organisatoriska målsättningar och policyer för klimatorienterade beslut inom mjukvaruutvecklingen (Patón-Romero et al., 2019). GBSW, å andra sidan, är mjukvara som syftar till att användas som verktyg för att minimera klimatpåverkan inom domäner bortom IT-systemet i fråga. Detta kan exempelvis ta sig uttryck genom digitalisering av fysiska företeelser (Ahmad Ibrahim et al., 2022).



Figur 1. Koncept för miljömässigt hållbar mjukvaruutveckling

Figur som visualiserar litteraturens koncept för miljömässigt hållbar mjukvaruutveckling och hur de relaterar till varandra. Anpassad från Calero och Piattini (2017).

2.2. Gröna utvecklingsmodeller

I organisatoriska kontexter utgörs mjukvaruutvecklingslivscykeln vanligen av flera olika processer. Dessa kan skilja sig organisationer emellan, men vanligt förekomna är (Haraty & Hu, 2018):

1) Kravinsamling
2) Design av specifikationer
3) Implementation
4) Tester
5) Underhåll

Utvecklingsmodeller är etablerade metoder som beskriver hur processerna ovan kan genomföras på ett systematiskt sätt i syfte att främja ett smidigt utvecklingsarbete (Haraty & Hu, 2018). En populär utvecklingsmodell är agil utveckling som förespråkar snabb återkoppling mellan involverade intressenter. Denna modell karaktäriseras av korta iterativa processer, vilket möjliggör snabb och effektiv adressering av oklarheter eller missförstånd innan utvecklingen gått för långt (Saravanan et al., 2020). Detta till skillnad från traditionella modeller, exempelvis vattenfallsmodellen, som har en linjär karaktär där varje process måste slutföras innan nästa börjar (Sinha & Das, 2021). Inom de respektive processerna arbetar olika roller med olikartade uppgifter. Exempelvis sammanställs intressenters krav och behov av affärsanalytiker, vilket andra roller såsom mjukvaruutvecklare sedan utgår från. Innan sjösättning är det vanligt att mjukvaran

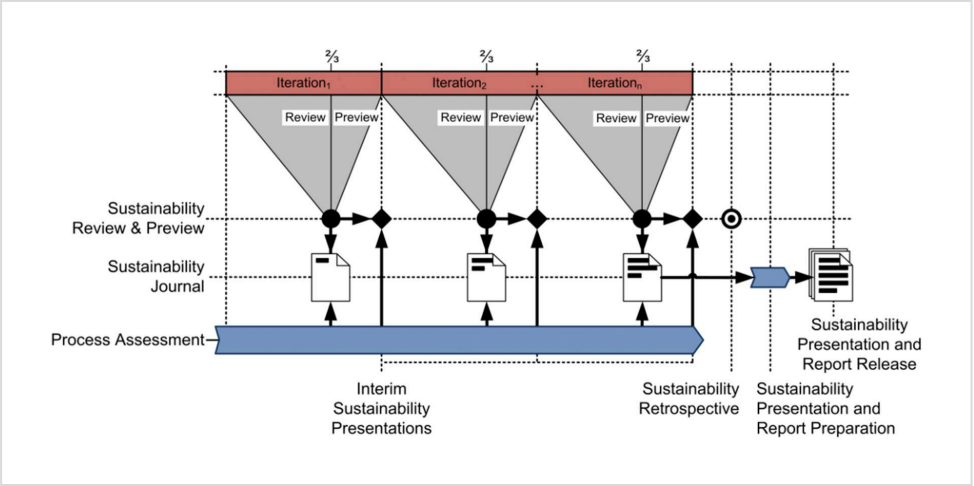
testas för att säkerställa dess kvalitet med hänsyn till de formulerade kraven. I vissa fall görs detta av så kallade testare (Hoda et al., 2012).

I en jämförande studie identifierar Anbarkhan (2023) med Fuzzy TOPSIS-metoden att agila utvecklingsmodeller främjar klimatarbete på ett mer effektivt sätt än andra. Detta då dess flexibla och iterativa karaktär bland annat möjliggör mer frekvent utvärdering av utvecklingsprocessens eventuella brister avseende klimatpåverkan som sedan kan åtgärdas. De innebär även snabbare utvecklingsprocess, vilket leder till lägre resursanvändning. Vidare uppmärksammas agila utvecklingsmodeller även av Rashid och Khan (2018) avseende utveckling av miljömässigt hållbar mjukvara. I sin litteratur- och enkätstudie med 106 industriexperter identifieras sex kritiska framgångsfaktorer:

1) Effektiv användning av tid och resurser
2) Bra kommunikation och samarbete
3) Mindre dokumentation
4) Polymorfisk mjukvarudesign
5) Miljömässigt hållbar utvecklingsprocess
6) Kontinuerlig validering av insatser

I en liknande studie av Bambazek et al. (2022) görs upptäckten att hållbarhetsarbetet kan stärkas ytterligare om det fördelas mellan olika roller. I synnerhet bör mjukvaruutvecklare ansvara för teknikrelaterade hållbarhetsaspekter medan en exklusiv roll fokuserar på det övergripande hållbarhetsarbetet med avseende på samtliga hållbarhetsdimensioner.

Flera av de aspekter som framhävs av Anbarkhan (2023), Rashid och Khan (2018) och Bambazek et al. (2022) kan finnas i en generisk modell, framtagen av Dick et al. (2013) (se figur 2). Modellen ämnar att systematiskt införliva social och miljömässig hållbarhet i mjukvaruutvecklingsarbetet på ett agilt sätt. Den utgörs av två kontinuerliga förbättringsprocesser (eng. Continuous improvement cycles) varav en fokuserar på mjukvaruutvecklingslivscykeln och den andra på mjukvarans distributions- och användningsfas. Under varje förbättringsprocess itereras fördefinierade processer som parallellt främjar en kontinuerlig inkorporering av hållbarhetsaspekter och därigenom successivt minska både mjukvaruutvecklingen och mjukvarans klimatpåverkan.



Figur 2. Utvecklingsmodell för social- och miljömässig hållbarhet inom mjukvaruutveckling.
Illustration av en utvecklingsmodell som ämnar att inkorporera social och miljömässig hållbarhet i mjukvarans utvecklings- och distributionsfas. *Bildkälla*: Dick et al. (2013).

Processerna som utgör modellen är:

A) Processbedömning:

   Under denna process initieras förbättringsprocessen som fokuserar på mjukvaruutvecklingslivscykeln. Processens syfte är att sammanställa data som vid ett framtida tillfälle kan användas i en livscykelanalys (LCA) och detta bör starta i början av utvecklingsprocessen. Dick et al. (2013) föreslår att detta görs av en hållbarhetsstrateg.

B) Hållbarhetsrecensioner och förhandsvisningar:

   Denna process initierar den andra förbättringsprocessen vars fokus är på mjukvarans distributions- och användningsfas och ska ske två tredjedelar in i en iteration. Processen inleds med en recension av tidigare hållbarhetsarbete avseende mjukvarans klimatpåverkan, med hjälp av existerande verktyg såsom mjukvara för mätning av energieffektivitet. Därefter ska identifierade brister åtgärdas i form av lösningar som sedan förhandsvisas. Detta ska i samråd med hållbarhetsstrategen utföras av utvecklingsgruppen, det vill säga de roller som ingår i det praktiska utvecklingsarbetet; designers, mjukvaruutvecklare etcetera.

C) Hållbarhetsloggbok:

   Under process A och B dokumenteras insatserna i en hållbarhetsloggbok för vidare arbete. Detta genomförs kontinuerligt av både hållbarhetsstrategen och utvecklingsgruppen under varje iteration.

D) Interimistiska hållbarhetspresentationer:

   I slutet av en iteration genomförs interimistiska presentationer för

kundrepresentanterna. Utvecklingsgruppen presenterar all identifierad klimatpåverkan från mjukvaran samt lösningarnas framgångsgrad. Hållbarhetsstrategen redogör för hela utvecklingsprocessens klimatpåverkan, inklusive växthusgasutsläpp från affärsresor med mera.

E) Slutgiltig hållbarhetspresentation och rapport:
   När mjukvaran är färdigställd framställer hållbarhetsstrategen samtliga hållbarhetsinsatser, med hänsyn till hållbarhetsloggboken och de interimistiska presentationerna, i en slutgiltig presentation och rapport för kundrepresentanterna.

F) Hållbarhetstillbakablick:
   Som avslutande process dokumenteras lärdomar och tankar av hållbarhetsstrategen tillsammans med utvecklingsgruppen för att främja nästa projekts hållbarhetsarbete.

Att påbörja hållbarhetsarbetet tidigt i utvecklingsprocessen är något som även Garćıa-Berná et al. (2021) förespråkar efter att i sin studie identifierat en korrelation mellan användarvänlighet och energiförbrukning. Upptäckten innebär att användarvänliga tjänster leder till mindre spenderad tid till att navigera i tjänsterna, vilket i sin tur innebär mindre användning av energi. Således anser Garćıa-Berná et al. (2021) att hållbarhetsaspekter bör diskuteras redan under kravinsamlingsprocessen där användarvänlighet är en central beröringspunkt.

2.3. Åtgärder för energieffektivisering

Då energieffektivisering är en central del gällande miljömässigt hållbar mjukvaruutveckling är det lämpligt att beskriva hur det kan gå till. Följande avsnitt förklarar olika angreppssätt från litteraturen.

2.3.1. Molnteknologi

Ett återkommande tema beträffande energieffektivisering i litteraturen är användning av molnteknologier. Detta framhävs även som en av flera bästa praxis i en litteraturstudie av Sriraman och Raghunathan (2023).

En vanlig teknik inom molnteknologier är virtualiserade serverdatorer, så kallade virtuella maskiner. Dessa nyttjar samma fysiska hårdvara, vilket innebär mindre hårdvaruresurser samt flexibel upp- och nedskalning utan omkonfigurering av den fysiska infrastrukturen. Detta gör att stora kostnadsbesparingar kan göras (Serrano et al., 2015), vilket har bidragit till att molnteknologier blivit allt populärare bland IT-organisationer (Kumar et al., 2022). Utöver ekonomiska fördelar innebär färre hårdvaruresurser även mindre klimatpåverkan från såväl hårdvaru- som energiförbrukning (Sriraman & Raghunathan, 2023).

Trots molnteknologiers potential att minska användning av hårdvara och energi finns det också negativa aspekter i termer av energianvändning. Servrar som står på tomgång och väntar på aktivitet förbrukar energi i onödan (Pawlish et al., 2014). Detta kan dock överkommas om molnteknologier används på ett effektivt sätt, vilket demonstreras i en studie av Kumar et al. (2022). Studien presenterar en lastbalanseringsmekanism som fördelar serveraktivitet mellan flera virtuella maskiner i

ett privat moln. Då en virtuell maskin gör anspråk på en definierad mängd hårdvaruresurser är det viktigt att ingen underutnyttjas. Mekanismen som Kumar et al. (2022) tar fram bygger på en dynamisk kostnadsmodell som utifrån belastningen tilldelar varje virtuell maskin en kostnad; ju mer, desto högre. Det är baserat på dessa kostnader som serveraktiviteten fördelas mellan de virtuella maskinerna och resulterar i att alla används. Om samtliga resurser i det privata molnet upplever hög belastning allokerar mekanismen även aktivitet till externa molntjänster för att mjukvaruapplikationen ska vara fortsatt tillgänglig för slutanvändarna (Kumar et al., 2022).

