LUNDS UNIVERSITET
Ekonomihögskolan

Institutionen för informatik

Hantering av teknisk skuld i små- och mikroföretag

Kandidatuppsats 15 hp, kurs SYSK16 i Informatik

Författare: Christopher Andersson
Martin Larsson
Felix Lidforsen

Handledare: Björn Svensson

Rättande lärare: Bo Andersson
Ahmad Ghazawneh

Hantering av teknisk skuld i små- och mikroföretag

ENGELSK TITEL: Managing technical debt in small- and micro enterprises

FÖRFATTARE: Christopher Andersson, Martin Larsson & Felix Lidforsen

UTGIVARE: Institutionen för informatik, Ekonomihögskolan, Lunds universitet

FRAMLAGD: maj, 2019

DOKUMENTTYP: Kandidatuppsats

ANTAL SIDOR: 56

NYCKELORD: Teknisk skuld, Hantering, Små- och mikroföretag, Teknisk skuldhantering,

SAMMANFATTNING (MAX. 200 ORD):
Teknisk skuld är ett begrepp som innebär att framtida kostnader kommer att uppstå för de som väljer att ta genvägar i utvecklingsarbetet. Teknisk skuld liknas ofta vid finansiella lån och behöver inte vara negativt så länge skulden hanteras och inte blir för stor. Den tekniska skulden löses av ”code refactoring” vilket innebär att koden förändras utan att funktionaliteten gör det. Små- och mikroföretag befinner sig i en svår situation då dessa typer av företag inte har samma resurser som större företag och har även en annan typ av organisationsstruktur. Därför ämnar vi i denna uppsatsen att förklara och beskriva hur små- och mikroföretag hanterar sin tekniska skuld trots begräsningarna de har. Kvalitativa data har skaffats genom intervjuer med personer som har erfarenhet med både arbete med teknisk skuld samt erfarenhet av att jobba i små- och mikroföretag. Resultatet av undersökningen visar att små- och mikroföretag inte följer teoretiskt framställda metoder för att hantera teknisk skuld, de använder sig av vissa små beståndsdelar av olika metoder men framhäver att erfarenhet och magkänsla är det viktigaste redskapet för att hantera den tekniska skulden. Hantering av den tekniska skulden är förövrigt ett fortlöpande arbete där skulden endast betalas när den blir ett problem.

Innehåll

Figurer

Tabeller

1 Introduktion

1.1 Bakgrund

År 2018 fanns det i Sverige 34 674 småföretag samt 270 300 mikroföretag som tillsammans anställde 1 320 888 människor (Ekonomifakta, 2018). Små- och mikroföretag utgör tillsammans 26,8 % av alla företag i Sverige (Ekonomifakta, 2018). En del av dessa företag arbetar med att skapa och underhålla mjukvara. Utvecklingsprocessen kräver ibland att genvägar tas för att underlätta arbetsprocessen och nå kortsiktig framgång på bekostnad av förhöjd underhållskostnad i framtiden och således kunna snabbare nå ut till marknaden (Maldonado, Abdalkareem, Shihab & Serebrenik, 2017). Detta fenomen förklaras med hjälp av begreppet teknisk skuld.

Teknisk skuld (TS) är ett begrepp som myntades redan 1992 av Ward Cunningham som förklarar vad vi idag känner till som ”Code Refactoring”. ”Code Refactoring” kallas en process då programkoden omstruktureras utan att man ändrar dess beteende (Cunningham, 1992). Detta gör programmerare för att förbättra omanvändningspotentialen samt göra koden lättare att underhålla (Kim, Hong, Yoon & Lee, 2018).

En studie som utfördes av onlinebetalningsföretaget Stripe (2018) visar att de finansiella förlusterna av teknisk skuld kan bli stora. Studien visade att utvecklare i snitt lägger 17 timmar i veckan på att underhålla kod, utöver de 4 timmar i veckan de spenderar på ”dålig kod” (Stripe, 2018). Detta kostar företag jorden runt sammanlagt 85 miljarder dollar i förlorade intäkter per år (Stripe, 2018). Om skulden växer sig för stor kan, förutom ekonomiska kostnader även säkerhetsproblem uppenbara sig, något som Facebook tvingats hantera då användares konton har varit tillgängliga för hackers och anställda vid ett flertal gånger (Ashford, 2019). Om problemen fortsätter riskerar Facebook att mista användare de behöver för att bedriva sin verksamhet.

1.2 Problemområde

Termen ”skuld” syftar till att den tekniska skulden måste betalas tillbaka förr eller senare (Besker, Martini & Bosch, 2018). Teknisk skuld har även liknats vid monetär skuld då räntefunktionen hos banker liknar hur skulden ökar så länge problemet inte åtgärdas (Kruchten, 2012). Dock behöver teknisk skuld inte vara ett problem utan det kan vara till ens fördel om det exempelvis leder till att en kund får produkten levererad inom en lukrativ tidsram, något som ökar kundnöjdheten (Yli-Huumo, Maglyas & Smolander, 2016). Skulden byggs upp genom att organisationer och utvecklare tar genvägar och producerar suboptimal kod (Yli-Huumo et al., 2016). Det finns olika anledningar till att detta görs; det påskyndar lansering på marknaden (time-to-market), användarna får produkten snabbare vilket leder till att företag får in feedback om sin produkt snabbare, förbättra mjukvaran och spara kapital (Besker, Martini,

Lokuge, Blincoe & Bosch, 2018). Företag tvingas applicera nödlösningar varje gång de revi-derar koden vilket gör skulden desto större och dyrare att betala (Sierra, Tahmid, Shihab & Tsantalis, 2019).

Trots riskerna med teknisk skuld kan det användas som en hävstång för företag som strävar efter att snabbt lansera sin produkt och få resurserna att räcka längre (Tom, Aurum & Vidgen, 2013). Därför kan ett mindre företag behöva dra på sig teknisk skuld för att nå marknaden snabbt, men av samma anledning vara oförmöget att betala tillbaka skulden (Stripe, 2018).

Att utveckla mjukvara är svårt för små- och mikroföretag, särskilt svårt blir det när företagen vill säkerställa kvalitet med avseende för tidsbegränsning, budget och kundnöjdhet (Majchrowski, Ponsard, Saadaoui, Flamand & Deprez, 2016). Brist på mognad i dessa aspekter kan leda till problem som att misslyckas att identifiera graden av ett problems bety-delse, brist på kompetens att lösa problem och resursprioritering (Majchrowski et al, 2016). Små- och mikroföretag skiljer sig från större organisationer genom att de har begränsade samt mindre specialiserade resurser, däremot är de mer flexibla och mer fokuserade på innovation och värdeskapande än större organisationer (Ponsard & Deprez, 2017). De har ofta en platt organisationsstruktur med en fritt flödande ledningsstil som främjar entreprenörskap och in-novation (Richardson & Wangeheim, 2007).

Till skillnad från stora företag har små- och mikroföretag inte tillräckligt med personal för att utföra sekundära uppgifter utöver utvecklingen av produkten (Richardson & Wangeheim, 2007). Bristen på resurser gör att hanterande åtgärder som till exempel code refactoring inte ryms i budgeten, och bidrar till en ökad teknisk skuld hos små- och mikroföretag (Martini & Bosch, 2016).

1.3 Forskningsfråga

Hur hanterar små- och mikroföretag sin tekniska skuld?

1.4 Syfte

Uppsatsens syfte är förklara och beskriva hur små- och mikroföretag arbetar med att hantera teknisk skuld samt hur de arbetar för att se till att skulden inte växer sig för stor och blir okon-trollerbar.

1.5 Avgränsningar

Vår studie avgränsas enbart till företag som primärt ägnar sig åt mjukvaruutveckling.

2 Teori

2.1 Begreppsförklaring

2.1.1 Små- och mikroföretag

För att definiera om företag klassificeras som små- eller mikroföretag kommer vi att använda oss av europeiska kommissionens (2015) definition av små- och mikroföretag.

"Mikroföretag definieras som företag som sysselsätter färre än 10 personer och vars årsomsättning eller balansomslutning inte överstiger 2 miljoner euro." (Europeiska kommissionen, 2015).

"Små företag definieras som företag som sysselsätter färre än 50 personer och vars årsomsättning eller balansomslutning inte överstiger 10 miljoner euro." (Europeiska kommissionen, 2015).

2.1.2 Hantera

Med hantera avser vi minska teknisk skuld med hjälp av Code refactoring.

2.1.3 Betala

Med betala, som i *betalning av teknisk skuld*, avses när tid aktivt läggs ner på att lösa teknisk skuld.

2.2 Teknisk skuld

Teknisk skuld är ett begrepp som myntades år 1992 av Ward Cunningham, och han beskrev det såhär;

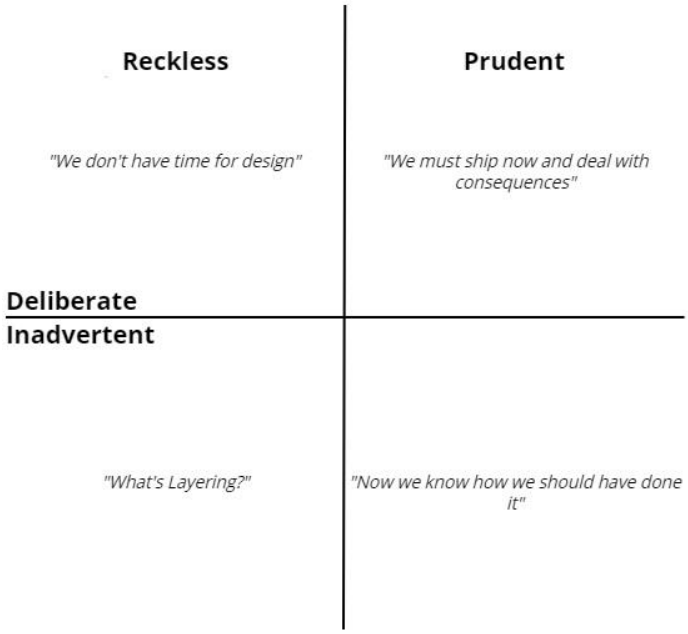
"Shipping first-time code is like going into debt. A little debt speeds development so long as it is paid back promptly with a rewrite. Objects make the cost of this transaction tolerable. The danger occurs when the debt is not repaid. Every minute spent on not-quite-right code counts as interest on that debt. Entire engineering organizations can be brought to a stand-still under the debt load of an unconsolidated implementation, object-oriented or otherwise." - (Cunningham, 1992, p. 29–30)

Teknisk skuld har även liknats vid finansiell skuld då båda skulderna innebär att företag får betala mer ränta ju större skuld företag har (Fowler, 2003). Teknisk skuld behöver dock inte vara något negativt, för precis som företag kan ta ut finansiellt lån för att lösa ett problem kan

företag även ta ett "tekniskt lån" för att exempelvis nå en deadline (Cunningham, 1992). Problemen uppstår när företaget inte betalar tillbaka lånen i tid utan låter skulden växa (Fowler, 2003).

2.2.1 Uppkomsten av teknisk skuld

Antalet definitioner av teknisk skuld är många, men alla definitioner säger att genvägar som tas för att exempelvis lansera en applikation snabbare är teknisk skuld som måste betalas tillbaka vid ett senare tillfälle (Trumler & Paulisch, 2016). Ett välkänt ramverk för uppkomsten av teknisk skuld är Martin Fowlers (2009) "Technical debt quadrant". Denna kvadranten säger att teknisk skuld byggs upp oberoende av medvetna beslut eller oavsiktliga beteenden (Trumler & Paulisch, 2016). Fowlers kvadrant består av 2 dimensioner: Reckless/Prudent och Deliberate/Inadvertent, utifrån dessa dimensioner kan 4 typer av hur teknisk skuld uppstår utläsas ur Fowlers kvadrant (Fowler, 2009):



Figur 1 Illustration återskapad från Fowlers kvadrant (Fowler, 2009)

Reckless and deliberate debt: Den här typen av skuld uppstår när ett team vet bättre men avsiktligt väljer att göra snabba och "fula" lösningar på problem med syftet att göra det så snabbt som möjligt (DevIQ, 2019).