Cappiello et al. (2016) och Saboor et al. (2021) presenterar också lösningar för allokering av resurser, men med hänsyn till datacenters energikälla som ytterligare variabel. Lösningen av Cappiello et al. (2016) fördelar serveraktivitet mellan olika datacenter baserat på energimixen i landet som datacentren är belägna. Vid hög belastning prioriteras datacenter som finns i länder med "renast" energimix. Den beaktar även responstid för att inte försämra mjukvaruapplikationens prestanda. I lösningen av Saboor et al. (2021) ligger fokus på datacenters specifika energikomposition i stället för landets energimix. Lösningen fokuserar även på datans färdsträcka i stället för serveraktivitet. Om en mjukvaruapplikations molnfunktionalitet residerar i datacenter som är långt ifrån varandra behöver datan färdas genom fler nätverksinfrastrukturer, vilket innebär mer energianvändning. Därför beräknar lösningen av Saboor et al. (2021) hur beroende molntjänsterna är av varandra i termer av interaktion; ju mer interaktion, desto högre är beroendet. Sedan allokeras de följaktligen till datacenter i form av kluster. Denna metod innebär mindre användning av icke-förnybar energi och kortare färdsträckor för datan.

Andra studier hanterar serveraktivitet på annorlunda sätt än allokering av aktiviteten mellan datacenter. Cioca och Schuszter (2022) presenterar ett system som dynamiskt stänger av och på containrar genom djup maskininlärning. Containrar är virtuellt instansierade komponenter som likt virtuella maskiner residerar i datorer. Skillnaden är att de använder datorns operativsystemkärna och kan instansieras i såväl fysiska som virtuella datorer för att exempelvis agera servrar (Bigelow, 2024). Precis som virtuella maskiner konsumerar containrar energi även i väntande eller inaktivt läge. Systemet av Cioca och Schuszter (2022) lär sig att förutspå när såväl låg som plötslig ökning av serveraktivitet är väntat. Utifrån det avslutar och startar systemet containerinstanser dynamiskt för att spara resurser.

En annan lösning är SaaScaler, framtagen av Hasan et al. (2017). Likt tidigare lösningar beaktar detta verktyg tillgängligheten av datacenter drivna med förnybar energi, serverbelastning och responstid, men minskar mjukvaruapplikationens energiförbrukning genom att dynamiskt ändra dess visuella innehåll. Det är vanligt att mjukvaruapplikationer innehåller element som egentligen inte behövs för deras huvudsakliga syfte, så kallade trevligt-att-ha-funktionaliteter (eng. Nice-to-have features). Ett exempel är produktrekommendationer i en webbutik. Utifrån de nämnda kriterierna växlar SaaScaler mellan olika fördefinierade versioner av mjukvaruapplikationen. Exempelvis visas färre trevligt-att-ha-funktionaliteter när "rena" datacenter är otillgängliga eller om serveraktiviteten är hög.

2.3.2. Energimätning

En annan metod för energieffektivisering av mjukvara som behandlas i litteraturen är mätning av mjukvarans energiförbrukning (Procaccianti et al., 2015; D'Agostino et al. (2021; Sriraman & Raghunathan, 2023). Genom energimätning kan källor till ökad

energianvändning identifieras och optimeras (Sriraman & Raghunathan, 2023). Noureddine et al. (2014) beskriver mjukvarans energiförbrukning som den mängd energi, mätt i joule (J), som hårdvaran konsumerar på begäran av mjukvaran. I sin studie tar Noureddine et al. (2014) fram ett mjukvaruverktyg som beräknar energiförbrukningen av hårdvaran, kopplat till funktionaliteten i mjukvaran. Energidatan hämtas direkt i programmeringsprogrammet genom två mjukvarubaserade mätsystem. Detta möjliggör kartläggande av kod som leder till höga energikällor (eng. Energy hotspots), vilka därigenom kan åtgärdas. Mancebo et al. (2021) föreslår också en lösning som mäter hårdvarans energiförbrukning, men med fokus på mjukvarans generella energiförbrukning vid användning av den. Verktyget använder en fysisk energimätare för att hämta energidatan, som är monterad på IT-enheten. Datan tillgängliggörs därefter i ett visuellt gränssnitt för användaren att beakta. Ett liknande verktyg presenteras av Beghoura et al. (2017). Det samlar också in data om IT-enhetens energiförbrukning vid användning av mjukvaror, men i stället för att endast visualisera datan används den för att genom maskininlärning träna verktyget till att estimera hur mycket energi som diverse mjukvarubeteenden orsakar.

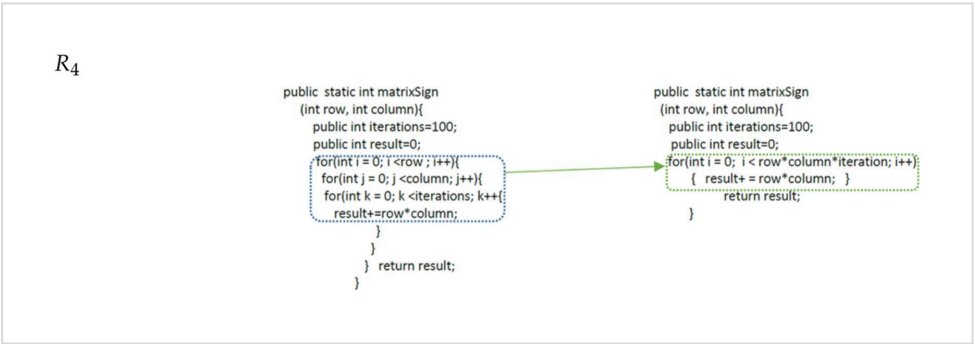
Ett annat verktyg, framställt av Munoz et al. (2018), tar in serverkonfigurationer som parametrar för att genom statistiska beräkningar analysera och visualisera hur mycket energiförbrukning konfigurationerna ger upphov till. Verktyget är i formatet av en webbapplikation och analyserna baseras på data från en databas som både forskare och mjukvaruutvecklare kan bidra till för att förbättra analysresultaten.

2.3.3. Refaktorering

Studier lyfter att mjukvara som innehåller många koddesignsfel (eng. Code smells) kan leda till mer energiförbrukning än nödvändigt och att detta kan åtgärdas genom att

refaktorera koden (Palomba et al., 2019; Sehgal et al., 2022; Şanlıalp et al., 2022). Refaktorering är en vanlig företeelse inom mjukvaruutveckling för att göra mjukvaran enklare att underhålla. Det innebär att existerande kod förbättras, exempelvis i termer av struktur, utan att mjukvarans slutgiltiga och avsiktliga funktionalitet förändras (Sehgal et al., 2022).

I en studie av Şanlıalp et al. (2022) jämförs flera refaktoreringstekniker med programmeringsspråken C# och Java, både enskilt och i kombination med varandra. Den refaktoreringsteknik som visar störst reducering av energianvändning, benämnd R4 i studien, är refaktorering av kapslade loopar (se figur 3). En loops uppgift är att upprepa funktionalitet, vanligtvis till dess att ett definierat villkor uppnåtts. Ett exempel är att multiplicera värden ett specifikt antal gånger. Vidare innebär en så kallad kapslad loop flera loopar i varandra.



Figur 3. Refaktoreringsteknik för kapslade loopar.

Kodexempel på en refaktoreringsteknik för kapslade loopar. *Bildkälla*: Şanlıalp et al. (2022).

Şanlıalp et al. (2022) presenterar ett R4-exempel som förkortar en kapslad loop till att endast innefatta en i stället för tre loopar. I exemplet är loopens övergripande uppgift att iterativt lägga till produkten av två numeriska parametrar, benämnda “row” respektive “column”, till ett ackumulerat värde. Detta 100 gånger multiplicerat med

parametrarnas produkt. Med mätverktyg visar studien att den refaktorerade versionen minskar hårdvarans energiförbrukning, i joule, med ett genomsnitt på 6,5 %. När olika refaktoreringstekniker kombineras ingår R4 i samtliga kombinationer med lägst energiförbrukning. Vidare formulerar Şanlıalp et al. (2022) även ett index som beskriver hur komplex koden är att underhålla efter att de olika kombinationerna tillämpats. Även i detta fall resulterar R4:s kombinationer i bäst utfall, vilket visar att refaktorering både kan minska mjukvarans energiförbrukning och samtidigt förbättra dess underhållbarhet.

Ineffektiva loopar identifieras också som en energibov i en liknande studie av Palomba et al. (2019). Detta åtgärdas genom att ändra loopen till en så kallad för-varje-loop (eng. For-each loop). För-varje-loopar upprepas inte baserat på ett specificerat villkor såsom looparna i figur 3. De itererar i stället antalet gånger som ett specifikt element tillåter, exempelvis en listas längd. I studien resulterar denna refaktorering i en 87 gånger lägre energiförbrukning; från 0,87 J till 0,01 J.

I en annan studie, av Sehgal et al. (2022), kartläggs flera koddesignsfel i 18 olika mjukvaruapplikationer, som sedan refaktoreras. Koddesignsfelen är bland annat onödigt lång kod för det mjukvaran ska utföra, vilket åtgärdas genom att korta ner koden eller bryta ut den i mindre block. Efter refaktoreringarna jämförs energiförbrukningen för respektive applikation och dess ursprungliga version i watt (W) och resultatet visar att den reducerats för samtliga. Applikationen med störst minskning reducerades från 23,94 W till 22,89 W.

2.3.4. Övriga metoder

Andra tillvägagångssätt för att minska energiförbrukningen av mjukvara är en blandning av tekniska val, både i termer av applikationsdesign och teknikval. Agosta et

al. (2012) föreslår memoisering (eng. Memoization), en vanlig metod för att optimera mjukvarans prestanda i termer av snabb responstid. Metoden innebär att resultatet från funktioner lagras för att undvika exekveringen av samma funktion vid förfrågan om samma resultat. På så sätt besparas hårdvaruresurser extra arbete och leder till lägre energiförbrukning. Lösningen av Agosta et al. (2012) bygger på ett automatiserat system som dynamiskt memoiserar med hänsyn till datorminne och kriterier såsom funktionernas frekvens och deras behov av hårdvarukapacitet. Genom en komponent bedöms potentialen för hur mycket energibesparing som memoisering av funktionen i fråga kan leda till. Baserat på detta tilldelas funktionen en mängd minne att nyttja för lagring. På så sätt bestämmer komponenten inte bara om en funktion ska memoiseras, utan även hur mycket minne den ska få tillgång till baserat på dess förväntade nytta i att minska energiförbrukningen. Denna dynamiska process möjliggör för systemet att anpassa sig till ändrade användningsmönster och optimerar kontinuerligt energieffektiviteten över tid.

Procaccianti et al. (2016) demonstrerar potentialen för minskad energianvändning genom två metoder: (1) modifiering av SQL-instruktioner och (2) användning av en sovfunktion för Apache-servrar. SQL är ett språk som används för att manipulera och hämta data i tabellariska databaser. En Apache-server är en så kallad webbserver, vilket är en server som hanterar kommunikation mellan IT-enheter. Studien visar att energiförbrukningen kan minskas med 25 % genom att undvika sortering av data i SQL-instruktioner och att sätta Apache-servrar i sovläge. Enligt Procaccianti et al. (2016) ger detta resultat en liten inblick i hur energieffektiva mjukvaruapplikationer skulle kunna åstadkommas.

Pereira et al. (2021) undersöker hur mycket energiförbrukning som vanligt använda programmeringsspråk ger upphov till vid exekvering av kod med liknande

syfte. De tio programmeringsspråken med den lägsta generaliserade energiförbrukningen är:

1. C (1,00 J)
2. Rust (1,03 J)
3. C++ (1,34 J)
4. Ada (1,70 J)
5. Java (1,98 J)
6. Pascal (2,14 J)
7. Chapel (2,18 J)
8. Lisp (2,27 J)
9. Ocaml (2,40 J)
10. Fortran (2,52 J)

De med högst generaliserad energiförbrukning är:

1. Perl (79,58 J)
2. Python (75,88 J)
3. Ruby (69,91 J)
4. JRuby (46,54 J)
5. Lua (45,98 J)

6. Erlang (42,23 J)
7. PHP (29,30 J)
8. Hack (24,02 J)
9. TypeScript (21,50 J)
10. Racket (7,91 J)

Enligt Pereira et al. (2021) kan detta förklaras med att de lågförbrukande programmeringsspråken främst är så kallade kompilerande språk (eng. Compiled languages). Detta innebär att koden konverteras till maskinkod (ettor och nollor) i form av en exekverbar fil, vilket möjliggör att hårdvaran direkt kan exekvera mjukvaran utan extra resurskrävande steg. Programmeringsspråken med högre energiförbrukning är primärt tolkande språk (eng. Interpreted languages), vilka tolkar koden rad för rad medan den körs. Detta ger upphov till ökad användning av datorresurser och således mer energianvändning. Studiens resultat ger mjukvaruutövare insikt om energisnåla programmeringsspråk, vilket kan vara fördelaktigt vid utveckling av "grönare" mjukvara (Pereira et al., 2021).