Reckless and inadvertent debt: Den här typen av skuld är den minst önskvärda då teamet inte har något annat val än att ådra sig den, de känner inte igen den eller så kan teamet inte lösa den när skulden väl har uppstått (DevIQ, 2019).

Prudent and deliberate debt: Den här typen av skuld uppstår när teamet kompromissar om kodkvaliteten, då teamet har en deadline att möta men har sett skulden och är beredda på att lösa den i framtiden samt konsekvenserna av att skjuta upp betalningen (DevIQ, 2019).

Prudent and inadvertent debt: Den här typen av skuld representerar vad teamet har lärt sig när de jobbat på ett projekt, i efterhand kan teamet se hur de skulle designat systemet annorlunda för att nå en högre kvalitet (DevIQ, 2019).

Författarna Tom et al. (2013) identifierar faktorer som bidrar till uppkomsten av teknisk skuld, dessa är listade nedan.

Tabell 1 Faktorer som bidrar till uppkomst av teknisk skuld (Tom et al., 2013)

Pragmatism	Pragmatism och prioritering har en stark koppling, att vilja leverera en adekvat produkt på kort sikt (Tom et al., 2013). Även om produkten har den funktionalitet som önskas kan kunden ha långsiktiga mjukvaruproblem som inte uppenbaras för kunden först när det är för sent eller svårt att åtgärda (Tom et al., 2013).
Prioritering	Prioritering kan orsaka teknisk skuld när det blir nödvändigt för utvecklare att prioritera kritiska funktioner istället för kvalitet för att kunna leverera enligt tidsschema (Tom et al., 2013). Team behöver ofta göra avvägningar inom projektets gränser om vad som är viktigt i respektive projekt och det är dessa avvägningar skapar ofta skapar teknisk skuld (Tom et al., 2013).
Processer	Processer som ett team använder sig av kan påverka mängden teknisk skuld (Tom et al., 2013). Är samarbets- och kommunikationsprocesser dåliga blir det lättare att saker glöms bort (Tom et al., 2013).
Attityder	Attityder på individnivå spelar en betydande roll i att bidra med teknisk skuld (Tom et al., 2013). En vanlig anledning till att skuld inte åtgärdas är att utvecklare och ledning är avskräckta för att introducera nya problem (Tom et al., 2013).
Ignorans och överseende	Ignorans och överseende avser mer möjligheten att utvecklare är omedvetna om misstaget eller problemet som skapat teknisk skuld men även att inte veta hur högkvalitativ och ren kod skrivs (Tom et al., 2013).

Både Fowler (2009) och Tom et al. (2013) förklarar beteenden som påverkar uppkomsten av teknisk skuld. De olika förklaringarna om beteenden är förklarade på olika sätt men påminner om varandra; till exempel beskrivningen av pragmatism och prioritering av Tom et al. (2013) liknar det Fowler (2009) beskriver som “Reckless and deliberate debt”.

2.2.2 Typer av teknisk skuld

Det finns likheter mellan grupperingarna som Li, Avgeriou & Liang (2015) och Tom et al. (2013) har gjort kring typer av teknisk skuld, och nedan redogörs de olika författarnas sammanställningar;

Li et al. (2015) samlade en mängd olika typer av teknisk skuld och dess förekomster. De olika typerna av teknisk skuld delades in i 10 olika typer som i sin tur har undertyper vilka kan förknippas med respektive typ av teknisk skuld. Typerna är enligt Li et al. (2015);

Tabell 2 Typer av teknisk skuld I (Li et al., 2015)

Krav teknisk skuld	Avser avståndet mellan den optimala kravspecifikationen och den faktiska implementeringen av systemet (Li et al., 2015).
Arkitektur teknisk skuld	Orsakas av beslut kring arkitektur som påverkar interna kvalitetsaspekter som underhållbarhet (Li et al., 2015).

Design teknisk skuld	Tekniska genvägar som tas i designen (Li et al., 2015).
Kod teknisk skuld	Kod som inte är best practice eller följer god kodsed. Till exempel redundant kod eller överkomplicerad kod (Li et al., 2015).
Test teknisk skuld	Genvägar som tas i testning till exempel otillräcklig testning (enhetstester, integrationstester och acceptanstester) (Li et al., 2015).
Build teknisk skuld	Brister i hur mjukvarusystem körs eller processen som gör mjukvaran överdrivet komplex eller svår (Li et al., 2015).
Dokumentation teknisk skuld	Otillräcklig, ofullständig eller utdaterad dokumentation i någon aspekt av mjukvaruutveckling (Li et al., 2015). Till exempel utdaterad dokumentation om arkitektur eller brist av kodkommentarer (Li et al., 2015).
Infrastruktur teknisk skuld	Suboptimerad konfiguration av utvecklingsrelaterade processer, teknologier, verktyg etcetera (Li et al., 2015). En suboptimerade konfiguration påverkar ett teams förmåga att producera god produktkvalitet (Li et al., 2015).
Versions teknisk skuld	Problem med versionshantering, till exempel när någon tagit källkod från ett annat projekt eller kodbas och återanvänds (Li et al., 2015). Till exempel att behöva underhålla flera olika versioner av en mjukvara (Li et al., 2015).
Defekt teknisk skuld	Defekter, buggar eller misslyckanden i mjukvarusystem (Li et al., 2015).

Teknisk skuld kan enligt Tom et al. (2013) definieras i olika dimensioner. Mjukvaruutvecklings dimensionerna är; kodskuld, design- och arkitekturskuld, omgivningsskuld, kunskapsdelning och dokumentationsskuld och testskuld (Tom et al., 2013). Typerna av teknisk skuld som Tom et al. (2013) nämner förklaras nedan.

Tabell 3 Typer av teknisk skuld II (Tom et al., 2013)

Kodskuld	Kodskuld avser dåligt skriven kod, genvägar eller omvägar som kan ackumulera teknisk skuld (Tom et al, 2013). Den här typen innefattar även redundans av kod, komplexitet, att koden är svår att läsa och dåligt organiserad logik (Tom et al., 2013).
Design- och arkitekturskuld	Den här typen av genvägar eller brister i design kan vara ett resultat av suboptimala lösningar eller att teknologier eller mönster blir ersatta (Tom et al., 2013). En typ av design- och arkitekturskuld kan vara underprioriterat fokus, som till exempel underhåll och anpassningsförmåga (Tom et al., 2013).
Omgivningsskuld	Omgivningsskuld inkluderar utvecklingsrelaterade processer men även hårdvara, infrastruktur eller supportapplikationer som kan ackumulera teknisk skuld (Tom et al., 2013). Manuella processer som har potential att bli automatiserade är en form av teknisk skuld (Tom et al., 2013). Utdaterade komponenter av en applikationsutvecklings- eller operativ miljö bidrar till teknisk skuld (Tom et al., 2013).
Kunskapsdelning och dokumentationsskuld	En annan typ av teknisk skuld är hur kunskap delas och sprids. Till exempel om utvecklare dokumenterar dåligt och lämnar ett projekt blir underhållskostnaderna större (Tom et al., 2013). Bristande eller dålig dokumentation kan bidra till teknisk skuld (Tom et al., 2013).
Testskuld	Teknisk skuld kan uppstå när det är brist på scripts och system måste testas manuellt för varje release (Tom et al., 2013). Testområdet måste vara tillräckligt omfattande, oavsett om det är automatiserad eller manuell testning (Tom et al., 2013). Testskuld kan resultera i ett skadat varumärke när kunder drabbas av kritiska defekter (Tom et al., 2013).

2.2.3 Icke typer av teknisk skuld

I studien "*A systematic mapping study on technical debt and its management*" tar författarna fram sex typer av problem som inte ska kopplas till teknisk skuld (Li et al., 2015). Dessa har tagits fram genom kompilering av data från åtta studier och är följande;

- Defekter
- Icke implementerad funktionalitet
- Brist på stöttande processer
- Oavslutade uppgifter i utvecklingsprocessen
- Allmänna brister i kodkvalité
- Bristande användarupplevelse

Enligt ett antal studier noterar Li et al. (2015) att man inte ser defekt mjukvara och en "broken user experience" som teknisk skuld, då *"TD is about the flaws of the internal quality and invisible to external users, which is not the case for defects; thus defects are not TD."*(Li et al., 2015, p. 9). Bland dessa studier finns *"Estimating the size, cost, and types of Technical Debt"* (Curtis, Sappidi & Szynkarski, 2012), men Li et al. (2015) nämner också att ett mindre fokus har legat på att definiera vad som *inte* är teknisk skuld, och att den ovan nämnda listan därför inte kan anses vida accepterad (Li et al., 2015). På senare år har definitionen av teknisk skuld gått från att beskriva suboptimal kod till att vara ett paraplybegrepp innehållande flertalet andra koncept, och är nu ett bredare begrepp som hanteras i flera segment av utvecklingsprocessen (Kruchten, Nord, Ozkaya & Falessi, 2013).

Forskningen på teknisk skuld är splittrad gällande vilken definition av teknisk skuld som ska gälla (Li et al., 2015). Vi har valt att inte exkludera någon typ av teknisk skuld från uppsatsen för att låta alla definitioner påverka vår slutsats. Den tidigare forskningen läggs fram i sin helhet, med tidigare nämnda skiljaktigheter inkluderade.

2.3 Teknisk skuldhantering

Enligt Li et al. (2015) finns det 8 stycken olika metoder för att hantera teknisk skuld. Dessa metoder kommer redovisas nedan. Teknisk skuldhantering refereras till som TSH i uppsatsen:

- Den första metoden kallas *identifiering,* här identifieras den tekniska skulden genom exempelvis kodanalys och designworkshops (Li et al., 2015).
- Den andra metoden kallas *mätning,* här görs ofta en kostnad-nytta-analys där man avgör om skulden är värd att betala (Li et al., 2015). Nyttoanalysen är ofta subjektiv och svår att mäta men för kostnadsvariabeln finns det mätinstrument (Li et al., 2015). Denna analysen är varken den hela eller objektiva sanningen men syftet med analysen är att starta en diskussion mellan intressenterna (Li et al., 2015).
- Den tredje metoden kallas *prioritering,* och här identifieras de olika skulderna (Li et al., 2015). Sen avgörs vilken skuld som skall prioriteras för att få så hög avkastning som möjligt när den betalas (Li et al., 2015). Denna metod säger således vilken skuld som skall betalas först (Li et al., 2015).
- Den fjärde metod kallas *återbetalning,* här återbetalas skulden man identifierade och prioriterade i den tredje aktiviteten och detta sker genom "code refactoring" (Li et al., 2015).

- Den femte metoden kallas *övervakning,* här övervakas skulder som inte har blivit återbetalda (Li et al., 2015). Detta görs för att skuldernas kostnad och nytta inte är statiska utan förändras över tid och kan bli svåra att hantera om de växer sig för stora (Li et al., 2015).
- Den sjätte metoden kallas *förebyggning* som syftar till att förhindra att teknisk skuld uppstår (Li et al., 2015).
- Den sjunde metoden kallas för *representation/dokumentation,* och denna metod syftar till att representera och "kodifiera" teknisk skuld på ett enhetligt sätt som adresserar stakeholders bekymmer (Li et al., 2015).
- Den åttonde metoden kallas för *kommunikation,* denna metod synliggör teknisk skuld till stakeholders så att den kan diskuteras och hanteras ytterligare (Li et al., 2015).

Yli-Huumo et al. (2016) kommer i sin studie som handlar om TSH i utvecklingsteam fram till att vissa metoder i TSH var mer förekommande än andra, som redovisat i tabellen nedan. Författarna har utgått från de metoder som Li et al. (2015) definierat som TSH.