2.4. Mjukvaruutövares uppfattningar

Som avslutande avsnitt är det passande att redogöra för vad tidigare studier identifierat gällande mjukvaruutövares uppfattningar om miljömässig hållbarhet i kontexten av mjukvaruutveckling. Följande avsnitt presenterar varierande perspektiv och utgångspunkter från fem olika studier.

2.4.1. Medvetenhet om miljömässigt hållbar mjukvaruutveckling

Enkät- och intervjustudier av både kvalitativ och kvantitativ karaktär visar att mjukvaruutövare har en snäv uppfattning om vad hållbar mjukvaruutveckling innebär. Detta i bemärkelsen att majoriteten av studiernas studiedeltagare främst fokuserar på tekniska och ekonomiska aspekter såsom enkelt underhåll av kod (Karita et al., 2019; Noman et al., 2022) och att hålla kostnader nere (Groher & Weinreich, 2017). Hur miljömässig hållbarhet relaterar till mjukvaruutveckling är inte självklart för studiedeltagarna. Många uppger att de har låg förståelse för ämnet då det är första gången de hör talas om det (Karita et al., 2019), medan vissa anser att företeelserna inte har med varandra att göra (Chitchyan et al., 2016). Andra nämner inte miljömässig hållbarhet alls vid diskussion om hållbar mjukvaruutveckling (Groher & Weinreich, 2017). Ett fåtal har en övergripande medvetenhet om att mjukvaruutvecklingen kan leda till klimatpåverkan genom pappersanvändning, energikrävande hårdvara och mjukvarans energiförbrukning. Djupare förklaringar uppges dock inte och enligt studierna visar detta på bristande medvetenhet (Chitchyan et al., 2016; Karita et al., 2019; Noman et al., 2022). I en studie av Ahmad (2022) å andra sidan visas ett annat resultat. Majoriteten av studiedeltagarna i den studien uppger att de är familjära med miljömässigt hållbar mjukvaruutveckling. Detta genom att aktivt arbeta med åtgärder i avseendet eller för att de tagit del av information om ämnet från organisationen och andra ställen. Detta speglar hur Chitchyan et al. (2016) förklarar den varierande medvetenheten i sin studie, det vill säga att den verkar bero på exponering och intresse för hållbarhetsfrågor. Groher och Weinreich (2017) tror att det kan ha att göra med vilken bransch det handlar om. I övrigt anser Karita et al. (2019) och Noman et al. (2022) att den låga medvetenheten är resultatet av att miljömässig hållbarhet inte är ett etablerat ämne inom mjukvaruindustrin i stort.

2.4.2. Kunskap om miljömässigt hållbara åtgärder

Flertalet studier betraktar kunskapen om miljömässigt hållbara åtgärder som bristfällig, både generellt och i termer av djup. Detta baserat på att de flesta eller samtliga studiedeltagare inte kan peka på konkreta åtgärder och att de svar som lyfts inte är djupgående (Chitchyan et al., 2016; Groher & Weinreich, 2017; Karita et al., 2019; Noman et al., 2022). Kunskapsbristen anses bero på:

1) Begränsad erfarenhet av praktisk tillämpning (Chitchyan et al., 2016; Groher & Weinreich, 2017; Noman et al., 2022).

2) Avsaknad av utbildning (Chitchyan et al., 2016).

3) Bristfällig kunskapsöverföring från akademin (Noman et al., 2022).

4) Miljömässig hållbarhet beaktas inte inom mjukvaruindustrin (Karita et al., 2019; Noman et al., 2022).

Likt medvetenheten om vad miljömässig hållbarhet innebär i kontexten av mjukvaruutveckling, identifierar Chitchyan et al. (2016) att kunskapen om åtgärder verkar vara högre hos studiedeltagare som har exponerats för hållbarhetsfrågor i större utsträckning.

Till skillnad från studierna ovan visar Ahmad (2022) ett annat resultat, det vill säga att kunskapen generellt är hög bland studiedeltagarna. I synnerhet inom organisationer där miljömässigt hållbara åtgärder tillämpas. De fem mest förekommande åtgärderna är (1) hårdvaruåtervinning, (2) användning av förnybar energi genom solpaneler, (3) användning av energisnåla datorskärmar, (4) energieffektiv kodning samt (5) digital kommunikation.

2.4.3. Betydelsen av att beakta miljömässig hållbarhet

Studiernas studiedeltagare uppger varierande perspektiv kring betydelsen av att beakta miljömässig hållbarhet inom mjukvaruutveckling.

2.4.3.1. För- och nackdelar

I studien av Ahmad (2022) anser flertalet studiedeltagare att miljömässigt hållbara åtgärder har flera fördelar. Däribland minskad klimatpåverkan, ökad kvalitet för mjukvaran och kostnadsbesparingar till följd av att resurser kan användas och fördelas på ett effektivare sätt. Karita et al. (2019) gör liknande fynd avseende de två förstnämnda fördelarna, men det är endast en studiedeltagare som framhäver detta. Resten kan varken ange för- eller nackdelar, vilket även är fallet i studien av Noman et al. (2022). Detta anses bero på den begränsade förståelsen för vad miljömässig hållbarhet innebär i kontexten av mjukvaruutveckling (Karita et al., 2019; Noman et al., 2022). En ytterligare fördel som lyfts är att praktisering av miljömässigt hållbara åtgärder kan attrahera kunder (Karita et al., 2019).

Beträffande nackdelar framhävs ökade kostnader av studiedeltagare i studien av Chitchyan et al. (2016). Uppfattningen är att åtgärderna skulle innebära mer arbete och att det är tidsmässigt och därmed ekonomiskt kostsamt. Studiedeltagarna menar även att tiden inte finns i övrigt.

2.4.3.2. Betydelsefullhet

Trots en generellt låg förståelse för vad miljömässig hållbarhet innebär i kontexten av mjukvaruutveckling, anser majoriteten av studiedeltagarna i studierna av Karita et al. (2019) och Noman et al. (2022) att ämnet är viktigt. Ett fåtal tycker dock att det är oviktigt. I studien av Chitchyan et al. (2016) görs liknande fynd, med den ytterligare

insikten att detta verkar bero på intresse och exponering för hållbarhetsfrågor. De få som anser att hållbarhet är mindre viktigt i avseendet hävdar att det inte finns en koppling mellan hållbarhet och mjukvaruutveckling.

2.4.4. Barriärer och möjligheter för praktisering av miljömässigt hållbar mjukvaruutveckling

Samtliga studier förutom den av Ahmad (2022) visar att miljömässigt hållbara åtgärder inte praktiseras alls eller i en större utsträckning. I detta avseende diskuterar studierna varierande barriärer och möjligheter.

2.4.4.1. Medvetenhet och kunskap

En central barriär anses vara bristande medvetenhet och kunskap, både gällande hur miljömässig hållbarhet relaterar till mjukvaruutveckling och vilka åtgärder som kan vidtas. En återkommande möjlighet som föreslås är vägledning i form av riktlinjer och verktyg (Chitchyan et al., 2016; Groher & Weinreich, 2017; Karita et al., 2019; Noman et al., 2022). Ett annat förslag är utbildning på flera nivåer, inklusive universitetsutbildningar och interna utbildningar på arbetsplatsen (Chitchyan et al., 2016). Slutligen är ett tredje förslag effektiv kunskapsöverföring från akademin. Detta med betoning på tydligt kommunicerad information eftersom vetenskapliga fakta kan vara alltför abstrakta för att förstå (Noman et al., 2022).

2.4.4.2. Organisatoriskt stöd

Chitchyan et al. (2016) framhäver att studiedeltagarna uppfattar att bristande organisatoriskt stöd är en betydande barriär för praktisering av miljömässigt hållbara åtgärder. De anser att det behövs vägledande riktlinjer och tydligt kommunicerade

hållbarhetsvärderingar, vilket även Noman et al. (2022) menar är viktigt för att andra aspekter än tekniska ska få utrymme. Enligt Noman et al. (2022) leder ett starkt fokus på tekniskt orienterade företeelser ofta till att annat förbises, däribland miljömässig hållbarhet.

2.4.4.3. Organisatorisk vilja

En annan barriär som upplevs av studiedeltagare är organisatorisk ovilja (Chitchyan et al., 2016; Karita et al., 2019). Flera menar att miljömässigt hållbara åtgärder skulle innebära högre kostnader och förlorad tid som annars hade kunnat användas på annat, vilket de tror gör det svårt att övertyga organisatoriska beslutsfattare (Chitchyan et al., 2016). Andra lyfter att vissa organisationer ser miljömässig hållbarhet som irrelevant och att det är ett hinder (Karita et al., 2019). För att främja organisatorisk vilja föreslår Chitchyan et al. (2016) att mjukvaruutövare tydligt demonstrerar fördelarna med att praktisera åtgärder som beaktar miljömässig hållbarhet.

2.4.4.4. Gemensamt ansvar

Chitchyan et al. (2016) identifierar att vissa studiedeltagare tenderar att lägga ansvaret på kunden för att hållbarhet ska adresseras inom mjukvaruutveckling. Några lägger det på organisationen. Enligt Chitchyan et al. (2016) är det viktigt att det finns en uppfattning om ett delat ansvar. För att uppnå detta föreslår Chitchyan et al. (2016) kunskapsspridning. Detta med avseende på både yrkesverksamma, beställare och användare av mjukvara. Chitchyan et al. (2016) menar att detta kan bidra till nya normer som successivt främjar en mer hållbar mjukvaruutveckling.

3. Metod och material

I detta kapitel beskrivs och motiveras studiens forskningsprocess, design och de metoder som använts för att kunna besvara frågeställningen. Kapitlet innehåller även reflektioner kring studiens tillvägagångssätt med avseende på etik samt reliabilitet och validitet.

3.1. Fallstudie

Denna studie genomfördes som en kvalitativ enfallsstudie av ett svenskt IT-företag. Fallstudier ämnar att skapa djupgående förståelse för företeelser i sociala miljöer. Detta bedömdes som en lämplig forskningsdesign för att på ett uttömmande sätt undersöka hur mjukvaruutövare uppfattade miljömässig hållbarhet i kontexten av mjukvaruutveckling. Valet att utforska ett enskilt fall baserades främst på tidsbrist, men även fallets typiska karaktär i hur mjukvaruutveckling vanligen tar sig uttryck i praktiken (Yin, 2013).

3.1.1. IT-företaget

Studiens fall var ett svenskt IT-företag som ingår i en global koncern med över 15 000 anställda. Koncernens olika företag erbjuder digitala tjänster riktade mot såväl privat

som offentlig sektor. Det undersökta företaget tillhandahåller mjukvaruapplikationer, framför allt för löne- och hjälpmedelshantering, till svenska kommuner och regioner. Mjukvaruutvecklingen på företaget följer konventionella processer, däribland agila utvecklingsmodeller.

3.2. Litteraturöversikt

Studien inkluderade en litteraturöversikt i syfte att skapa förståelse för vad miljömässig hållbarhet innebar i kontexten av mjukvaruutveckling samt lokalisera tidigare forskning om mjukvaruutövares uppfattningar om företeelsen. Den kom att sätta studiens frågeställning i kontext och utgöra en teoretisk ram för såväl resultat som diskussion. Litteraturen erhölls genom en systematisk litteratursökning, vilket lämpar sig när litteratur om ett särskilt ämne eftersöks (Rienecker & Jørgensen, 2014). Detta genomfördes i Web of Science, inställt på alla databaser, med två inklusionskriterier. Det första kriteriet var att endast vetenskapliga artiklar var av intresse för att säkerställa pålitlighet. Därmed ställdes dokumenttyp in som “Article”. Det andra var att artiklarna behövde vara skrivna på engelska då inga andra språk behärskades. Således valdes “English” under språkinställningar. Sökströngen som användes resulterade i 403 resultat den 15 februari 2024 och löd:

> (TS=(“green software development” OR “green software engineering” OR "sustainable software development” OR “sustainable software engineering” OR “software sustainability” OR ”green software”)) OR (TS=(“software development” OR “software engineering” OR “software”) AND TS=(“green

it" OR "green ict")) OR (TS=("software development" OR "software engineering") AND TS=("climate impact" OR footprint OR carbon)).

För att exkludera artiklar som ansågs vara irrelevanta för studiens ändamål användes exklusionskriterier i en granskningsprocess bestående av tre iterationer. Under den första iterationen exkluderades artiklarna som inte var referentgranskade. För de två sista iterationerna var exklusionskriteriet att artikelns innehåll inte berörde miljömässig hållbarhet med fokus på mjukvaruutveckling, först utifrån abstract och sedan fullt innehåll.

Avslutningsvis strukturerades litteraturöversikten genom kodning, vilket är ett sätt att organisera litteraturen och underlätta syntetisering av dess innehåll (Saunders et al., 2023). Koderna formulerades som kortare meningar och resulterade i fyra övergripande teman:

1. Koncept för miljömässigt hållbar mjukvaruutveckling
2. Gröna utvecklingsmodeller
3. Åtgärder för energieffektivisering av mjukvara
4. Mjukvaruutövares uppfattningar om miljömässig hållbarhet i kontexten av mjukvaruutveckling

Under arbetets gång kompletterades litteraturöversikten med ytterligare material för att fylla luckor. Detta gjordes genom snöbollsurval samt slumpmässigt urval i Google Scholar. Det slutgiltiga antalet vetenskapliga artiklar blev 38.

3.3. Semistrukturerade intervjuer

Den primära datan som användes för att besvara studiens frågeställning var kvalitativ och samlades in empiriskt genom semistrukturerade intervjuer. Denna intervjumetod ansågs passande då den främjar djupgående insikt i intervjupersonernas föreställningar om ett visst ämne, vilket lämpar sig för fallstudier där just detta är målet (Bryman, 2012). Inför intervjuerna konstruerades en intervjuguide med förbestämda frågor (bilaga 1), vilka formulerades med inspiration från tidigare studier. Genom möjligheten att avvika från intervjuguidens innehåll och frågornas ordningsföljd, såsom intervjumetoden tillåter (Bryman, 2012), kunde samtalen röra sig fritt och följsamt med intresset hos intervjupersonerna. Därigenom skapades en inblick i uppfattningarna om miljömässig hållbarhet i kontexten av mjukvaruutveckling på ett uttömmande sätt. Intervjuerna genomfördes med videomöte och varade mellan 40 och 60 minuter. De spelades även in med intervjupersonernas samtycke, varpå inspelningarna först transkriberades digitalt och sedan manuellt inför analysarbetet.

3.3.1. Urval av intervjupersoner

Totalt intervjuades sju personer. Detta bedömdes som tillräckligt avseende datamättnad, det vill säga att fler intervjuer inte skulle leda till nya insikter eller teman i intervjupersonernas svar. Intervjupersonerna valdes genom bekvämlighetsurval via en kontaktperson på IT-företaget, vilket innebär att urvalet baseras på tillgänglighet (Bryman, 2012). Det är dock viktigt att dessa är representativa avseende vad som undersöks (Saunders et al., 2023) och därmed efterfrågades endast kandidater som

arbetade med mjukvaruutveckling på något sätt; mjukvaruutvecklare, chefer, grafiska formgivare etcetera. Se tabell 1 för samtliga intervjupersoner och deras roller.

Tabell 1.

Studiens intervjupersoner, deras arbetstitlar och när intervjuerna ägde rum.

INFORMANT	ARBETSTITEL	DATUM
A	Business analyst	20/3 2024
B	UI-designer (tidigare mjukvaruutvecklare)	20/3 2024
C	IT-chef	21/3 2024
D	Utvecklingskonsult (mjukvaruutvecklare)	22/3 2024
E	Team manager consultant (tidigare mjukvaruutvecklare)	25/3 2024
F	Utvecklingskonsult (mjukvaruutvecklare)	26/3 2024
G	Product and development manager	2/4 2024

3.3.2. Etisk reflektion

När data samlas in genom intervjuer blir överväganden kring respekt för intervjupersonerna aktuella. Detta avseende frivillighet, integritet, konfidentialitet och anonymitet. Således tillämpades fyra etiska principer, beskrivna av Bryman (2018): (1) informationskravet, (2) samtyckeskravet, (3) konfidentialitetskravet och (4) nyttjandekravet. Informationskravet innebär att studiens syfte och avsedda process kommuniceras till samtliga medverkande, vilket gjordes genom ett informationsbrev

innan intervjuerna ägde rum. Informationsbrevet inkluderade även information avseende de tre resterande principerna. Samtyckeskravet avser de medverkandes rätt att bestämma över sin medverkan, varpå rätten att hoppa av när som helst under studiens gång kommunicerades. Konfidentialitetskravet innebär att personuppgifter förvaras på ett säkert sätt och nyttjandekravet att material som samlas in inte används bortom studiens ändamål. Detta respekterades genom att inte dela denna typ av information med utomstående parter.

3.4. Tematisk analys

Intervjumaterialet analyserades genom tematisk analys, vilket är en vedertagen och vanlig metod för att analysera kvalitativa data. Metoden ämnar att identifiera teman i data som kan förklara de företeelser som är av intresse; i detta fall intervjupersonernas uppfattningar om miljömässig hållbarhet i kontexten av mjukvaruutveckling (Bryman, 2012). Det finns inga strikta regler för hur tematiska analyser bör genomföras, vilket innebär stor frihet i detta avseende (Bryman, 2012; Saunders et al., 2023). Vanligen utvinns temana genom kodning i en iterativ process där datans innehåll kodas. För detta finns inte heller fasta riktlinjer, utan det kan göras på ett sätt som passar situationen i fråga. Kodningens syfte är att systematiskt extrahera betydelsen i datan för att genom kombination av koder bilda slutgiltiga teman. I denna studie utfördes kodningen genom formulering av kortare meningar och in-vivo-kodning, varav det sistnämnda är en kodningsteknik som innebär att intervjupersonernas egna ord bildar koderna. Detta gjordes på ett abduktivt sätt där teman från litteraturöversikten användes initialt, för att sedan successivt revideras och leda till nya. Det är viktigt att

revidera flera gånger för att hitta essensen i datan och därigenom uppnå ett välgrundat resultat (Saunders et al., 2023).

3.5. Reliabilitet och validitet

Studier med kvalitativ ansats brukar kritiseras då resultatens kvalitet är svår att bedöma på grund av dess icke mätbara karaktär. Det går dock att ifrågasätta om resultaten kan betraktas som tillförlitliga och giltiga (Lantz, 2013).

3.5.1. Reliabilitet

Inom kvalitativ forskning är extern och intern reliabilitet koncept för att bedöma studien och dess resultats tillförlitlighet. Extern reliabilitet avser replikerbarhet, det vill säga huruvida en studie kan replikeras och producera samma resultat. Detta är dock inte enkelt då det inte går att "kopiera" en social miljö och dess kontextuella betingelser (LeCompte & Goetz, 1982, refererat i Bryman, 2018). Något som kan göras är att redogöra för hur studien genomförts, vilket gjordes i detta metodkapitel (Miles et al., 2013). Intern reliabilitet betyder att studiens involverade forskare är överens om tolkningar, men då denna studie genomfördes självständigt är detta inte tillämpbart (LeCompte & Goetz, 1982, refererat i Bryman, 2018).

3.5.2. Validitet

Kvalitativa studiers giltighet kan bedömas genom extern och intern validitet. Extern validitet handlar om i vilken utsträckning studiens resultat kan generaliseras till andra

sociala miljöer (LeCompte & Goetz, 1982, refererat i Bryman, 2018). Fallstudier och i synnerhet enfallsstudier begränsar den statistiska generaliserbarheten på grund av det kontextbundna fokuset. Däremot kan resultaten bidra till analytisk generalisering, vilket innebär att existerande fynd byggs vidare på. I övrigt kan de skapa värdefulla insikter om kontexten i fråga (Yin, 2013). Intern validitet innebär att det finns en överensstämmelse mellan forskarens observationer och verkligheten (LeCompte & Goetz, 1982, refererat i Bryman, 2018). Detta kan säkerställas genom triangulering, vilket innebär att flera forskningsresurser används för att verifiera studiens fynd, exempelvis datakällor eller metoder. I denna studie användes den empiriska datan tillsammans med befintlig teori från tidigare studier, vilka utgjordes av såväl kvalitativa som kvantitativa resultat (Miles et al., 2013).

4. Resultat

I detta kapitel presenteras resultaten från det tematiskt analyserade intervjumaterialet. Kapitlet är uppdelat i huvud- och underteman som analysen resulterat i, vilka skildrar hur intervjupersonerna uppfattar miljömässig hållbarhet i kontexten av mjukvaruutveckling.

4.1. Medvetenhet och kunskap

Intervjupersonerna uttrycker varierande uppfattningar kring förbindelsen mellan miljömässig hållbarhet och mjukvaruutveckling.

4.1.1. Svårt att se en koppling

Fem intervjupersoner påpekar vid ett eller flera tillfällen att det är svårt att se hur miljömässig hållbarhet och mjukvaruutveckling hänger ihop. En anledning som lyfts är att ämnet inte reflekterats kring tidigare. Enligt informanterna C, D, E och G gör detta att inga associationer uppstår intuitivt. En annan uttryckt orsak är mjukvarans abstrakta karaktär. Informant A uppger att hen förknippar miljömässig hållbarhet med klimatpåverkan och att det är svårare att se hur mjukvara skulle kunna påverka klimatet jämfört med andra produkter. Informanten lyfter ett exempel om bilar, varpå hen anser att avgaser är tydligare att greppa i avseendet. Ett liknande resonemang lyfts av

informant G. De resterande intervjupersonerna, det vill säga informanterna B och F, uttrycker inga större svårigheter kring att pussla ihop miljömässig hållbarhet och mjukvaruutveckling. Båda uppger till och med att de tagit del av information om ämnet tidigare. Informant F förklarar även varför det kan vara svårt att se en koppling ur ett generellt perspektiv på ett sätt som resonerar med tidigare uttalanden:

> Oftast tänker man ju inte på att man kan göra någonting åt det med kod, utan det är hårdvaran och allting runt omkring som påverkar (Informant F).

4.1.2. Mjukvaruutvecklingens klimatpåverkan

Samtliga intervjupersoner uttrycker tankar om hur mjukvaruutveckling kan leda till klimatpåverkan. Det bör dock noteras att de som uppfattar kopplingen som svår att se kontinuerligt uppger att tankarna är sprungna ur antaganden.

4.1.2.1. Resor

Sex intervjupersoner framhäver att bil- och flygresor påverkar klimatet genom växthusgasutsläpp. Samtidigt finns det dock funderingar om huruvida detta faller under mjukvaruutveckling. Informanterna C och D uttrycker en tveksamhet medan informant G uttryckligen menar att det inte har med mjukvaruutveckling att göra:

> Jag tänker resor och sånt, kan man inkludera det? (Informant C).