Tabell 4 Förekomsten av metoder för TSH (Yli-Huumo et al., 2016)

Mest förekommande metoder	Kommunikation
Frekvent förekommande metoder	Återbetalning Förebyggande Representation/Dokumentation Identifiering Prioritering
Sällan förekommande metoder	Mätning Övervakning

Alves, Mendes, Mendonça, Spínola, Shull & Seaman (2016) har i deras litteraturstudie tagit fram de hanteringsstrategier som blivit refererade till mest i TSH-studier.

Den första strategin är en *Cost-Benefit Analysis,* denna strategi innebär att man utvärderar om den potentiella räntan på skulden är hög nog för att rättfärdiga betalning av skulden (Alves et al. 2016). Räntan delas upp i två delar, sannolikheten av ränta och dess värde (Alves et al. 2016). Sannolikheten av ränta avser sannolikheten att ränta uppstår om skulden inte betalas vilket medför en större kostnad för projektet (Alves et al. 2016). Värdet avser en uppskattning av ytterligare arbete som kommer krävas om skulden inte betalas (Alves et al. 2016).

Den andra strategin kallas för *Portfolio Approach*, denna strategi listar upp olika typer av teknisk skuld, och i listan finns exempelvis information om var skulden finns, när den identifierades, personen som hittade den, varför den anses vara teknisk skuld, uppskattning om vad det kommer kosta att betala skulden, uppskattning om räntan och en uppskattning om korrelationen mellan denna skuld och andra skulder (Alves et al. 2016). När man planerar att uppdatera mjukvaran så analyseras denna lista och ett beslut tas om vilka skulder som skall betalas och vilka man kan skjuta på (Alves et al. 2016).

Den tredje strategin kallas för *Options,* och denna strategi innebär att betala av skulden i förväg genom att investera i olika alternativ som underlättar förändring av mjukvaran i framtiden, om mjukvaran måste ändras och då utan omedelbar vinst (Alves et al. 2016).

Den fjärde strategin kallas för *Analytic Hierarchy Process (AHP)*, denna strategi strukturerar ett problem och jämför det med alternativ med hänsyn till vissa kriterier (Alves et al. 2016). Detta leder till att man får en rankning av varje alternativ, och om vi applicerar detta på teknisk skuld så är dessa alternativen olika instanser av tekniska skulder och resultatet av denna strategi blir en rankning av dessa skulder som identifierar vilka som skall betalas av först (Alves et al. 2016).

Den femte strategin kallas *Calculation of TD Principal*, denna strategin fokuserar på den uppskattade kostnaden, där målet är att använda en definierad process för att uppskatta kostnaden för den tekniska skulden och att associera de identifierade problemen med olika kvalitetsstandarder ex. ISO 9126 (Alves et al. 2016). Enligt författarna leder detta till att man fattar bättre beslut gällande betalningen (Alves et al. 2016).

Den sista metoden Alves et al. (2016) fann är att forskare främst hänvisade till var *Marking of Dependencies and Code Issues*. Denna strategi som fokuserar på att hantera problem i källkoden, som görs genom att sätta in taggar i källkoden på ett sätt som gör det lätt för utvecklingsteamet att visualisera var skulden finns och därmed kunna bestämma när de vill betala den utifrån faktorer som hur mycket tid de har, samt hur mycket ansträngning som kommer krävas (Alves et al. 2016).

Metoderna för TSH som Alves et al. (2016) och Li et al. (2015) nämner i sin respektive forskning överensstämmer till stor del, med undantag för metoder för synliggörande av teknisk skuld för stakeholders och kommunikation som inte återges av Alves et al. (2016).

- *Cost-benefit analysis* och *portfolio approach* från Alves et al. (2016) passar in under Li et al. (2015) metod för *mätning*.
- *Options* från Alves et al. (2016) motsvarar Li et al. (2015) metod för *förebyggning*.
- *Analytic hierarchy process* och *calculation of TD principal* från Alves et al. (2016) passar in under Li et al. (2015) metod för *prioritering*.
- *Marking of dependencies & code issues* från Alves et al. (2016) inkluderar Li et al. (2015) metoder för *identifiering* och *återbetalning*.

2.4 Teoretiskt resultat

Nedan redovisas uppsatsens teoretiska resultat och hur dess olika delar är sammankopplade (se figur 2). Det teoretiska resultatet används för att besvara vår forskningsfråga som lyder:

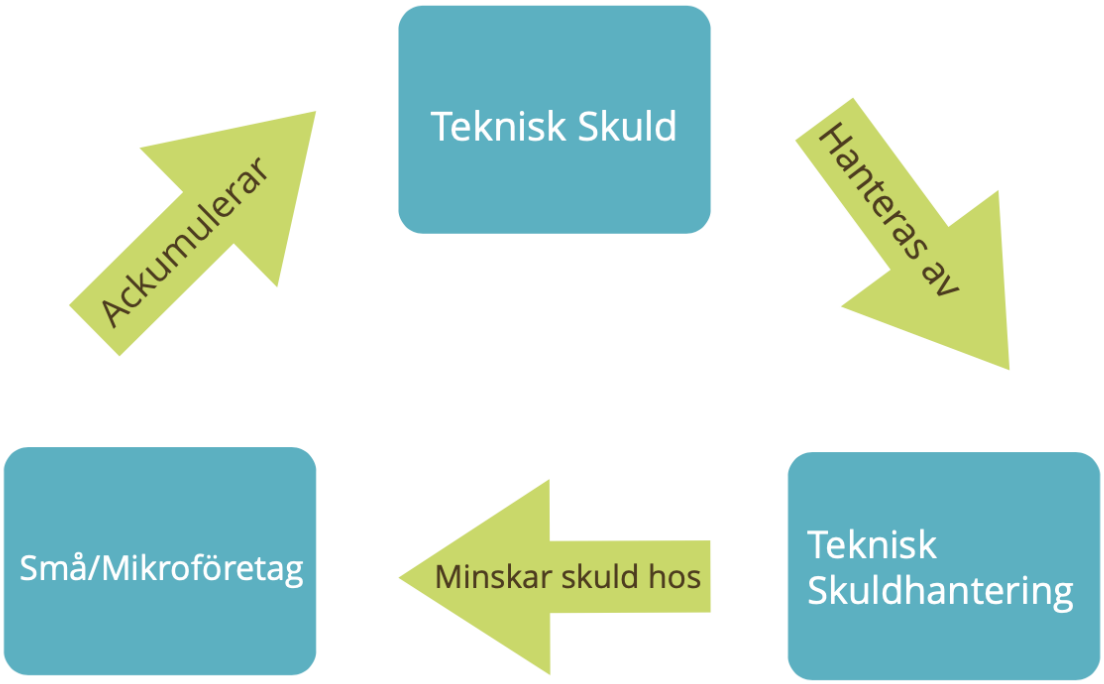
Hur hanterar små- och mikroföretag sin tekniska skuld?

Teknisk skuld och hantering av detta (TSH) är uppsatsens huvudämne, och läsaren behöver en förståelse för ämnet för att ta till sig uppsatsen. Teknisk skuld är en metafor som används för att beskriva en situation där genvägar tas i en teknisk beslutsfattning (Yli-Huumo et al., 2016). Detta görs som följd av prioriteringar inom organisationen, till exempel. vill företag ibland hellre lansera en funktionell produkt inom kort än en perfekt produkt när den är “slutförd”. Genvägarna i fråga resulterar i en suboptimal lösning som i framtiden kommer behöva åtgär-

das och liknas därför vid belåning eller *skuld* (Kruchten, 2012). Olika utövanden för att begränsa teknisk skuld faller under begreppet Teknisk skuldhantering som i denna uppsats förkortas TSH.

Teknisk skuldhantering (TSH) är ett samlingsnamn för ett antal olika metoder som används för att hantera teknisk skuld. I en litteraturstudie från 2015, bestående av data från 94 olika studier, tar Li et al. (2015) fram de åtta vanligaste metoderna för hantering av teknisk skuld. En senare studie av Yli-Huumo et al. (2016) rangordnade sedan de åtta vanligaste metoderna och till vilken grad de förekom, i sin studie ”*How do software development teams manage technical debt? – an empirical study”.* En sådan utvärdering är värdefull för företag som påverkas av teknisk skuld, och än mer så för små- och mikroföretag som kan sakna resurserna för en omfattande omstrukturering (Richardson & Wangeheim, 2007).

De företag som denna uppsats fokuserar på är små- och mikroföretag, vars utmaningar inom området teknisk skuld skiljer sig på ett antal sätt jämfört med mer etablerade bolag. Att producera högkvalitativa produkter tidseffektivt är en stor utmaning för små- och mikroföretag, vars resurser ofta är begränsade och mindre specialiserade (Ponsard & Deprez, 2017). De mindre företagen har inte heller möjlighet att utföra andra uppgifter än att utveckla produkten (Richardson & Wangeheim, 2007). Detta torde leda till bristande TSH.



Figur 2 Illustration av teoretiskt resultat

3 Metod

3.1 Metodval

Valet stod mellan en kvalitativ ansats eller en kvantitativ ansats i vår undersökning. Den kvalitativa metodansatsen valdes för denna studien, då forskningsfrågan vi ställt är av explorativ natur och en kvalitativ metodansats passar bäst för att besvara en sådan (Jacobsen, 2002). För att vi ska kunna undersöka och besvara vår forskningsfråga så krävs det att vi kan samla in nyanserade och djupgående data, vilket den kvalitativa ansatsen är bäst lämpad till (Jacobsen, 2002). Eftersom vi vill få in djupgående och nyanserad data så kan vi inte få så stor bredd på antalet respondenter då vi helt enkelt inte har tiden som krävs för det, därför passar det inte att göra studien med en kvantitativ ansats. Ytterligare en anledning till att vi valt att den kvalitativa ansatsen är att den gör det möjligt för oss att ställa följdfrågor under intervjuerna som kan ge oss nya insikter i ett komplext ämne (Jacobsen, 2002).

3.1.1 Urval

För att kunna genomföra vår studie valdes respondenter baserat på två huvudkriterier; att de arbetat eller varit delaktiga i mjukvaruutveckling under mer än 3 års tid, varav minst 1 år i små- och mikroföretag.

Respondenter identifierades genom ett bekvämlighetsurval (Jacobsen, 2002) där vi kontaktade personliga kontakter som föll inom ramen för våra kriterier. Respondenter hittades även via snöbollsmetoden (Jacobsen, 2002), med hjälp av personliga kontakter som rekommenderade lämpliga respondenter för vår uppsats.

Tabell 5 Översikt av respondenter

Namn	Erfarenhet	Erfarenhet i små- och mikroföretag	Datum	Typ	Plats	Appendix
R1	18 år	9 år	6 maj 2019 kl. 12.30	Face-to-face	Malmö	2
R2	18 år	8 år	6 maj 2019 kl. 13.00	Telefonintervju	Malmö/Stock-holm	3
R3	5 år	2 år	7 maj 2019 kl. 12.00	Face-to-face	Malmö	4
R4	11 år	8 år	8 maj 2019 kl. 10.00	Face-to-face	Lund	5
R5	4,5 år	1 år	9 maj 2019 kl. 13.00	Face-to-face	Malmö	6

3.1.2 Tillvägagångssätt

Vid valet av vilken typ av intervjuer vi vill utföra så valde vi att använda oss av semistrukturerade intervjuer. Detta gjorde vi då våra frågor handlade om samma ämne och vi ville ha möjligheten att kunna ändra ordningen på dessa frågor för att konversationen mellan oss och respondenten skulle bli så naturlig som möjlig och på så sätt få bättre kvalitet på svaren (Oates, 2006). Semistrukturerade intervjuer tillåter även oss att ställa nya frågor vi inte tänkt på tidigare, beroende på vad respondenten säger (Oates, 2006).