> Det är väl resandet [...] men det känns så fjuttigt att bara beskriva den biten tycker jag för det måste ju finnas så mycket mer (Informant D).

> För det är ju som två delar. Jag menar hur vi reser är ju en sak, men hur vi då gör det i utvecklingen är ju en annan sak (Informant G).

4.1.2.2. Mjukvarans resursanvändning

Samtliga intervjupersoner diskuterar varierande aspekter kring mjukvarans klimatpåverkan. Detta med fokus på resursanvändning. Centrala teman i avseendet är ineffektiv mjukvara och trender.

Ineffektiv mjukvara

Fem intervjupersoner uppger att mjukvara kan konstrueras på ett sätt som leder till ökad användning av hårdvaruresurser och energi. Informanterna A och F diskuterar detta ur ett serverperspektiv, varpå de lyfter att det finns en tendens bland mjukvaruutövare att utöka serverarkitekturens hårdvara vid optimering av prestanda. Enligt informant A är det i detta avseende viktigt att lastbalansera serveraktivitet för att undvika överdriven användning av nämnda resurser. Liknande resonemang görs av informant F som lyfter virtualisering som en potentiell lösning. Hen uttrycker även vidare att vissa datacenter drivs av fossila bränslen, vilket enligt informanten gör det desto viktigare att optimera mjukvara i termer av resursanvändning för att minska växthusgasutsläpp. Datacenters användning av "dåliga" energikällor är även något som informant E lyfter som problematiskt, men ur ett generellt perspektiv gällande molnteknologi i stort.

Vidare belyser informanterna C och B att lång och slarvig kod, så kallad spagettikod, kan ha en negativ påföljd. Detta ur ett perspektiv som inkluderar flera hållbarhetsdimensioner. Informant B berättar att hen tagit del av information om hållbar mjukvaruutveckling på en konferens där spagettikod beskrevs som negativt ur både arbets- och klimatsynpunkt. Detta då den är svårare att underhålla och vid användning kräver mer hårdvarukapacitet och därmed energi. Informant C lyfter också ett mångsidigt perspektiv kring spagettikod, men lägger större vikt vid de tekniska och ekonomiska aspekterna. Informanten återkommer kontinuerligt till att slarvig kod

innebär ökad underhållstid och således fler arbetstimmar som i sin tur leder till högre kostnader för företaget. I ett kort uttalande uppger hen att detta även har en påverkan på "klimat och samhälle". Med hänsyn till sina uttalanden anser båda informanterna att det finns flera fördelar med att skriva "bra kod". I synnerhet minskad klimatpåverkan genom lägre energiförbrukning från mjukvaran samt enklare underhåll av kod och därigenom ekonomiska vinningar avseende tid.

Trender

Ett annat perspektiv som diskuteras är att trender inom mjukvaruutveckling kan leda till ökad resursanvändning. Informanterna B, E och F lyfter AI som en association kring resurskrävande mjukvara och de sistnämnda diskuterar ytterligare hur dess snabba utveckling är problematisk beträffande påverkan på klimatet. Informant E menar att tekniken kräver stora mängder resurser och att hen observerat att en viss tillhandahållare av AI-tjänster inte nämnde tjänsternas klimatpåverkan vid en lanseringspresentation. Enligt informanten tyder detta på ett negligerande som hen tror bottnar i en växande trend där målet endast är att snabbt lansera AI på marknaden. Vidare lyfter informant F ett perspektiv med fokus på andra sidan myntet; efterfrågan. Informanten menar att många organisationer verkar vilja implementera AI i sina verksamheter eller tjänster och att detta riskerar att leda till en ackumulerad mängd växthusgasutsläpp. I synnerhet ifall varje enskild aktör tränar sina egna AI-modeller och om datorerna i fråga drivs av fossila bränslen. Samtidigt menar informant F att det även finns positiva aspekter med AI såsom automatisering av processer, vilket gör frågan komplex enligt informanten.

En annan typ av trend lyfts av informant B som uppger att vissa visuella val i mjukvaruapplikationer kan leda till ökad energiförbrukning. På samma konferens som nämns i föregående avsnitt blev informanten upplyst om att videor är mer

hårdvarukrävande än bilder. Hen menar att rörlig bild är en designtrend och detta innebär att trender på så sätt har en påverkan på energianvändning. Ett annat exempel som informant B lyfter är att vissa färger kan influera användare till att höja skärmljusstyrkan och på så sätt orsaka användningen av energi.

4.1.3. Mjukvara som verktyg

Ett annat tema avseende hur intervjupersonerna uppfattar förbindelsen mellan miljömässig hållbarhet och mjukvaruutveckling är mjukvarans potentiella användningsområden inom klimatarbetet i stort.

4.1.3.1. Digitalisering

Fem intervjupersoner pekar på hur mjukvara är en hörnsten för digitaliserade företeelser som på olika sätt möjliggjort lägre klimatpåverkan inom diverse områden. Informanterna A, C, D, E och G lyfter digitala möten som ett sätt att minska bilresor och därmed växthusgasutsläpp. Ett annat exempel, lyft av informanterna A, D och G, är mindre pappersanvändning tack vare digital kommunikation. Detta anses minska klimatpåverkan från materialanvändning. Andra exempel berör varierande tillämpningsområden, varav ett är effektivare leveransplanering som informant A menar minskar körsträckor.

4.1.3.2. Puffning

Ett annat perspektiv kring mjukvarans roll i klimatarbetet är beteendeförändring genom puffning (eng. Nudging). Informanterna B och E menar att mjukvara kan konstrueras på ett sätt som gör att konsumenter agerar till fördel för klimatet, varpå båda lyfter visuell återkoppling. Informant B ger ett exempel på en webbutik som

skulle kunna visualisera hur returnering av varor påverkar klimatet för att motverka onödiga köp. Enligt informant E skulle streamingtjänster kunna, genom en symbol, visa vilken typ av energi dess datacenter använder i realtid. Detta för att göra konsumenterna medvetna och ifrågasätta huruvida det är värt att använda tjänsten i fråga vid tillfällen då energikällan inte är "grön".

4.2. Miljömässig hållbarhet beaktas inte

En gemensam uppfattning bland intervjupersonerna är att miljömässig hållbarhet inte beaktas inom mjukvaruutvecklingen. Däremot anses vissa befintliga åtgärder främja minskad klimatpåverkan på ett indirekt sätt. Även barriärer och möjligheter för praktisering av miljömässigt hållbara åtgärder diskuteras.

4.2.1. Indirekta åtgärder

Flera intervjupersoner uttrycker att miljömässig hållbarhet inte är en etablerad företeelse inom mjukvaruutvecklingen, men att de kan se hur vissa åtgärder har en indirekt positiv påverkan. De mest lyfta åtgärderna är koncentrerade kring den övergripande arbetsprocessen och utgörs av arbete hemifrån samt digital kommunikation, vilka enligt intervjupersonerna minskar växthusgasutsläpp från bilresor respektive pappersanvändning. Intervjupersonerna beskriver dessa som indirekta åtgärder då de inte huvudsakligen syftar till att minska klimatpåverkan. Det förstnämnda anses utövas ur bekvämlighetssynpunkt och det andra för att det är ett standardiserat kommunikationssätt.

Vidare lyfts varierande indirekta åtgärder. Informant A uppger att lastbalansering av serveraktivitet har "den trevliga bieffekten" av att fysiska resurser inte går åt lika frekvent tack vare att hårdvara inte kraschar och förbrukas i förtid. Informant E uppger att företagets tjänster distribueras i en icke fysisk form, vilket innebär mindre användning av fysiskt material.

4.2.2. Barriärer och möjligheter

Intervjupersonerna diskuterar varierande aspekter som skulle kunna hindra inkorporeringen av miljömässigt hållbara åtgärder inom mjukvaruutvecklingen, varpå möjliga lösningar även lyfts. Detta med hänsyn till såväl individuell som organisatorisk nivå.

4.2.2.1. Medvetenhet och kunskap

Bristande medvetenhet om mjukvaruutvecklingens klimatpåverkan anses vara en stor barriär för att miljömässigt hållbara åtgärder ska kunna inkorporeras. Även bristen på kunskap om åtgärder. Detta uttrycks främst ur ett individuellt perspektiv, varpå en återkommande möjlighet som föreslås är information. Informanterna C, D, E och F beskriver detta som tydliga exempel, tillhandahållna av företaget genom utbildning. Främst i form av föreläsningar och seminarier. Ett annat format, lyft av informant B, är fysiska mötesplatser där expertkunskap kan erhållas och idéer utbytas:

> Gärna någonting som inte bara finns på vårt företag, utan är öppet för alla. Så man kan förlita sig på någon som faktiskt är expert. Samlad expertkunskap någonstans. [...] Inte forum som onlineforum, utan forum som i mötesplats på något sätt där folk som är miljöintresserade och som kanske har teknisk kunskap kan mötas (Informant B).

Vidare lyfts individuell kunskapsspridning som ett annat sätt för att öka medvetenheten och kunskapen inom företaget. Informant B har sett att detta fungerat gällande tillgänglighet, det vill säga aspekter kring design som tar hänsyn till individers olika funktionsvariationer. Informanten berättar att kollegor visat intresse när hen lyft ämnet och att det potentiellt skulle kunna fungera gällande klimatfrågor. Detta resonerar med ett uttalande från informant E:

> Att få en att tänka i lite nya banor [...] så man får en liten sån där aha-upplevelse. I alla fall för mig är det oftast bara det som krävs (Informant E).

4.2.2.2. Organisatoriskt stöd

En annan barriär avseende individuell nivå är avsaknad av organisatoriskt stöd. Det finns en genomgående åsikt att ansvaret för inkorporeringen av miljömässigt hållbara åtgärder inom mjukvaruutvecklingen ligger hos både företaget och individen, men att det är företagets ansvar att initiera. Resonemanget är att medarbetarna behöver ges de nödvändiga förutsättningarna. I detta avseende lyfts aspekter kring tid, tydlig vägledning och enkelhet.

Enligt informanterna A, B, C och F skulle uttalade riktlinjer innebära befogenheten att utforska ämnet. Informanterna C och F menar att det rådande arbetet kantas av tidsbrist, vilket de anser hämmar möjligheten att introducera nya arbetssätt överlag. Vidare anser informant C att riktlinjerna även bör vara tydliga. Informanten hänvisar till nuvarande hållbarhetsmål som diffusa, vilket hen menar leder till att målen inte resulterar i praktiska åtgärder. De bör enligt informanterna B, D och G även vara enkla att integrera i bemärkelsen att de inte försvårar arbetet. Informant B hänvisar till post-it-lappar som hen använder i sitt arbete, varpå informanten menar skulle göra arbetsprocessen svårare om dessa förbjöds. Informanterna D och G

diskuterar risken med att integrera åtgärder för snabbt och drar en parallell till transportsektorn där höga bensinpriser. De menar att en abrupt omställning inom mjukvaruutvecklingen skulle kunna ge upphov till irritation och hämma intresse och kanske även resultera i uppsägningar.

4.2.2.3. Organisatorisk vilja

En tredje barriär som lyfts av samtliga intervjupersoner är organisatorisk ovilja. I detta avseende diskuteras barriärer och möjligheter kring huruvida individuella förslag skulle kunna leda till organisatoriska insatser. Även potentialen kring högre direktiv.