Intervjufrågorna skickades ut till respondenterna i förväg, då Oates (2006) menar att det är lämpligt att låta respondenten förbereda sig och kunna ge så bra svar som möjligt. Det hjälper oss även att öka vår trovärdighet som forskare.

Efter varje intervju hade vi en diskussion för att samla våra tankar och idéer samt utvärdera oss själva, men även frågorna och deras svar. Hade vi missat något och ville komplettera någon fråga i intervjun skickades ett mail till respondent och förtydligande gjordes. När data samlats in transkriberades den för att sedermera analyseras. När transkriberingen var utförd skickades den ut till respondenten för godkännande och eventuella mindre ändringar.

3.1.3 Utformning av intervjuguide

Inför intervjuerna togs en intervjuguide fram. Frågorna formulerades på ett öppet och icke ledande sätt för att respondenterna skulle känna sig bekväma och kunna på bästa sätt ge sin bild av hantering av teknisk skuld. Intervjun var upplagt på ett sätt som gör att den börjar med allmänna frågor och kallprat för att sedan introducera frågor generellt om teknisk skuld samt teknisk skuld på respondentens företag. Sedan övergick intervjun till frågor om teknisk skuld med hänsyn till företagets storlek (se appendix 1).

Tabell 6 Koppling intervjufrågor och teori

Kategori	Fråga	Koppling till teori
Allmänt om respondent	Vad heter du?	
	Vad är din roll på företaget?	
	Vad har du för bakgrund?	
Allmänna frågor om företaget och teknisk skuld	Är ert företag påverkat av teknisk skuld?	
	Hur har den tekniska skulden uppkommit?	Uppkomsten av teknisk skuld (2.2.1)
	Har företaget några utmaningar gällande teknisk skuld?	
Hantering av teknisk skuld	Är du medveten om vad teknisk skuld är?	Teknisk skuld (2.2)
	Vad är din definition av teknisk skuld?	Teknisk skuld (2.2)
	Har ni några medvetna strategier för att hantera den tekniska skulden? • Om ja: Vilka då? • Om ja: Kan ni beskriva de?	Teknisk skuldhantering (2.3)
	Hur kommer ni fram till att en skuld måste betalas?	Teknisk skuldhantering (2.3)
	Händer det att ni väljer att skjuta upp en betalning? • Vilka anledningar finns för att skjuta upp en betalning?	Teknisk skuldhantering (2.3)
Hantering av teknisk skuld avseende företagets storlek	Finns det några svårigheter med hantering av teknisk skuld med hänsyn till företagets storlek? • Finns det några fördelar?	Teoretiskt resultat (2.4)
	Om ni hade haft mer resurser hade ni gjort något annorlunda?	Teoretiskt resultat (2.4)

3.2 Etik

Det finns enligt Oates (2006) fem stycken rättigheter som deltagare har, dessa är rätten att inte delta, rätten att dra tillbaka samtycket, rätten att ge informerat samtycke, rätten till anonymitet, rätten till konfidentialitet. Dessa rättigheter kommer vi att förmedla på följande sätt:

Rätten att inte delta innebär att om någon inte vill delta i en undersökning, vare sig det är en individ eller en organisation, så behöver de inte (Oates, 2006). Om någon inte vill delta i undersökningen så kan det påverka forskarnas förmåga att slutföra forskningen men det är forskarnas problem och inte deras (Oates, 2006). Det händer ofta att denna rättigheten bortses från då det exempelvis kan finnas en maktskillnad mellan forskarna och personer de vill ha med i sin forskning (Oates, 2006). Det är viktigt att komma ihåg att forskarna inte har rätt att bestämma över vad någon ska göra (Oates, 2006). För att tillgodose denna rättigheten har vi inte lagt någon press på varken organisationer eller individer att de måste delta, vi har inte heller försökt övertyga dem genom att locka med belöningar eller försöka skuldbelägga någon genom att säga att vi måste ha deras medverkan för att fullborda forskningen.

Rätten att dra tillbaka samtycket innebär att om en respondent initialt sagt ja till att medverka i forskningen så har de rätt att när som helst ändra sig (Oates, 2006). Även om detta påverkar eller rentav omöjliggör forskningen är det forskarnas problem och inte respondentens (Oates, 2006). Denna rättigheten innebär även att en respondent har rätt att neka att svara på vissa frågor eller att inte vilja medverka på vissa delar av forskningen (Oates, 2006). Denna rättigheten har vi tillgodosett genom att inte skuldbelägga en respondent om den inte velat svara på en fråga eller delta i vissa delar av forskningen.

Rätten att ge informerat samtycke innebär att om en deltagare går med på att delta i forskningen så ges samtycket när de har informerats helt om vad forskningen och deras medverkan handlar om (Oates, 2006). Deltagarna ska informeras om syftet med forskningen, vem som utför forskningen, vad som kommer ingå i forskningen, om det kommer ingå kostnader för deras del, hur deras data kommer att användas, exempelvis om den blir anonymiserad och hur forskningen kommer spridas (Oates, 2006). Deltagarna ska även bli informerade att de har rätt att inte delta samt att de har rätt att närsomhelst dra tillbaka sitt samtycke (Oates, 2006). Denna informationen har vi förmedlat innan varje intervju har startat så att deltagaren är fullt medveten och kan ge sitt informerade samtycke

Rätten till anonymitet innebär att deltagare i forskningen har rätt att få sin identitet och plats skyddad (Oates, 2006). Om forskarna vill använda namn för deltagarna så ska påhittade namn användas (Oates, 2006). Forskare kan ibland även tvingas byta kön på deltagarna för att hemlighålla deras identitet (Oates, 2006). Organisationer har även rätt att hemlighålla sin identitet om forskarna inte får tillstånd att använda deras namn (Oates, 2006). Vi har tillgodosett denna rättighet genom att fråga deltagarna om de vill vara anonyma och vill de det så tar vi bort deras namn i rapporten. vi kommer även ge deltagarna och organisationerna möjlighet att granska forskningsrapporten innan den publiceras så att de kan vara säkra på att deras identitet inte röjs.

Rätten till konfidentialitet innebär att data som forskarna samlar in skall hållas konfidentiell. Det betyder att data exempelvis inte ska ligga på ett skrivbord där vem som helst kan få tillgång till den, data skall hållas säker (Oates, 2006). Detta kommer vi säkerställa genom att först och främst spara data lokalt och inte på molnbaserade lagringstjänster, men även i ett tidigt skede diskutera företagens förväntan på konfidentialitet i vår hantering av data som framställs under intervjuerna.

3.3 Validitet och reliabilitet

3.3.1 Reliabilitet

Med reliabilitet menas tillförlitlighet och det innebär att vi måste kunna veta om vi kan lita på den data som samlats in, detta är viktigt att ta reda på då undersökningens upplägg påverkar resultaten på olika sätt (Jacobsen, 2002). För att säkerställa tillförlitligheten har vi innan varje intervju pratat lite kallprat för att få respondenten att bli lugn och avslappnad och inte känna sig pressad eller nervös vilket kan leda till felaktiga data enligt undersökareffekten (Jacobsen, 2002). Vi har även hållit alla intervjuer utom en telefonintervju på respondentens arbetsplats, detta för att respondenten skulle känna sig så bekväm som möjligt och ge så korrekt information som möjligt då forskning visat att människor ändrar sitt beteende efter den omgivningen de befinner sig i (Jacobsen, 2002). Det sista steget vi tog för att öka reliabiliteten var att vi skickade ut frågor i förhand till respondenterna så att de kunde förbereda sig inför intervjun och ge så korrekta och nyanserade svar som möjligt.

3.3.2 Validitet

Med validitet menas giltighet som innebär att all data måste vara giltig, relevant och mäta det som ska mätas (Jacobsen, 2002). Jacobsen (2002) säger att inga kvalitativa undersökningar är bättre än de data som samlas in under de första faserna. Data kommer alltid från en källa vilket gör källkritik till ett viktigt verktyg för att säkerställa validiteten (Jacobsen, 2002). Vi har i vår forskning varit källkritiska och försökt använda oss av peer-reviewed material så mycket som möjligt samt källor som är välciterade.

Vi fastställde även ett par olika krav som våra respondenter var tvungna att tillgodose gällande deras arbetslivserfarenhet. Vi ville säkerställa intervjuernas validitet och såg därför till att endast intervjua personer med minst tre års arbetslivserfarenhet inom mjukvaruutveckling, varav minst ett av dessa åren skulle ha varit vid ett bolag som faller under våra kriterier. Respondenter som uppfyller dessa krav anser vi besitta den kunskap som vår frågeställning kräver för att besvaras.

3.4 Metodkritik

Genom att vi använt oss av ett snöbollsurval så är det möjligt att respondenterna inte har varit de som varit bäst lämpade att svara på våra frågor utan vi har intervjuat de för att de föll inom våra kriterier samt var tillgängliga. Hade vi letat runt mer så är det möjligt att vi fått kontakt med personer som varit ännu bättre på att svara på våra frågor. Ett annat problem vi stötte på var att vi inte gjorde någon övningsintervju som Oates (2006) rekommenderar, om vi gjort detta så hade vi kunnat skriva fler frågor att ta med oss till intervjuerna. De två första intervjuer var inte samtliga författare närvarande vid då intervjuerna skedde samtidigt, det finns en möjlighet att de intervjuerna hade blivit bättre om vi samlade från intervju 1 och alla hade möjlighet att ställa följdfrågor.

Intervju två utfördes över telefon, och avsaknaden av kroppsspråk kom att spela in i hur intervjun artade sig. Det blev den minst strukturerade intervjun i vår uppsats. Även om alla frågor

som förberetts blev ställda så blev svaren mer fragmenterade som ett resultat av svårigheter att kommunicera med varandra. Trots detta kom ett antal lärorika insikter vår väg tack vare intervjun, troligtvis då respondenten var väldigt kunnig inom ämnet.

4 Resultat

4.1 Strategier för hantering av teknisk skuld

Respondenterna frågades om det fanns några strategier som företaget använde sig av i sin hantering av den tekniska skulden. Det som kom fram efter intervjuerna var att hanteringen av teknisk skuld ofta skedde på magkänsla och erfarenhet snarare än uttalade strategier. Respondent 1 sade så här:

"[..]Jag tror att det finns många som medvetet jobbar bort den kontinuerligt, känner man att man har lite tid över "och att det inte finns något annat" så är det väldigt tillfredsställande att fixa någonting som borde fixas. Så det är inte så mycket uttalad strategi som magkänsla". (2:19).

Respondent 2 nämnde dock att det finns strategier som används men att dessa strategier fokuserar på hur den tekniska skulden ska hanteras långsiktigt och med ett fokus på lönsamheten snarare än uttalade strategier för hur den tekniska skulden skall hanteras och betalas av.

"[..]där måste man ha ett perspektiv som är vilken långsiktig lönsamhetspress har man. Hur viktigt är det att lösa den tekniska skulden, och över vilken tid? Ska produkten in på en specifik marknad, är den tekniska skulden relevant där eller på en annan marknad. Har den marknaden samma tekniska behovsbilden? Måste vi uppnå en viss vinstmarginal just nu eller kan vi vänta med det? Kan vi ta den tekniska skulden och finansiera den nu och vänta på lönsamheten? Det är såna där perspektiv som jag tror driver hantering av den tekniska skulden mest.". (3:36).

Respondent 4 nämnde en unik strategi som kom fram under intervjuerna som vi inte hört talas om tidigare:

"[..]vi har sagt att produktgruppen får bestämma hälften av vad utvecklarna ska ägna sig åt och resten är upp till det tekniska teamet själv, och av det arbetet som de bestämmer själva tar vi upp väldigt mycket av den tekniska skulden. Det är lite vår strategi, att dela upp tiden mellan produktteam och teknikteam.". (5:22).

Kontentan av intervjuerna var att små- och mikroföretag inte förlitar sig på de uttalade strategier som finns beskrivna i litteraturen för att lösa sin skuld utan de tar skulden när den kommer och försöker använda sig av sin erfarenhet och magkänsla för att se till så skulden inte blir för stor och ohållbar.

Respondenterna frågades även om hur de kom fram till att en skuld skulle betalas av samt hur de kom fram till att betalningen måste ske och att den inte kan skjutas upp till ett senare tillfälle. Vid frågan om när företagen kommer fram till att en skuld måste betalas så är de flesta eniga. Skulder betalas när man stöter på problem eller när smärtgränsen är uppnådd och arbetet och produktiviteten lider på grund av skulden.

"Erfarenhet, det känns väldigt givet, vi vet att man stöter på problem helt enkelt. Ju mindre skuld desto bättre överlag, men all skuld är inte negativ heller". (2:23).

"Det är väl ofta att det kommer en bugg, som kanske inte alltid beror på den tekniska skulden, men för att fixa buggen måste man kanske hantera skulden också. Då diskuterar man och kanske kommer fram till att man kan lika gärna göra något annat samtidigt. Det är nästan alltid så att det fixas i samband med att det kommer en bugg". (5:24).

Den som skiljer sig i sin åsikt är Respondent 2 som menar på att återbetalning av skulden är en ägarfråga vilket han menar är ett problem i flera bolag.

"Ja i ett litet bolag är det en ägarfråga egentligen. Den som äger mest har mest att säga till om. Så när alla delar ligger på bordet och det ska fattas beslut är det den som äger mest som bestämmer. Sen kan ju andra påläsa individer påverka men det är ändå den som äger mest rösträtt som har sista ordet. Och det är det som ibland blir lite oprofessionellt, eller inte riktigt balanserat. En sån person är ofta äldst och har minst teknisk insikt eller är minst digital. Så det finns för och nackdelar med att ha få huvuden som sitter på beslutet". (3:42).

F: "Har det varit ett problem som du har upplevt?". (3:43).

R2: "Ja, i flera bolag[..]". (3:44).

Vid frågan om vilka orsaker som finns för att skjuta upp en betalning var svaren mer spridda och varierade. Respondent 1 menar att skulden kan förändras över tid och att betalning av skulder som inte är akuta därför skjuts upp.

"En anledning att inte göra någonting direkt är för att skuldens natur kan ändras. Bara för att jag nu vet att det är en skuld, beroende på vad det är såklart. Just nu kan jag ha bilden av att fixa det på ett visst sätt men i framtiden på ett annat sätt... och om jag fixar det på det ena sättet först och sen på det andra sättet så har jag ett dubbelt arbete så av den anledningen så fixar man det när man väl vet att man måste fixa det. Och just att det kan vara olika anledningar, man kan veta att det är ett problem eller en skuld men om det inte är något som akut måste göras så kan man ju leva med det. Vi måste ju leva med skulden också, man kan inte vara helt skuldfri hela tiden, det känns orimligt". (2:31).

Respondent 3 menar att det främst är tiden man har tillgänglig som avgör vad man betalar och att de då väljer det som är mest kritiskt.

"[..]Vad är det vi hinner med och vad är det mest kritiska. Det får man jobba med[..]". (4:28).

Respondent 4 säger att skuldens påverkan på verksamheten är vad som avgör om skulden betalas eller ej.

"Att man tycker att den tekniska skulden inte har en för stor påverkan på verksamheten, eller produkten, eller kunderna alternativt att det är för mycket jobb att fixa den tekniska skulden jämfört med vad man vinner". (5:30).

Respondent 5 styrker vad respondent 4 säger men lägger även till att krav utifrån även spelar roll.

"Då tror jag att det finns två anledningar, antingen är det då att man tar beslutet aktivt att "okej vi lider inte av det här så mycket just nu så vi kan fortsätta ha det som det är". Men den andra grejen är också att när man har krav utifrån, det kanske är mer applicerbart om man är konsult, då måste ju kunden också gå med på att man spenderar den här tiden på att åtgärda en teknisk skuld vilket de inte alltid är så pigga på att göra. Då blir det på något sätt en, ja då får man de kraven på sig utifrån att man ska fortsätta att jobba med den här tekniska skulden". (6:26).

4.2 Teknisk skuld i små och mikroföretag

Respondenterna fick frågor som handlade om hantering av teknisk skuld med hänsyn till företagets storlek. Respondenterna nämnde främst att svårigheten med att hantera teknisk skuld när man är ett litet företag främst handlar om att man inte har tillräckligt med resurser och på så sätt inte kan få lika mycket gjort.

"[..]Jag tror att i mindre företag med färre utvecklare har man mindre resurser att lägga tid på sånt där och på att utreda. Då blir det lite mer skuld jämfört med när jag jobbat i större företag där man haft hundratals utvecklare och projektledare och liknande med analysmänniskor och allt möjligt. Då har man haft mer tid att behandla de här frågorna men då har utvecklingstiden å andra sidan lidit av det. Jag tror det är lite svårare i mindre företag". (5:32).

"[..]En svårighet att man faktiskt inte levererar. Man sa kanske att man skulle leverera och då finns det stakeholders och andra intressenter och själva imagen av företaget står ju på spel så det är väldigt nära businessen och extremt viktigt att trycka saker som folk ser värde i. Vilket gör att saker som utvecklare upplevelse som att det är lätt att slå upp en miljö eller att återställa en databas är inte lika prioriterade för att det är inte lika tydlig vinst för dem eftersom det är ingen som ser business caset i det[..]". (4:30).

Respondent 1 ansåg dock att det inte fanns några svårigheter med att vara ett litet företag utan det tvärtom.

"Nej, jag skulle nästan säga tvärtom. Ju större desto svårare verkar det bli. Eftersom att vi fortfarande är små är det ganska lätt. Man kan ganska snabbt samla alla utvecklare på ett ställe och prata och få alla med på banan med vad som ska göras och inte göras[..]". (2:33).

Respondent 5 håller med om detta påstående och säger att en fördel med att vara ett litet företag är att det är mycket mer flexibelt och rörligt.

"Det är mycket lättare att vara rörligt och flexibel kring hur man hanterar teknisk skuld när man är mindre. Det blir mindre att synka mellan programmerarna och alla som är inblandade och jobbar med koden, så definitivt. Ju större teamet är desto svårare blir det att jobba med det". (6:32).

En unik fördel med att vara ett litet företag tar respondent 4 upp, han menar att man som litet företag inte har samma press att inte göra något "dåligt" på samma sätt som i stora företag samt konsultuppdrag.

"Fördelen är att i större företag kan det ibland bli så att man måste fixa till det här, eller om man jobbar på konsultföretag och man gör något åt kunden vill man på något vis täcka sin rygg och inte åka på att det här gjorde ni dåligt. Då fixar man såna saker som man vet att man egentligen inte behövt göra, eller till exempel om man har tio saker som man måste göra en avvägning om man vill ta en teknisk skuld eller inte. Då kanske en av dem faktiskt har en impact men de andra nio har inte det. I vårt företag kanske man valt att skjuta upp alla tio och tänker att det blir en mindre kostnad sen medan ett större företag tänker att man måste fixa alla tio för att en sak kan få stor impact i deras fall". (5:34).

Vid frågan om företagen hade gjort något annorlunda om de haft mer resurser tillgängliga så höll alla med om att de inte hade gjort något annorlunda i hur de arbetade. Däremot hade de kunnat få mer gjort då de haft mer tid och mer människor som kunnat jobba med det.

"Jag tror inte det, jag tror vi hade jobbat likadant, vi är hyfsat nöjda med liksom betalnings takten även om det är klart att man drömmer om att ha all skuld betald och en fräsch ny kod-bas men i verkligheten kommer man ju ändå aldrig dit, allting ändras hela tiden". (2:41).

"Vi skulle utrett vissa frågor mer, nu diskuterar vi väldigt informellt. Det hade kanske varit mer fokus på att faktiskt utreda vissa implementeringar jämfört med andra. Sen skulle vi såklart ha haft mer tid att lägga på att betala av skulden". (5:36).

"Ja alltså. Mer resurser då kan man till exempel skaffa mer utvecklare eller testare, någon testare i vårt fall, vi jobbar ju lite som testare också. Resurser hjälper ju alltid[..] ". (4:34).

"Hade vi haft mer resurser så hade vi haft mer människor som jobbade på det och då hade vi nog behövt ha betydligt mer struktur kring hur vi jobbar med teknisk skuld. För det kan jag säga, nu när man är ett litet team så kan det bli ganska informellt. Att man bara beslutar sig för att förändringar snabbt, men ju större team man är och ju mer resurser man har ju mer behov finns det kring struktur". (6:38).

5 Diskussion

5.1 Strategier för hantering av teknisk skuld

Gällande vilka strategier som förekommer i små- och mikroföretag för att hantera sin tekniska skuld så fann vi att inga företag som frågades formellt använde sig av någon strategi som nämndes i teorin. Respondenterna sade att arbetet sker väldigt informellt. Då utvecklarna inte är så många vet dessa utvecklare om vad som kan bli ett problem senare, det finns därför inget behov att tydligt strukturera upp strategier.

Li et al. (2015) vidhåller åtta generella metoder för TSH som till viss del återfinns i de små- och mikroföretag vi varit i kontakt med, men som också skiljer sig från vad vi lärde oss av våra utförda intervjuer. Vi kommer nedan jämföra dessa mot vad vi fann i våra intervjuer.

Den första metoden, *identifiering*, som via aktiviteter som workshops ska uppenbara teknisk skuld hos företaget, återfinns inte hos något av de företag vi intervjuat. Detta har att göra med att företag av en viss storlek sällan har behovet, då all kod i företagets produkt ofta skrivits av en handfull utvecklare som alla är kvar på bolaget. I andra fall där man hunnit byta personal har det inte funnits en oöverskådlig kodbas att sätta sig in i och alla utvecklare tenderar att ha insikt i alla delar av koden och därför i var skulden finns. Respondent 4 nämner explicit att de inte använder sig av workshops för att identifiera teknisk skuld, som ett direkt resultat av att företaget inte har de resurser som krävts.

Den andra och tredje metoden, *mätning* och *prioritering*, som går ut på att väga kostnad mot nytta hos olika hanteringsåtgärder och sedan avgöra vilken som är mest kritisk att implementera. Denna typ av tänk präglar hur små- och mikroföretag hanterar teknisk skuld, även om inget av den strategiska och formella strukturen som beskrivs av Li et al. (2015) efterlevs hos våra respondenter. Våra resultat visar snarare att man i mindre företag uppdaterar varandra om teknisk skuld vid dagliga standup-möten, enligt svar från respondent 4, eller genom fortlöpande samarbete som eliminerar behovet av författningar då alla vid företaget hålls uppdaterade.

Återbetalning, som enligt Li et al. (2015) innebär Code Refactoring, är på vissa vis mest väsentlig av alla metoder. Detta beror på att det är genom refactoring som kodens kvalitet ökas och den tekniska skulden minskas i praktiken, och det används av alla företag för att förbättra strukturen hos kod som redan har den tilltänkta funktionaliteten.

Både *övervakning* och *förebyggning* är del av det dagliga arbetet, att döma av de intervjuer som utförts. Skulder övervakas på ett informellt vis av alla som hanterar koden och diskuteras utvecklare emellan, för att man som företag och utvecklarteam ska kunna förebygga framtida skuld. Som respondent 5 lägger stor vikt vid, spelar utvecklarnas erfarenhet in mycket i hur kompetent ett litet team kan vara i sin förebyggning:

”Men vad som kanske är ännu viktigare är hur man förebygger teknisk skuld. Att man kan se ungefär hur man vill strukturera upp koden.”. (6:20).