Det finns en gemensam uppfattning om att företaget skulle ta frågan om miljömässigt hållbara åtgärder seriöst om den lyftes av anställda, men att det finns barriärer som kan hämma handling. Informanterna A, B, F, G uttrycker att det troligen skulle vara svårt att få igenom åtgärder som innebär höga kostnader. Informant B menar att det i detta avseende är desto viktigare att det finns en gedigen förståelse för vad miljömässig hållbarhet innebär i kontexten av mjukvaruutveckling då övergripande eller diffusa förslag annars blir okända kostnader. En annan barriär är att förslag inte visar en tydlig nytta för företaget, varpå informanterna A, B och F lyfter marknadsföring som ett möjligt argument; ett sätt för företaget att sticka ut i kundens ögon. Informanterna A och B uttrycker även vidare att detta skulle främjas av kundönskemål. Informant A menar att företaget styrs av vad kunden vill ha och att kundkrav troligen är det som krävs främst. Att detta skulle ske i en nära framtid är informant F skeptisk till. Hen menar att kommuner, som är företagets kunder, tar lång tid på sig att fatta beslut:

> Kommuner och sånt är så tröga. Det händer väldigt långsamt, händer det saker. Visst, det kan ske saker snabbt inom vissa avdelningar, men i det stora hela är det

> trögt. Det krävs så mycket kommunikation emellan och det är så många olika delar i kommuner. Men handlar det bara om IT-avdelningen så skulle, alltså min generella känsla är att de nog är trögast att röra på sig utav alla (Informant F).

Flera intervjupersoner uttrycker att direktiv uppifrån skulle kunna främja organisatorisk vilja. Ett förslag av informanterna C, E, F och G är bestämmelser från moderbolaget som koncernens företag måste respektera. Om detta inte fungerar anser informanterna E och F att lagar kan behövas som en sista utväg. Informant E menar att detta är ett snabbt och effektivt sätt generellt sett medan informant F anser att det kan vara särskilt viktigt gällande företag som inte beaktar sin verksamhets klimatpåverkan utan tvång. Lagar diskuteras även av informant B som hänvisar till ett EU-direktiv om tillgänglighet inom digitala tjänster. Direktivet resulterade i svensk lagstiftning och informanten menar att liknande skulle kunna införas för miljömässigt hållbar mjukvaruutveckling.

4.2.3. Potentiella åtgärder

Flera intervjupersoner diskuterar hur miljömässigt hållbara åtgärder skulle kunna inkorporeras i mjukvaruutvecklingen. Informanterna B, C och F betonar vikten av bra planering under mjukvaruutvecklingens tidiga stadie. Detta för att undvika utveckling av ineffektiv mjukvara på grund av tidsbrist. För detta föreslår informant B checklistor vid design av mjukvarans visuella delar och informant C pekar på en existerande process i anslutning till kravinsamlingsprocessen där ämnad funktionalitets för- och nackdelar redan reflekteras kring. Vidare belyser informant C även agila utvecklingsmodeller som fördelaktiga avseende planering. Detta med fokus på kontinuerliga avstämningar.

Beträffande tekniska aspekter, diskuterar informant F möjligheter kring att inventera servrar för att ta bort saker som inte används. Informanten uppger att det finns verktyg som kan användas för att göra detta på ett automatiserat sätt.

4.3. Betydelse

Intervjupersonerna uttrycker varierande tankar om betydelsen av att praktisera mjukvaruutveckling med hänsyn till klimatet. Både gällande det egna arbetet och klimatarbetet i stort.

4.3.1. Parallella fördelar för det egna arbetet

Med hänsyn till tidigare avsnitt ser flera intervjupersoner parallella fördelar med att beakta mjukvaruutvecklingens klimatpåverkan. Det finns en uppfattning om att detta kan bidra till välskriven kod som vidare anses minska mjukvarans energiförbrukning och därmed klimatpåverkan samt göra koden enklare underhålla, vilket i sin tur betraktas som fördelaktigt arbetsmässigt och ekonomiskt sett (se 4.1.2.2. Mjukvarans resursanvändning). Enligt informant B finns det dock en komplex balansgång mellan olika nyttor. Informanten hänvisar till att färger kan influera mjukvarans energiförbrukning (se 4.1.2.2. Mjukvarans resursanvändning) och att det blir en fråga om att tumma på användarupplevelsen eller klimatpåverkan. Hen beskriver även vidare att avvägningen mellan nyttorna har med mjukvarans användningsområde och syfte att göra, varpå informanten lyfter företagets tjänster som exempel. Informant B menar att tjänsterna ämnar att förenkla kundernas arbetsprocesser och om de är designade på ett användarvänligt sätt behöver användarna inte spendera lika lång tid i dem, vilket

innebär lägre användning av energi. Detta i kontrast med att bibehålla användarens uppmärksamhet så länge som möjligt såsom i sociala medier. I det avseendet anser informanten att det är desto viktigare att mjukvaran är effektiv i termer av resursanvändning.

4.3.2. Varierande betydelsefullhet för det generella klimatarbetet

Vid tillfrågan hävdar samtliga intervjupersoner att miljömässig hållbarhet är viktigt ur ett generellt perspektiv. Däremot uttrycks olika syn på betydelsefullheten av miljömässigt hållbara åtgärder inom mjukvaruutveckling. Informanterna A och B anser att "allt spelar roll" och att det är viktigt att även mjukvaruutvecklingens klimatpåverkan uppmärksammas. En gemensam nämnare mellan dessa informanter är att de i jämförelse med andra självmant uttrycker miljömässiga hållbarhetsfrågor som intressanta och viktiga generellt sett. Detta tar sig uttryck genom uttalanden som:

> Jag tycker de här frågorna är väldigt intressanta och just att det är viktigt att man inte bara lutar sig tillbaka (Informant A).

> Jag tycker att det här är intressant. Jag kan väl beskylla min bästa kompis. [...] Hon är väldigt miljömedveten [...] så hon har lärt en väldigt mycket. [...] Det oroar mig när folk inte tar det på allvar (Informant B).

Informanterna C och G är mer skeptiska. Informant C menar att andra industrier troligen ger upphov till mer växthusgasutsläpp och pekar på transportsektorn som viktigare att adressera. Detta håller informant G med om som är "absolut tveksam till" ifall minskande av växthusgasutsläppen från mjukvaruindustrin skulle ha samma påverkan som att minska bilresor. Vidare uttrycker informant E också en form av

skepsis. Hen menar att det troligen är effektivare att nyttja mjukvara för att minska klimatpåverkan inom andra domäner, än att fokusera på att minska mjukvaruutvecklingen eller mjukvarans.

5. Diskussion

I detta kapitel diskuteras studiens resultat i relation till litteraturöversikten. Kapitlet innehåller även en reflektion kring studiens begränsningar, dess bidrag till forskningen samt tankar om framtida forskning.

5.1. Resultatdiskussion

Denna studie syftade till att undersöka hur mjukvaruutövare på ett IT-företag uppfattade och såg på miljömässig hållbarhet inom mjukvaruutveckling. Följande avsnitt diskuterar detta utifrån studiens resultat och dess koppling till litteraturöversiktens innehåll.

5.1.1. Medvetenhet och kunskap

Intervjupersonerna förknippade miljömässig hållbarhet i kontexten av mjukvaruutveckling med två huvudsakliga företeelser: (1) påverkan på klimatet genom resursanvändning från arbetsprocessen och mjukvaran vid användning samt (2) att mjukvara kan användas som verktyg för att minimera klimatpåverkan. Beträffande den förstnämnda lyftes även förebyggande åtgärder. Båda företeelserna ligger inom de koncept som litteraturen benämner GISW respektive GBSW (Calero & Piattini, 2017)

och flera av åtgärderna överlappar de i litteraturöversikten. Däribland hantering av serveraktivitet för att minska energiförbrukning och tidig kassering av hårdvaran (Naumann et al., 2011; Kumar et al., 2022) samt kodning som minskar energiförbrukningen (Sun et al., 2011; Palomba et al., 2019; Sehgal et al., 2022). Majoriteten av intervjupersonerna uttryckte dessa aspekter som antaganden då de uppfattade kopplingen mellan miljömässig hållbarhet och mjukvaruutveckling som svår att se. Med hänsyn till detta och aspekternas förbindelse med litteraturen verkar det finnas en viss undermedveten förståelse. Huruvida förståelsen kan betraktas som djupgående är diskutabelt, men i relation till tidigare studier rimmar den med resultaten av Ahmad (2022) där de flesta studiedeltagarna visade en generellt hög medvetenhet och kunskap om miljömässigt hållbara åtgärder. Andra studier visade ett motsatt resultat (Chitchyan et al., 2016; Groher & Weinreich, 2017; Karita et al., 2019; Noman et al., 2022). En förklaring till denna variation kan vara exponering för frågor i avseendet som Chitchyan et al. (2016) menar. Detta resonerar både med fynden av Ahmad (2022) och denna studie. Ahmad (2022) identifierade att familjariteten med miljömässigt hållbar mjukvaruutveckling härstammade från att studiedeltagarna på något sätt hade exponerats för ämnet. Synnerligen genom arbetet eller andra källor. I denna studie var det två intervjupersoner som lyfte att de tagit del av information om ämnet tidigare, vilka även uppgav relativt djupa svar. Andra kunde också göra det, men hade å andra sidan inte reflekterat kring miljömässigt hållbar mjukvaruutveckling tidigare. Vad detta beror på är inte självklart. En förklaring kan vara att de exponerats för anknutna ämnen på något sätt utan större reflektion.

Ett intressant fynd som inte uttryckts av studiedeltagare i tidigare studier är intervjupersonernas uppfattning om att AI kan leda till ökad resursanvändning och därmed klimatpåverkan. AI-tjänster är en relativt ny företeelse som verkar ha en

betydande påverkan på klimatet (MIT Technology Review, 2019). Förutom att det finns en tydlig medvetenhet kring detta, har en intervjuperson observerat att AI-leverantörer tenderar att negligera problematiken. Denna studie tar inte ställning till huruvida detta negligerande faktiskt förekommer, men om så är fallet är det djupt problematiskt avseende IKT-sektorns klimatpåverkan.

5.1.2. Miljömässig hållbarhet beaktas inte

En central uppfattning bland intervjupersonerna var att miljömässig hållbarhet inte beaktas inom mjukvaruutvecklingen, vilket bekräftar tidigare studiers fynd (Chitchyan et al., 2016; Groher & Weinreich, 2017; Karita et al., 2019; Noman et al., 2022). I detta avseende uttrycktes barriärer och möjligheter för praktisering, varav samtliga lyfts i tidigare studier. En var bristande medvetenhet och kunskap om såväl mjukvaruutvecklingens klimatpåverkan som anknutna åtgärder. En möjlig lösning som majoriteten av intervjupersonerna uttryckte var tydlig information via företaget. Detta i form av utbildning, vilket även Chitchyan et al. (2016) föreslår. Således bekräftas förslaget som ett faktiskt behov. Vidare lyfte en intervjuperson fysiska mötesplatser för att kunna erhålla expertkunskap. Detta visar att kunskapsöverföring från akademin, som Noman et al. (2022) förespråkar, är önskat i praktiken. Det synliggör även ett format för hur kunskapsöverföringen skulle kunna gå till. Ett annat förslag som lyftes var kunskapsspridning, vilket även det resonerar med Chitchyan et al. (2016).

En annan barriär som lyftes var brist på organisatoriskt stöd i form av tydliga riktlinjer med betoning på tydlighet. Detta rimmar med studiedeltagarnas uppfattningar i studien av Chitchyan et al. (2016). Intervjupersonerna menade även att detta skulle innebära befogenheten att undersöka miljömässigt hållbara åtgärder, vilket bekräftar resonemanget att aspekter utöver tekniska annars lätt förbises som Noman et

al. (2022) hävdar. Ett resonemang som dock är nytt avseende riktlinjer i relation till tidigare studier är att riktlinjerna inte bör försvåra arbetet. Detta ansågs kunna hämma intresset för miljömässigt hållbara åtgärder och potentiellt resultera i uppsägningar.