Representation/Dokumentation och *kommunikation* är de sista metoderna Li et al. (2015) nämner. Dessa metoder fokuserar på att synliggöra skulden för stakeholders. Respondenterna nämnde att det är viktigt att kunna förklara teknisk skuld och dess konsekvenser för stakeholders samt värdet som finns i att lösa den. En av respondenterna vi intervjuade tyckte det var viktigt att kunna synliggöra teknisk skuld för stakeholders men sade ej uttryckligen om de använde sig av några metoder för att göra detta. Då mängden resurser ett företag har tillgång till påverkar deras förmåga att betala skulden (Martini & Bosch, 2016) så är dessa strategier nödvändiga att använda sig av för att kommunicera behovet av att resurser läggs på att lösa skulden.

Istället för att använda sig av uttryckliga och forskningsbaserade hanteringsstrategier hade varje företag hade istället sina egna metoder för att lösa den tekniska skulden som inte finns beskrivna i litteraturen. Istället för att förlita sig på strategier så förlitar sig små- och mikroföretag istället på erfarenhet. Fyra av fem respondenter nämner att erfarenhet är väldigt viktigt för att hantera den tekniska skulden vilket inte är något som litteraturen på ämnet förespråkar som viktig faktor. Prudent and inadvertent debt är en typ av skuld som representerar vad teamet lärt sig när de jobbat på projekt och hur de hade kunnat göra det annorlunda för att nå en högre kvalitet (DevIQ, 2019). När teamet skaffar erfarenhet så är det lättare för dem att undvika att dra på sig den här typen av skuld.

5.2 Teknisk skuld i små- och mikroföretag

Respondenterna hade olika uppfattningar om hur företagets storlek påverkade deras förmåga att hantera den tekniska skulden, en del tyckte att det var lättare då det är enkelt att samla teamet med utvecklare och informera samt åtgärda och någon tyckte det var svårare då det finns mindre resurser att lägga på sådant.

De respondenter som hade uppfattningen av att det var enklare att hantera teknisk skuld i små- och mikroföretag, sade att anledningen till att detta var, i enighet med Richardson och Wangeheim (2007), att organisationsstrukturen i små- och mikroföretag var mer flexibel och de anställda hade större ansvarsområden och var således mer medvetna om vad de andra utvecklarna höll på med. Detta gjorde det enkelt att samla utvecklare och få alla med på banan för att lösa problem.

De respondenter som hade uppfattningen av att det var svårare att hantera den tekniska skulden i små- och mikroföretag motiverade detta med att de har mindre resurser samtidigt som det fanns förväntan av att leverera en produkt. Små- och mikroföretag har färre samt mindre specialiserade resurser (Ponsard & Deprez, 2017) detta i kombination med att man har krav utifrån på att leverera en bra produkt gör att dessa företagen inte har tid eller resurser att lägga ner på sekundära uppgifter (Richardson & Wangeheim, 2007), som teknisk skuldhantering är förens det faktiskt blir ett problem som påverkar verksamheten.

Respondenterna fick även frågan om de hade gjort något annorlunda om de hade fått mer resurser. Majoriteten svarade att de inte hade gjort något annorlunda i hur de arbetade, däremot hade de kunnat få mer gjort ifall de haft mer tid och arbetskraft som kunnat arbeta med det.

När respondent 5 fick samma fråga sade denna personen att om företaget hade blivit större så hade även organisationsstrukturen förändrats och arbetet hade krävt mer struktur. Det är organisationsstrukturen i små- och mikroföretag som tillåter de att arbeta informellt då strukturen är platt (Richardson & Wangeheim, 2007). Om strukturen förändras kommer således arbetssättet också förändras enligt Respondent 5 då det måste struktureras upp. Detta innebär att även om respondenterna inte avser att de ska förändra sitt arbetssätt så kommer de behöva det i olika stor utsträckning om de får mer resurser tillgängliga.

6 Slutsats

Syftet med den här uppsatsen var att förklara och beskriva hur små- och mikroföretag hanterar teknisk skuld. Därav blev vår forskningsfråga: *Hur hanterar små- och mikroföretag sin tekniska skuld?*

I vår studie kan vi konstatera att små- och mikroföretag till stor del präglas av ett informellt arbetssätt. Små- och mikroföretag använder sig inte utav uttryckliga metoder för att hantera sin tekniska skuld, men vi ser att delar av vissa strategier används men inte fullt ut. Detta beror på att organisationsstrukturen i små- och mikroföretag inte kräver att tydliga och skriftliga metoder används. Den platta organisationsstrukturen innebär att de anställda har en generell förståelse av vad som sker med utvecklingsarbetet. På grund av färre resurser och utvecklare så vet utvecklarna själva var den tekniska skulden kommer att uppstå och kan lätt kommunicera det till sina medarbetare. Om företaget däremot skulle växa, fler utvecklare skulle jobba med mjukvaran och organisationsstrukturen förändras så hade behovet för att strukturera upp arbetet med att hantera den tekniska skulden uppstått.

Små- och mikroföretag betalar av sin skuld när skulden blir ett problem för dem och inte i förväg innan problemet uppstår. Företagens anledningar till detta är att skulden kan förändras över tid och skapar det inget problem för stunden blir risken att de måste göra dubbelt så mycket arbete då skulden behöver betalas två gånger istället för en. En annan anledning som togs upp var att om skulden inte var ett problem och den betalades fanns risken att ytterligare skuld kunde uppstå och en snöbollseffekt uppstår. När resurserna är begränsade finns därför ingen anledning att göra detta extra arbete, resurserna behöver istället läggas på att utveckla en så bra produkt som möjligt och inte fin kod.

Den mest effektiva metoden för att göra detta på ett så bra sätt som möjligt är att skaffa erfarenhet. Fyra av fem respondenterna sa att erfarenhet var viktigt för att hantera den tekniska skulden. Erfarenhet är även en viktig faktor för att se till så att skuld inte uppstår från första början vilket gör att många resurser och arbetstimmar sparas in och kan läggas på att utveckla produkten.

Med detta sagt vill vi slutligen besvara forskningsfrågan som ställdes i början. Små- och mikroföretag hanterar sin tekniska skuld genom att lösa problem när de uppstår, de använder sig inte fullt ut av några strategier som finns i teorin utan mindre beståndsdelar av vissa strategier används istället. Detta görs då organisationsstrukturen tillåter att man arbetar mer informellt och mindre strukturerat vilket sparar på tid och resurser som är en bristvara i dessa typer av företag.

När vi reflekterar över uppsatsen tycker vi att vi fått ett godtagbart svar på forskningsfrågan. Vi är nöjda över hur strukturen av både uppsatsen samt vårt arbete har sett ut. Vi anser att vi baserat våra frågor på trovärdigt material samt att våra respondenter har varit kvalificerade att svara på dessa.

Appendix 1: Intervjuguide

Innan intervju

- Introducera oss.
- Berätta om ämnet för uppsatsen.
- Förklara målet med intervjun att vi vill undersöka hur TS hanteras i små- och mikroföretag.
- Förklara att respondent har rätt att vara anonym och be om informerat samtycke.
- Be om lov att spela in intervjun.
- Förklara att intervjun transkriberas och respondent ges möjlighet att läsa transkriptionen och rätta till eventuella missförstånd.
- Berätta att respondent ges möjlighet att ta del av den färdiga uppsatsen.

Respondentens bakgrund och roll på företaget

Vad heter du?
Vad är din roll på företaget?
Skulle du kunna berätta lite om din bakgrund?

Generellt om teknisk skuld och teknisk skuld på respondentens företag

Är du medveten om vad teknisk skuld är?
Är ert företag påverkat av teknisk skuld?

- Om ja: På vilket sätt?

Hur har den tekniska skulden uppkommit?
Har företaget några utmaningar gällande teknisk skuld?
Har ni några medvetna strategier för att hantera den tekniska skulden?

- Om ja: Vilka då?
- Om ja: Kan ni beskriva de?

Hur kommer ni fram till att en skuld måste betalas?
Händer det att ni väljer att skjuta upp en betalning?

- Vilka anledningar finns för att skjuta upp betalningen?

Teknisk skuld med hänsyn till företagets storlek

Finns det några svårigheter med hantering av teknisk skuld med hänsyn till företagets storlek?

- Finns det några fördelar?

Om ni hade haft mer resurser tillgängliga hade ni gjort något annorlunda?
Finns det något mer ni vill tillägga?

Appendix 2: Intervju 1

Tid: 6 maj 2019 kl. 12.30

Plats: Respondentens arbetsplats

Deltagande: Christopher Andersson, Martin Larsson

Respondent: Anonym

Författare/Forskare = F
Respondent = R1

Nummer	Person	Fråga/svar
1	F	Vad heter du?
2	R1	*anonym*
4	F	Vad är din roll på företaget?
5	R1	Head of products och planera produkten, features, vad som ska göras, när det ska göras, vem som ska göra det så det är ganska mycket.
5	F	Vad har du för bakgrund?
6	R1	Datavetare och informatik, har jobbat som utvecklare på google och gapminder.
7	F	Är du medveten om vad teknisk skuld är?
8	R1	Yes, mycket medveten.
9	F	Är ert företag påverkat av teknisk skuld?
10	R1	Definitivt, man får alltid ta med det i beräkningen på alla nya saker som det påverkar
11	F	Finns det något specifikt sätt ni är påverkade av teknisk skuld?
12	R1	Ja det blir ju ofta, ofta går ju saker långsammare att göra. Ju större skuld man har beroende på vad det är så går det ju långsammare att göra, det kommer ofta och biter en vid fel tillfällen.
12	F	Hur uppstår då den här tekniska skulden ni har?
13	R1	Ofta är det för att vi ändrar oss och kommer på något bättre och byter riktning, man gör på ett sätt och sen ska man göra det på ett annat sätt. Så innan man helt har gått över till det nya sättet eller rensat bort alla negativa konsekvenser med det gamla sättet, eller bara över tid, förutsättningar förändras och det som var en tillgång blir en skuld.
14	F	Har företaget några utmaningar gällande teknisk skuld?
15	R1	Alltså utmaningar är ju att prioritera det rätt för att vi skulle kunna sitta i ett år och bara betala av och inte göra något annat i princip tror jag men det skulle ju vara kontraproduktivt.
16	F	Har ni något speciellt sätt som ni prioriterar?
17	R1	Ja det brukar vara när vi stöter på problem helt enkelt, där det märks för vi vet att vi har en massa som inte märks också och då spelar det ingen roll i nuläget men när det kommer dit så fixar vi det också så det är väldigt reaktivt.