Den sista barriären som intervjupersonerna lyfte fram var organisatorisk ovilja, där det fanns en uppfattning om att höga kostnader skulle kunna hindra organisatoriskt stöd. Samma resonemang lyftes av studiedeltagare i studien av Chitchyan et al. (2016). Vidare såg intervjupersonerna även potentiella hinder med förslag som inte visar en tydlig nytta för företaget. Detta bekräftar resonemanget att åtgärdernas fördelar bör demonstreras på ett tydligt sätt, framhävt av Chitchyan et al. (2016). En nytta som flera intervjupersoner lyfte var marknadsföring i syfte att attrahera kunder, vilket även betraktades som en positiv aspekt av studiedeltagare i studien av Karita et al. (2019). Kundkrav, i bemärkelsen att kunder begär miljömässigt hållbara produkter, upplevdes vidare av en intervjuperson som en betydande förutsättning för att uppnå organisatorisk vilja. Detta då företaget styrs av vad kunderna vill ha enligt intervjupersonen. Liknande resonemang återfinns i studien av Chitchyan et al. (2016) där det bland studiedeltagare fanns en tendens att frånsäga sig ansvaret att agera miljömässigt hållbart till kunden. Detta verkar dock inte vara fallet i denna studie eftersom samtliga intervjupersoner uttryckte att ansvaret var delat. Däremot ansågs det vara företagets ansvar att initiera ett agerande, vilket även Chitchyan et al. (2016) identifierar i sin studie. Om detta inte sker eller ifall nämnda angreppssätt fallerar är lagstiftning en möjlighet enligt flera intervjupersoner. Detta är en ny insikt i relation till tidigare studier.

5.1.3. Betydelse

Ett sista övergripande tema som diskuterades av intervjupersonerna var betydelsen av att beakta miljömässig hållbarhet inom mjukvaruutveckling. Det fanns en uppfattning om att beaktande av klimatpåverkan innebär välskriven kod och att detta medför flera fördelar. I synnerhet lägre klimatpåverkan och effektivare underhållbarhet av kod, vilket även ansågs vara fördelaktigt ur ett arbetsmässigt och ekonomiskt perspektiv. Detta rimmar med uppfattningen hos studiedeltagarna i studierna av Karita et al. (2019) och Ahmad (2022).

Ett annat centralt diskuterat tema avseende betydelse var betydelsefullheten av att adressera mjukvaruutvecklingens klimatpåverkan. I detta avseende rådde det olika uppfattningar. Vissa menade att företeelsen var viktig avseende klimatarbetet i stort medan andra ansåg att det finns viktigare insatsområden, synnerligen transportsektorn. Varierande åsikter kan även återfinnas i studierna av Karita et al. (2019), Noman et al. (2022) och Chitchyan et al. (2016). Skillnaden kan bero på olika intresse för hållbarhetsfrågor såsom Chitchyan et al. (2016) menar. Detta då de intervjupersoner som betraktade miljömässigt hållbar mjukvaruutveckling som viktigt också uttryckte intresse för miljömässig hållbarhet.

5.2. Begränsningar

Studien har begränsningar som bör tas i beaktande. Då den genomfördes som en enfallsstudie begränsas generaliserbarheten, trots att fallet speglar hur mjukvaruutveckling vanligen tar sig uttryck i praktiken. Däremot kan fynden användas för analytisk generalisering i framtida forskning (Yin, 2013). Resultatens trovärdighet

är också utmanad, men stärks genom triangulering med fynd från tidigare studier (Miles et al., 2013). En annan aspekt som begränsar studien är intervjuguidens innehåll. Dess fördefinierade frågor har påverkat resultaten och kan potentiellt ha gett upphov till att vissa uppfattningar inte täckts. Eventuella missförstånd vid analys av intervjupersonernas svar är också en möjlig begränsning.

5.3. Bidrag till forskningen och framtida forskning

Att skapa inblick i mjukvaruutövares uppfattningar om miljömässig hållbarhet i kontexten av mjukvaruutveckling är viktigt för att förstå och synliggöra aspekter som kan påverka antagandet av miljömässigt hållbara åtgärder i praktiken, såväl positivt som negativt. Litteraturen visar att både verktyg och metoder har tagits fram, men att de inte används eller praktiseras generellt sett. Denna studie bekräftar detta samt vad studier anser hämma och främja praktisering. Studien bidrar även med nya insikter i avseendet och adderar till kunskapsläget som idag är begränsat.

För framtida forskning vore det intressant att undersöka hur framtagna verktyg och metoder faktiskt påverkar mjukvaruutvecklingen. Detta skulle kunna ge en förståelse för huruvida de uttryckta barriärerna, såsom ökade kostnader och försvårande av arbetet, faktiskt inträffar och i vilken utsträckning. Ett annat förslag är att undersöka AI-leverantörers uppfattningar om miljömässig hållbarhet i kontexten av sina tjänster eller AI-tjänster i stort. En sådan studie skulle kunna ge insikt i huruvida ett negligerande av teknikens klimatpåverkan är en realitet samt hur detta anses kunna hanteras.

6. Slutsatser

Studiens frågeställning “vilka centrala uppfattningar uttrycker mjukvaruutövare på ett IT-företag om miljömässig hållbarhet i kontexten av mjukvaruutveckling?” kan sammanfattas genom slutsatserna nedan.

Mjukvaruutövarna:

- Förknippar miljömässig hållbarhet i kontexten av mjukvaruutveckling med klimatpåverkan från såväl arbetsprocessen som mjukvaran samt att mjukvara kan användas som verktyg för att främja det generella klimatarbetet. Utöver denna medvetenhet visar de även en viss förståelse för hur mjukvaruutvecklingens påverkan på klimatet kan angripas.
- Anser att miljömässig hållbarhet inte beaktas inom mjukvaruutvecklingen och att det finns barriärer och möjligheter som kan hämma respektive främja praktisering.
- Ser flera fördelar med att praktisera miljömässigt hållbar mjukvaruutveckling.
- Har olika syn på betydelsefullheten av att beakta mjukvaruutvecklingens klimatpåverkan med hänsyn till klimatarbetet i stort. Vissa betraktar det som intressant och viktigt medan andra är skeptiska till dess betydelsefullhet avseende det generella klimatarbetet.

Det är viktigt att minska IKT-sektorns ökande växthusgasutsläpp för att bromsa den globala uppvärmningen. Fynden ovan visar en passivitet i praktiken avseende mjukvaruutveckling, som behöver adresseras. Även upplevda barriärer och möjligheter för att det ska kunna ske. De synliggör också att det finns en varierande uppfattning kring dess betydelsefullhet gällande det generella klimatarbetet. Därmed kan denna studie betraktas som både en vädjan till relevanta aktörer att agera och en indikator på vilka aspekter som kan avgöra framgångsrika interventioner ur mjukvaruutövares perspektiv. Som avslutande mening passar det att låna ett uttalande av Karita et al. (2019):

> Företag bör inte betraktas som "gröna" förrän de börjar beakta mjukvaruutvecklingens klimatpåverkan.

7. Tack

För det första vill jag tacka min handledare Andrius Plepys samt Belinda Bergstedt, Chanram Persson och Jan Persson för ert gränslösa stöd i form av reflektioner och vägledning. Sedan vill jag även rikta ett stort tack till intervjupersonerna för er medverkan. Ni alla möjliggjorde denna studie.

8. Referenser

Agosta, G., Bessi, M., Capra, E., & Francalanci, C. (2012). Automatic memoization for energy efficiency in financial applications. *Sustainable Computing: Informatics and Systems*, *2*(2), 105-115. https://doi.org/10.1016/j.suscom.2012.02.002

Ahmad, R. (2022). Software Sustainability Practices and Awareness Amongst Software Practitioners in Malaysia: An Exploratory Study. *2022 2nd International Conference on Electronic and Electrical Engineering and Intelligent System (ICE3IS)*, 192-197. https://doi.org/10.1109/ICE3IS56585.2022.10010195

Ahmad Ibrahim, S. R., Yahaya, J., & Sallehudin, H. (2022). Green Software Process Factors: A Qualitative Study. *Sustainability*, *14*(18), 11180. https://doi.org/10.3390/su141811180

Ali, S. S., & Choi, B. J. (2020). State-of-the-Art Artificial Intelligence Techniques for Distributed Smart Grids: A Review. *Electronics*, *9*(6), 1030. https://doi.org/10.3390/electronics9061030

Anbarkhan, S. H. (2023). A Fuzzy-TOPSIS-Based Approach to Assessing Sustainability in Software Engineering: An Industry 5.0 Perspective. *Sustainability*, *15*(18), 13844. https://doi.org/10.3390/su151813844

Bambazek, P., Groher, I., & Seyff, N. (2022). Sustainability in Agile Software Development: A Survey Study among Practitioners. *2022 International Conference on ICT for Sustainability (ICT4S)*, 13-23. https://doi.org/10.1109/ICT4S55073.2022.00013

Beghoura, M. A., Boubetra, A., & Boukerram, A. (2017). Green software requirements and measurement: random decision forests-based software energy consumption profiling. *Requirements Engineering*, *22*, 27-40. https://doi.org/10.1007/s00766-015-0234-2

Belkhir, L., & Elmeligi, A. (2018). Assessing ICT global emissions footprint: Trends to 2040 & recommendations. *Journal of Cleaner Production*, *177*, 448-463. https://doi.org/10.1016/j.jclepro.2017.12.239

Bigelow, S. J. (2024, 26 mars). *Containers vs. VMs: What are the key differences?*. https://www.techtarget.com/searchitoperations/tip/Containers-vs-VMs-What-are-the-key-differences

Bryman, A. (2012). *Social Research Methods* (4. uppl.). Pearson Education.

Bryman, A. (2018). *Samhällsvetenskapliga metoder* (3. uppl.). Liber.

Calero, C., & Piattini, M. (2017). Puzzling out Software Sustainability. *Sustainable Computing: Informatics and Systems*, *16*, 117-124. https://doi.org/10.1016/j.suscom.2017.10.011

Cappiello, C., Ho, N. T. T., Pernici, B., Plebani, P., & Vitali, M. (2016). CO2-Aware Adaptation Strategies for Cloud Applications. *IEEE Transactions on Cloud Computing*, *4*(2), 152-165. https://doi.org/10.1109/TCC.2015.2464796

Capra, E., Francalanci, C., & Slaughter, S. A. (2012). Is software “green”? Application development environments and energy efficiency in open source applications. *Information and Software Technology*, *54*(1), 60-71. https://doi.org/10.1016/j.infsof.2011.07.005

Chauhan, N. S., & Saxena, A. (2013). A Green Software Development Life Cycle for Cloud Computing. *IT Professional*, *15*(1), 28-34. https://doi.org/10.1109/MITP.2013.6

Chitchyan, R., Becker, C., Betz, S., Duboc, L., Penzenstadler, B., Seyff, N., & Venters, C. C. (2016). Sustainability Design in Requirements Engineering: State of Practice. *2016 IEEE/ACM 38th International Conference on Software Engineering Companion (ICSE-C)*, 533-542. https://doi.org/10.1145/2889160.2889217

Cioca, M., & Schuszter, I. C. (2022). A System for Sustainable Usage of Computing Resources Leveraging Deep Learning Predictions. *Applied Sciences*, *12*(17), 8411. https://doi.org/10.3390/app12178411

D'Agostino, D., Merelli, I., Aldinucci, M., & Cesini, D. (2021). Hardware and Software Solutions for Energy-Efficient Computing in Scientific Programming. *Scientific Programming*, *2021*. https://doi.org/10.1155/2021/5514284

Dick, M., Drangmeister, J., Kern, E., & Naumann, S. (2013). Green software engineering with agile methods. *2013 2nd International Workshop on Green and Sustainable Software (GREENS)*, 78-85. https://doi.org/10.1109/GREENS.2013.6606425

Erdélyi, K. (2013). Special factors of development of green software supporting eco sustainability. *2013 IEEE 11th International Symposium on Intelligent Systems and Informatics (SISY)*, 337-340. https://doi.org/10.1109/SISY.2013.6662597

Europeiska kommissionen. (u.å.). *ICT Environmental impact (RP2023)*. Hämtad den 14 mars 2024 från https://joinup.ec.europa.eu/collection/rolling-plan-ict-standardisation/ict-environmental-impact-rp2023

Förenta nationerna. (u.å.-a). *Causes and Effects of Climate Change*. Hämtad den 11 mars 2024 från https://www.un.org/en/climatechange/science/causes-effects-climate-change

Förenta nationerna. (u.å.-b). *Climate Change*. Hämtad den 11 mars 2024 från https://www.un.org/en/global-issues/climate-change

Förenta nationerna. (u.å.-c). *Transforming our world: the 2030 Agenda for Sustainable Development*. Hämtad den 15 april 2024 från https://sdgs.un.org/2030agenda

García-Berná, J. A., Ouhbi, S., Fernández-Alemán, J. L., Carrillo de Gea, J. M., & Nicolás, J. (2021). Investigating the Impact of Usability on Energy Efficiency of Web-based Personal Health Records. *Journal of Medical Systems*, *45*(65). https://doi.org/10.1007/s10916-021-01725-8

Green Software Foundation. (2024, 25 april). *Software Carbon Intensity (SCI) Specification Achieves ISO Standard Status, Advancing Green Software Development*. https://greensoftware.foundation/articles/sci-specification-achieves-iso-standard-status

Green Software Foundation. (u.å.). *Software Carbon Intensity (SCI) Specification*. Hämtad den 17 maj 2024 från https://sci.greensoftware.foundation/

Groher, I., & Weinreich, R. (2017). An Interview Study on Sustainability Concerns in Software Development Projects. *2017 43rd Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, 350-358. https://doi.org/10.1109/SEAA.2017.70

Haraty, R. A., & Hu, G. (2018). Software process models: a review and analysis. *International Journal of Engineering and Technology*, *7*(2.29), 325-331. https://doi.org/10.14419/ijet.v7i2.29.13206

Hasan, M. S., Alvares, F., Ledoux, T., & Pazat, J-L. (2017). Investigating Energy Consumption and Performance Trade-Off for Interactive Cloud Application. *IEEE Transactions on Sustainable Computing*, *2*(2), 113-126. https://doi.org/10.1109/TSUSC.2017.2714959

Hoda, R., Noble, J., & Marshall, S. (2012). Self-Organizing Roles on Agile Software Development Teams. *IEEE Transactions on Software Engineering*, *39*(3), 422-444. https://doi.org/10.1109/TSE.2012.30

Internationella energirådet. (u.å.). *Executive summary*. Hämtad den 14 mars från https://www.iea.org/reports/world-energy-outlook-2023/executive-summary

Karita, L., Mourão, B. C., & Machado, I. (2019). Software industry awareness on green and sustainable software engineering: a state-of-the-practice survey. *SBES '19: Proceedings of the XXXIII Brazilian Symposium on Software Engineering*, 501-510. https://doi.org/10.1145/3350768.3350770

Kumar, C., Marston, S., Sen, R., & Narisetty, A. (2022). Greening the Cloud: A Load Balancing Mechanism to Optimize Cloud Computing Networks. *Journal of Management Information Systems*, *39*(2), 513-541. https://doi.org/10.1080/07421222.2022.2063551

Lantz, A. (2013). *Intervjumetodik* (3. uppl.). Studentlitteratur AB.

Mancebo, J., Calero, C., Garcia, F., Moraga, M. A., & Garcia-Rodriguez de Guzman, I. (2021). FEETINGS: Framework for Energy Efficiency Testing to Improve Environmental Goal of the Software. *Sustainable Computing: Informatics and Systems*, *30*, 100558. https://doi.org/10.1016/j.suscom.2021.100558

Miles, M. B., Huberman, A. M., & Saldaña, J. (2013). *Qualitative Data Analysis: A Methods Sourcebook* (3. uppl.). SAGE Publications, Inc.

MIT Technology Review. (2019, 6 juni). *Training a single AI model can emit as much carbon as five cars in their lifetimes*. https://www.technologyreview.com/2019/06/06/239031/training-a-single-ai-model-can-emit-as-much-carbon-as-five-cars-in-their-lifetimes/

Moreno-Vozmediano, R., Montero, R. S., Huedo, E., & Llorente, I. M. (2017). Orchestrating the Deployment of High Availability Services on Multi-zone and Multi-cloud Scenarios. *Journal of Grid Computing*, *16*, 39-53. https://doi.org/10.1007/s10723-017-9417-z

Munoz, D-J., Pinto, M., & Fuentes, L. (2018). Finding correlations of features affecting energy consumption and performance of web servers using the HADAS eco-assistant. *Computing*, *100*, 1155-1173. https://doi.org/10.1007/s00607-018-0632-7

Naumann, S., Dick, M., Kern, E., & Johann, T. (2011). The GREENSOFT Model: A reference model for green and sustainable software and its engineering. *Sustainable Computing: Informatics and Systems*, *1*(4), 294-304. https://doi.org/10.1016/j.suscom.2011.06.004

Noman, H., Mahoto, N. A., Bhatti, S., Abosaq, H. A., Al Reshan, M. S., & Shaikh, A. (2022). An Exploratory Study of Software Sustainability at Early Stages of Software Development. *Sustainability*, *14*(14), 8596. https://doi.org/10.3390/su14148596

Noureddine, A., Rouvoy, R., & Seinturier, L. (2014). Monitoring energy hotspots in software. *Automated Software Engineering*, *22*, 291-332. https://doi.org/10.1007/s10515-014-0171-1

Palomba, F., Di Nucci, D., Panichella, A., Zaidman, A., & De Lucia, A. (2019). On the impact of code smells on the energy consumption of mobile applications. *Information and Software Technology*, *105*, 43-55. https://doi.org/10.1016/j.infsof.2018.08.004

Pawlish, M., Varde, A. S., Robila, S. A., & Ranganathan, A. (2014). A call for energy efficiency in data centers. *ACM SIGMOD Record*, *43*(1), 45-51. https://doi.org/10.1145/2627692.2627703

Pereira, R., Couto, M., Ribeiro, F., Rua, R., Cunha, J., Fernandes, J. P., & Saraiva, J. (2021). Ranking programming languages by energy efficiency. *Science of Computer Programming*, *205*. https://doi.org/10.1016/j.scico.2021.102609

Procaccianti, G., Fernández, H., & Lago, P. (2016). Empirical evaluation of two best practices for energy-efficient software development. *Journal of Systems and Software*, *117*, 185-198. https://doi.org/10.1016/j.jss.2016.02.035

Procaccianti, G., Lago, P., & Bevini, S. (2015). A systematic literature review on energy efficiency in cloud software architectures. *Sustainable Computing: Informatics and Systems*, *7*, 2-10. https://doi.org/10.1016/j.suscom.2014.11.004

Saboor, A., Mahmood, A. K., Omar, A. H., Hassan, M. F., Shah, S. N. M., & Ahmadian, A. (2021). Enabling rank-based distribution of microservices among containers for green cloud computing environment. *Peer-to-Peer Networking and Applications*, *15*, 77-91. https://doi.org/10.1007/s12083-021-01218-y

Saunders, M. N. K., Lewis, P., & Thornhill, A. (2023). *Research Methods for Business Students* (9. uppl.). Pearson.

Şanlıalp, İ., Öztürk, M. M., & Yiğit, T. (2022). Energy Efficiency Analysis of Code Refactoring Techniques for Green and Sustainable Software in Portable Devices. *Electronics*, *11*(3), 442. https://doi.org/10.3390/electronics11030442

Saravanan, T., Jha, S., Sabharwal, G., & Narayan, S. (2020). Comparative Analysis of Software Life Cycle Models. *2020 2nd International Conference on Advances in Computing, Communication Control and Networking (ICACCCN)*, 906-909. https://doi.org/10.1109/ICACCCN51052.2020.9362931

Sehgal, R., Mehrotra, D., Nagpal, R., & Sharma, R. (2022). Green software: Refactoring approach. *Journal of King Saud University - Computer and Information Sciences*, *34*(7), 4635-4643. https://doi.org/10.1016/j.jksuci.2020.10.022

Serrano, N., Gallardo, G., & Hernantes, J. (2015). Infrastructure as a Service and Cloud Technologies. *IEEE Software*, *32*(2), 30-36. https://doi.org/10.1109/MS.2015.43

Singh, N., & Ogunseitan, O. A. (2022). Disentangling the worldwide web of e-waste and climate change co-benefits. *Circular Economy*, *1*(2). https://doi.org/10.1016/j.cec.2022.100011

Sinha, A., & Das, P. (2021). Agile Methodology Vs. Traditional Waterfall SDLC: A case study on Quality Assurance process in Software Industry. *2021 5th International Conference on Electronics, Materials Engineering and Nano-Technology (IEMENTech)*, 1-4. https://doi.org/10.1109/IEMENTech53263.2021.9614779

Sriraman, G., & Raghunathan, S. (2023). A Systems Thinking Approach to Improve Sustainability in Software Engineering - A Grounded Capability Maturity Framework. *Sustainability*, *15*(11), 8766. https://doi.org/10.3390/su15118766

Sun, Y., Zhao, Y., Song, Y., Yang, Y., Fang, H., Zang, H., Li, Y., & Gao, Y. (2011). Green challenges to system software in data centers. *Frontiers of Computer Science in China*, *5*, 353-368. https://doi.org/10.1007/s11704-011-0369-3

Rashid, N., & Khan, S. U. (2018). Agile practices for global software development vendors in the development of green and sustainable software. *Journal of Software: Evolution and Process*, *30*(10). https://doi.org/10.1002/smr.1964

Rienecker, L., & Jørgensen, P. S. (2014). *Att skriva en bra uppsats* (3. uppl.). Liber AB.

Taina, J. (2010). How Green Is Your Software?. I P. Tyrväinen, S. Jansen, & M. A. Cusumano (Red.), *Lecture Notes in Business Information Processing: Vol. 51. Software business* (s. 151-162). Springer. https://doi.org/10.1007/978-3-642-13633-7_13

United Nations Framework Convention on Climate Change. (2016, 12 augusti). *ICT Sector Helping to Tackle Climate Change*. https://unfccc.int/news/ict-sector-helping-to-tackle-climate-change

Yin, R. K. (2013). *Case Study Research: Design and Methods* (5. uppl.). SAGE Publications, Inc.

9. Bilagor

I detta kapitel presenteras studiens intervjuguide som användes i de semistrukturerade intervjuerna.

Bilaga 1.

Inledande frågor

1. Vad heter du?
2. Vad är din arbetstitel?
3. Kan du kort beskriva dina arbetsuppgifter?

Huvudfrågor

1. Vad tänker du på när jag säger miljömässigt hållbar mjukvaruutveckling?
    - Skulle du säga att det finns en koppling mellan mjukvaruutveckling och klimatpåverkan?
        - Om ja, hur? Om nej, varför?
2. Hur skulle du beskriva betydelsen av miljömässigt hållbar mjukvaruutveckling med avseende på det generella klimatarbetet?

3. Praktiserar du mjukvaruutveckling som tar hänsyn till miljömässig hållbarhet?
    - Om ja, hur? Om nej, vad skulle krävas för att du skulle göra det?
4. Möjliggör din arbetsgivare mjukvaruutveckling som tar hänsyn till miljömässig hållbarhet?
    - Om ja, hur? Om nej, vad tror du de skulle kunna göra?
5. Hur skulle du beskriva företagets och din egen roll i att mjukvara utvecklas på ett sätt som tar hänsyn till miljömässig hållbarhet?
    - Vem bär ansvaret för att det sker?
    - Ser du några potentiella barriärer och/eller möjligheter för att det ska ske?

Avslutande frågor

1. Hur ser du på miljömässig hållbarhet personligen?
2. Är det något du vill tillägga som du kommit på under intervjuns gång?

LUNDS
UNIVERSITET

WWW.CEC.LU.SE
WWW.LU.SE

Lunds universitet

Miljövetenskaplig utbildning
Centrum för miljö- och
klimatforskning
Ekologihuset
223 62 Lund