18	F	Har ni några medvetna strategier för att hantera den tekniska skulden?
19	R1	Ja fast jag tror att detta också är ett personligt drag liksom om man lider av att man har den bara för att man vet att den finns eller inte så att jag tror att det finns många som medvetet jobbar bort den kontinuerligt, känner man att man har lite tid över "och att det inte finns något annat" så är det väldigt tillfredsställande att fixa någonting som borde fixas. Så det är inte så mycket uttalad strategi som magkänsla.
20	F	Men lite såhär om ni ska gå tillbaka och kolla på någon gammal kod och inte bara, oj det ser ut så här. Det är lite den typen av.
21	R1	Ja jag förstår vad du menar. I vissa fall men för det mesta säger den här erfarenheten att det som är gammalt kan ju fungera och bara för att man uppdaterar det till det nya sättet att göra det så är det ju inte säkert att det fungerar bättre. Så vi ändrar så lite som möjligt i processen.
22	F	Hur kommer ni fram till att en skuld måsta betalas?
23	R1	Erfarenhet, det känns väldigt givet, vi vet att man stöter på problem helt enkelt. Ju mindre skuld desto bättre överlag, men all skuld är inte negativ heller.
24	F	Har ni några analysverktyg för att se det här?
25	R1	Nej
26	F	Utan det är problem som uppstår så fixar man det då eller?
27	R1	Ja det är nog mycket också medvetet om vi vet historiskt vad vi gjort, det sitter i bakhuvudet och gnager lite som ett dåligt samvete för något som borde fixas men inga verktyg.
28	F	Händer det att ni väljer att skjuta upp en betalning?
29	R1	Ja hela tiden, vi har ju egentligen man kan säga att det är ett berg framför oss hela tiden i någon mening och många små saker som inte spelar någon roll just nu är medvetet uppskjutna. Så vi tar uti med saken när det blir problem, det är ganska reaktivt som sagt.
30	F	Vilka anledningar finns det att skjuta upp betalningen?
31	R1	En anledning att inte göra någonting direkt är för att skuldens natur kan ändras. Bara för att jag nu vet att det är en skuld, beroende på vad det är såklart. Just nu kan jag ha bilden av att fixa det på ett visst sätt men i framtiden på ett annat sätt… och om jag fixar det på det ena sättet först och sen på det andra sättet så har jag ett dubbelt arbete så av den anledningen så fixar man det när man väl vet att man måste fixa det. Och just att det kan vara olika anledningar, man kan veta att det är ett problem eller en skuld men om det inte är något som akut måste göras så kan man ju leva med det. Vi måste ju leva med skulden också, man kan inte vara helt skuldfri hela tiden, det känns orimligt.
32	F	Finns det några svårigheter med hantering av teknisk skuld med hänsyn till företagets storlek?
33	R1	Nej, jag skulle nästan säga att tvärtom. Ju större desto svårare verkar det bli. Eftersom att vi fortfarande är små är det ganska lätt. Man kan ganska snabbt samla alla utvecklare på ett ställe och prata och få alla med på banan med vad som ska göras och inte göras. Så att.. Men och man märker snabbt också i och med att det finns teknisk skuld i

		koden, klart kommer det in en ny utvecklare så kommer den stöta på de gamla sakerna. Det gamla sättet att göra det så uppstår förvirring och det är en av anledningarna till att man vill bli av med det också.
34	F	För att det finns en mindre personbank att fråga om det är 6 till 10 utvecklare som har skapat koden så är det lättare istället för..?
35	R1	Ja precis det är lättare att hitta den som är ansvarig.
36	F	Tror du det finns några fördelar?
37	R1	Med?
38	F	Företagets storlek
39	R1	Ja precis, ju mindre desto lättare bli det att hantera. Så att det tror jag. Ju färre inblandade man är desto lättare är det att ha koll på det. Nu är det fortfarande så att vi som har skrivit koden fortfarande är kvar, nästan alla och då vet man också var problemen är och var lösningen ofta sitter. Till skillnad om man tagit över en kodbas från någon annan som man inte förstår sig på fullständigt.
40	F	Om ni haft mer resurser tillgängliga hade ni gjort något annorlunda då?
41	R1	Jag tror inte det, jag tror vi hade jobbat likadant, vi är hyfsat nöjda med liksom betalnings takten även om det är klart att man drömmer om att ha all skuld betald och en fräsch ny kodbas men i verkligheten kommer man ju ändå aldrig dit, allting ändras hela tiden.
42	F	Finns det något mer du vill tillägga?
43	R1	Det är ett intressant ämne, jag skulle gärna veta mer om det egentligen om det fanns uttalade strategier för allt jag har är egentligen bara magkänsla som jag utvecklat av erfarenhet. Det är inget jag har studerat eller läst om eller tagit mig tid att sätta in mig i. Men det har väl också med projektets storlek och natur. Jobbar man ännu mer abstrakt med projekt är det såklart en annan faktor man får räkna in men det är intressant. Vi är ett ganska ungt företag med ganska ung kodbas och vår tekniska skuld är väl mindre eftersom systemen är färska men å andra sidan går det ganska snabbt när man ändrar riktning och utvecklingsmetodik och det går ganska snabbt att bygga på saker men det är inte ett gigantiskt COBOL system med 100 miljoner rader som är vår skuld, det kan man tänka sig på en bank.
44	F	Hur definierar du teknisk skuld? (kompletterat via mail)
45	R1	Ett lån av tid från sig själv. Göra man en ändring och tar genvägar så blir det snabbare gjort, men man har en skuld. Gör man det ordentligt från början tar det längre tid, men med utan skuld. Skulden kan betalas genom att lägga utvecklingstiden på att ta bort det dåliga som uppstod genom genvägen. (kompletterat via mail)

Appendix 3: Intervju 2

Tid: 6 maj 2019 kl. 13.00

Plats: Telefonintervju

Deltagande: Felix Lidforsen

Respondent: Anonym

Författare/Forskare = F
Respondent = R2

Nummer	Person	Fråga/svar
1	F	Då börjar vi. Vad är ditt namn, *anonym*?
2	R2	Korrekt.
3	F	Vi kommer i denna intervjun fokusera på dina tidigare erfarenheter av arbete på ett startup-bolag, inte där du arbetar i nuläget.
4	R2	Ja det stämmer.
5	F	Kan du berätta vad du gjorde på ditt tidigare bolag?
6	R2	Jag var en del av ett grundarteam som byggde en software service-plattform, en ren startup. Min roll där var att vara affärsutvecklings-ansvarig och titta långt fram och strategiskt för tjänsten.
7	F	Okej, så du var del av uppstartarteamet. Var du alltså en del av själva visionen och idéen av produkten också?
8	R2	Nej, jag kom in strax efter. De var ett tekniskt team som saknade ent-reprenörskap och affärskompetens och då tog dem in mig. Jag var alltså inte del av behovsidentifieringen eller kodningen utan jag kom in i andra varvet.
9	F	Okej, så sådan är din bakgrund. Du har arbetat vid ett startup-bolag som business-mind kan man säga. Under din tid där, tampades ni med teknisk skuld? Vet du vad teknisk skuld är?
10	R2	Ja
11	F	Det företag du satt på tidigare, hur påverkades det företaget av tek-nisk skuld skulle du säga?
12	R2	En hel del, och den var ökande och krympande i omgångar. Över åren så var den alltid hanterad av att finansieras av nya beställningar. Så i ett mindre startup eller där kundbasen är hanterbar där funkar det att jobba med nybeställningar och driva dessa för att passa den tek-niska skulden.
13	F	Okej. Intressant det du säger att den tekniska skulden betalas med in-täkter från nya beställningar.

14	R2	Ja, och hanterbarheten ligger i att samma personer som jobbar med det strategiska säljet har insikt i den tekniska skulden. Men när storleken på bolaget eller grupperna blir för åtskilda då faller den möjligheten, är väl en kort summering.
15	F	Var det något du hann bevittna innan du slutade på bolaget?
16	R2	Verkligen, det brottades vi om i en åtta-tioårsperiod. Man hade teknisk skuld från dag ett kan man säga. Bolagen är ofta tvingade att gå till kommersialisering eller försäljning innan plattformarna är klara, så teknisk skuld är ett standardläge.
17	F	Så du arbetade där i 8-10 år?
18	R2	Ja ungefär så.
19	F	Då skulle jag vilja fråga såhär; när ni satt på företaget och diskuterade hanteringen av teknisk skuld, pratade ni om hur den tekniska skulden uppkommit? Var det något som vi visste?
20	R2	Nej, det gjorde vi nog inte, att vi försökte problematisera eller skuldbelägga. Det var underförstått att den tekniska utvecklingen av olika plattformsintegrationer eller kundkrav, standarder och sånt går så fort. Detta var den stora drivkraften för den tekniska skulden då. Sen kunde man dock ångra vissa strategiska val.
21	F	Okej, som vilka då?
22	R2	Jag kommer ihåg en gång, då var det databasplattformsvalet.
23	F	Så ni gjorde ett val som senare visade sig vara opassande?
24	R2	Ja det var för snålt tilltaget, inte tillräckligt kraftfull databasplattform som valdes. Vi blev tvungna att skifta mitt i. Det gick dock bra. Sen var det väl andra saker, som vilken kodbas man väljer generellt kan påverka väldigt många strategiska val under tiden, tidigt, och då sitter man kanske i ett vägvalskonsekvent längre fram.
25	F	Okej, så man vet inte vad man kommer behöva i framtiden.
26	R2	Exakt, och jag tror att, återigen då hanterbart att marknad och utvecklingsteam och strateger är väldigt tajta i början och kanske har örat mycket mot vad som är populära plattformsval vid startuptillfället medan senare när bolaget är etablerat och vi är mitt iförsäljnings/kommersialiseringsfasen då är det svårare att hålla den aktualiserad.
27	F	Okej. Små företag som hanterar teknisk skuld, har de några speciella utmaningar? Som är specifika för just dem?
28	R2	Ja. Såhär är det, att den mest ansvariga till den tekniska skulden är också den med mest teknisk cred.
29	F	Ursäkta, vad sa du?
30	R2	Att den som har störst ansvar för den tekniska skulden och de tidigare plattformsvalen eller strategiska valen är också samma person som har mest ansvar och saker att säga när det kommer till att hantera den tekniska skulden. Så antalet huvuden i ett startup är det som är dilemmat egentligen. Och ekonomiska resurser och så vidare.
31	F	Menar du på gott och ont då eller?

32	R2	Ja, det ligger inbyggt. En av anledningarna till att jag valdes in i den där startupen var att jag hade en bucket-list på saker man ska tänka på i förväg.
33	F	Okej, som du tillförde företaget direkt?
34	R2	Exakt, som jag tycker det är viktigt att man ha samsyn på en femårsnivå eller installamentsfrågor som dyker upp efter ett halvår redan. Man måste ha en femårsplan tillsammans. Vad är en exit sen efter, men på en detaljnivå saknar vi till exempel hur ska vi hantera när hela plattformen blir ifrågasatt. Såna frågor hade vi inte pratat om, utan det blev ofta ganska kladdigt.
35	F	Okej jag förstår. Men du säger att när ni pratar om att hantera den tekniska skulden, hade ni medvetna uttalade strategier för detta?
36	R2	Ja, och där måste man ha ett perspektiv som är vilken långsiktig lönsamhetspress har man. Hur viktigt är det att lösa den tekniska skulden, och över vilken tid? Ska produkten in på en specifik marknad, är den tekniska skulden relevant där eller på en annan marknad. Har den marknaden samma tekniska behovsbilden? Måste vi uppnå en viss vinstmarginal just nu eller kan vi vänta med det? Kan vi ta den tekniska skulden och finansiera den nu och vänta på lönsamheten? Det är såna där perspektiv som jag tror driver hantering av den tekniska skulden mest.
37	F	Jag förstår, så det är en fråga om prioriteringar?
38	R2	Ja, eller om det ens går att lösa. Finns det ens resurser att lösa problemet? Att komma överens om att den tekniska skulden är där och att man skulle vilja göra saker, det kan alla ganska snabbt vara överens om. Man sammanfattar det i tidsrymd, i vilken tid är den viktig?
39	F	Okej, alla är överens om problemen med teknisk skuld, man bearbetar de mest akuta problemen först?
40	R2	Ja.
41	F	Okej. Hur är det man kommer fram till att en skuld måste betalas?
42	R2	Ja i ett litet bolag är det en ägarfråga egentligen. Den som äger mest har mest att säga till om. Så när alla delar ligger på bordet och det ska fattar beslut är det den som äger mest som bestämmer. Sen kan ju andra pålästa individer påverka men det är ändå den som äger mest rösträtt som har sista ordet. Och det är det som ibland blir lite oprofessionellt, eller inte riktigt balanserat. En sån person är ofta äldst och har minst teknisk insikt eller är minst digital. Så det finns för och nackdelar med att ha få huvuden som sitter på beslutet.
43	F	Har det varit ett problem som du har upplevt?
44	R2	Ja, i flera bolag. Jag tycker man ser det även i närbevakligt affärsliv. Så att det är samma dilemman då. Sen blir det också ett drev, ett tekniskt drev, om vad som är en uppdaterad plattform eller inte. Det kan säsongsvariera, beroende på vilka stora riskkapitala affärer som görs just den perioden. Om det görs en jättestor kapitalsatsning i en open-source som har global mediebevakning får den ganska stort inflytande i tekniska val.
45	F	Men kanske inte över jättelång tid menar du?

46	R2	Nej, det kanske är helt irrelevant två månader senare. Så den tekniska skulden har ett inbyggt dilemma i just små bolag där det är liten input.
47	F	Okej. Jag vill fråga såhär med; om ni väljer att skjuta upp en betalning, vad finns det för anledningar till det?
48	R2	Betalning vet jag inte vad du menar, jag förstår inte riktigt frågan.
49	F	När jag säger betalning av en teknisk skuld menar jag hantering eller återbetalning av teknisk skuld.
50	R2	Jag ser det mer som strategiska val, det går inte att svara på det. Inte i detalj i alla fall, har man valt fel databas så måste man någon gång ändra på den. Har man den insikten måste den ändras, annars kommer inte bolaget överleva. Då får ingen betalt. Om tjänsten fallerar för att man valt fel teknik måste du ändra den för att överleva. Men det där är ofta inte samma personer, och därmed är frågan i min värld lite felställd. Det är ofta inte samma individer som sitter på betalningsmandatet som hanterar den tekniska skulden? Förstår du hur jag menar?
51	F	Som faktiskt gör det menar du? De som konkret ser efter så att åtgärderna utförs?
52	R2	Nej, de där besluten fattas av en ägare.
53	F	Okej, det verkar vara ett perspektiv som vi inte riktigt fångat i våra frågor, så som du nämner kommunikationen mellan ledning och utvecklare.
54	R2	Ja, den yngsta medarbetaren är sällan ledare eller VD eller ledning, men kan ha mest insikt i den tekniska delen. Ofta har en yngre medarbetare sitt mandat i att den är passionerad tekniskt, snarare än att den är väldigt driven av styrelsefrågor. Det är det som är storheten med om bolag lyckas, tekniska bolag, att man hittat en bra balans där. Den tekniska skulden är konstant, den finns alltid där och det är en stor del av ett strategiskt arbete. Det var det jag tyckte var triggande i att svara på denna enkäten, att det var en intressant frågeställning. Den upptar så stor del av en startup.
55	F	Ja man har börjat förstå det nu efter all läsning och av att höra dig prata nu. Jag läste en undersökning som säger att många utvecklare spenderar nära halva arbetsveckor med teknisk skuld.
56	R2	Ja, och jag tycker att den ska vara konstant också. Ligger man helt rätt har man sannolikt lagt för lite tid på marknad och sälj. Det är när man träffar kunder och partners och återförsäljare det är då man får sin tekniska skuld uppdaterad. Det är de som sitter på svaren, vad som är rätt och fel. Inte det teoretiska kodarteamet. Tänk på Ericsson.
57	F	Hur då menar du?
58	R2	Ja de var ett klassiskt ingenjörsdrivet bolag som inte hade kontakt med sin marknad riktigt, lite på 90-talet hade de konkurs.
59	F	Okej, det visste jag inte om.
60	R2	Och det har väl skett i omgångar i deras 100-åriga historia. Så det är väl intressant hur den tekniska skulden hela tiden finns där, fastän de trodde att de var uppdaterade.
61	F	Okej så de trodde att de var i fas. Hur kunde de missbedöma det tror du, om du vill svara på det?

62	R2	Ja, sannolikt var det så att ingenjörerna hade för mycket att säga till om, jämfört med de som skulle sälja produkten på marknaden. Det fanns inte en kommersiell balans mellan teknikerna och marknadsteamet.
63	F	Så produkterna var optimerade efter ingenjörer och inte konsumenter?
64	R2	Exakt. Jag tycker också det kan vara en intressant infallsvinkel att ta med i er rapport. Vad är teknisk skuld? Vem definierar den? Teknisk skuld för vem? För jag menar, för ingenjörerna i det exemplet uppfattade inte att det fanns en teknisk skuld, eller hur? De tyckte att det hade jättemycket mandat.
65	F	Okej, de förbisåg det och byggde på ny funktionalitet?
66	R2	Ja, medans marknad då inte hade något att säga till om eller inte blev lyssnade på, i deras värld var det ju uppenbart att den tekniska skulden var ofantlig.
67	F	Och det var den tekniska skulden som sedan påförde konkursen eller?
68	R2	R: Nej men det var ett hot, de klarade sig precis. De fick göra nyemission på en massa pengar och hade en stor ägarstädning, så det höll ju på att gå illa. Ett annat bra exempel att titta på om ni har möjlighet är ju Microsoft eller Apple. Microsoft som ju i förra veckan gick upp till världshistoriens mest värderade bolag. Då kan man ju vara helt fascinerad över att bolag som grundades på 70-talet har överlevt den här digitala transformationen. De har uppenbarligen haft en bra balans eller definition och hantering av sin tekniska skuld kan jag tycka. De har ju mycket insyn. Sen har vi Apple då som, jag vet inte vilket årtionde de går i konkurs nästan. Efter sina iMacs och Macintosh-datorer.
69	F	Menar du hur de dippar just nu eller har de dippat tidigare?
70	R2	De har dippat kraftigt tidigare då, med succén med Macintosh blev de övermodiga och trodde att den typen av teknisk skulle räcka. Man fick ta nödlån och allt möjligt. Så det Apple vi känner till idag, så var det inte fram till slutet på 90-talet.
71	F	Hur tänker du då? I vilken bemärkelse?
72	R2	Då hade man den här självgode, tekniska avdelningen som inte tittade så mycket mot marknad. Det var först när Steve Jobs fick sitt mandat som man fick en teknisk avdelning som gjorde saker som marknaden ville ha. Kolla på årtalen när ni använder det, det står i alla böcker.
73	F	Det ska jag göra, tack för tipset. Jag har en fråga jag vill ställa, men jag tycker du har besvarat våra frågor väldigt väl. För att anknyta till det du sa tidigare, du anser alltså att den platta företagsstrukturen hos ett litet företag tillåter att man får mer input av varandra.
74	R2	Jag är inte ute efter någon demokrati där riktigt, för alla sitter ju med olika erfarenheter att hantera en sån här fråga som teknisk skuld. Den som har mest positivt inflytande för att hantera det ska givetvis ha det. Det korrelerar ju inte alltid med antal huvuden och gruppbemötanden eller ägarskapsdelning. Här är det då huvudägarens uppgift att verkligen låta alla komma till tals och utvärdera. Vi hade även ett helt externt, oberoende advisory board, eftersom vi var medvetna om att vi bara var fem delägare som dessutom alla var relativt unga. Vi kan

		inte ha en teknisk kompetens för att hantera någonting som ska hålla i tio-femton år.
75	F	Ni utgick från det från början alltså?
76	R2	Ett annat sätt att hantera det kanske är att ha externa medlemmar i styrelsen, i just de erfarenheterna.
77	F	Men tenderar man att ha en styrelse på en gång i ett startup?
78	R2	Ja, det tror jag man ska ha. Åtminstone en fiktiv styrelse, en advisory board eller gäng kompisar eller nätverk.
79	F	Okej, i syftet att få perspektiv utifrån bolaget tänker du?
80	R2	Ja syftet är ju att vara lönsamma. Syftet är att bli kommersiellt gångbar, annars ska man inte starta bolag. Och ska man komma dit är 95 % av allting själva genomförandet. Idén och plattformen är bara si eller så mycket, hur vi hanterar alla de frågorna är det som betyder något.
81	F	Jag upplever ditt synsätt som väldigt hands-on, det märks att du gjort detta ett antal gånger i olika former.
82	R2	Ja, men vilken skola tillhör du?
83	F	Jag läser vid Lunds universitet.
84	R2	Ja jag flyttade till UC, University of California när jag var tjugo. Jag satt på min första lektion i stora aulan och läste business 101. Så ställde professorn den första frågan till de 800 i åhörarskaran, ”Why do you start a business?”. Jag och min svenska kompis, vi var två år äldre än alla andra då vi hade gjort lumpen innan första universitetsdagen. Vi hade långa utredningar om samhällsansvar och självförverkligande för medarbetarna och såna saker, och längst bak i klassrummet satt en nockfull, kepsprydd, finnig, artonårig amerikan som sa ”To make a profit”. Det tycker jag visar hur USA jämfört med resten av världen ser på affärsmannaskap.
85	F	Ja för jag tänker nu när vi pratat att du resonerar lite amerikanskt, i brist på bättre benämning.
86	R2	Ja, den här första veckan på ett amerikanskt universitet påverkade mig väldigt mycket efter att ha haft socialdemokratiska utdelningar vi får här i Sverige med alla referenser till samhällsansvar och det finns bara en anledning att ett företag ska finnas, att skapa värde. Det ska finnas i er rapport där om hur teknisk skuld hanteras, det är ganska oväsentligt att titta på hur ramverket hanterar sin tekniska skuld. Den vet vi hur den är hanterad, den ligger 20–30 år efter. Det finns inte kapacitet tekniskt att möta samhällets behov 2019. Och hade ramverket haft ett kommersiellt ansvar hade den tekniska skulden varit uppdaterad.
87	F	Så du menar att om man visste exakt hur man gjorde hade alla gjort det liksom?
88	R2	Nej men jag tror att kundernas kravställan mot ramverket kommer när tåget inte går. Och hade en ägare fått ta konsekvenserna av att inte kunna leverera tåg i tid hade man hanterat den tekniska skulden innan. Annars hade bolaget gått i konkurs, eller hur? Inget tekniskt bolag som ni tittar på med den här frågeställningen vill ju ha en teknisk skuld där tjänsten fallerar.

89	F	Och du menar att om man då jobbar med tågtrafik finns det andra incitament för att upprätthålla en stabil verksamhet?
90	R2	Just banverket är bara ett exempel men kanske ägarskapet. Missionen för bolagsägandet, vad är syftet med banverket? Jo att tåg ska komma fram i tid. Levererar man inte den tjänsten kommer det bolaget upphöra att finnas, medans en statlig ägare struntar i det, uppenbarligen. SAS är likadant, kostar fyra miljarder i kvartalet för svenska medborgare, och kan ändå inte leverera i tid. Jag tycker den här tekniska skulden, du kan hitta på en triangel med ägarskap, profit och teknisk skuld. Det går inte att hantera teknisk skuld, det finns ingen som har incitamentet till att hantera den om det inte finns ett lönsamhetskrav längre fram.
91	F	Tack För att du medverkat. Har du något du vill tillägga?
92	R2	En annan balans var det den med antalet huvuden, man kan ha jättemycket ambitioner men har man för få inputs blir det inte balans i alla fall. Jag tar jättegärna del av ert resultat och höra vad andra har sagt.

Appendix 4: Intervju 3

Tid: 7 maj 2019 kl. 12.00

Plats: Respondentens arbetsplats

Deltagande: Christopher Andersson, Martin Larsson & Felix Lidforsen

Respondent: Anonym

Författare/Forskare = F
Respondent = R3

Nummer	Person	Fråga/svar
1	F	Vad heter du?
2	R3	*anonym*
3	F	Vad gör du här på företaget?
4	R3	Utvecklare så programmerare. Jag är väl mer back-end skulle jag säga, i och med att det är ett litet företag så då gör man mycket så jag är väl även front-end och lite infrastruktur.
5	F	Så det är mest generella roller?
6	R3	Ja. Nu har vi vuxit mer så nu har vi mer dedikerad front-end och UX. Men jag var första anställda så jag fick göra mycket av allt i början och gör fortfarande det, men mer back-end.
7	F	Vad har du för bakgrund?
8	R3	Systemvetare och jobbat tidigare på Sigma och IBM och Jayway. Tidigare var jag musiker egentligen. Systemvetenskap som ofta är relevant.
9	F	Är du medveten om vad teknisk skuld är?
10	R3	Ja, jag har en aning om vad jag själv skulle definiera det som.
11	F	Ja, jag har en aning om vad jag själv skulle definiera det som.
12	R3	Det är någonting som man har, specifikt kod. Det kan ju antingen vara infrastruktur eller kod. Men någonting som är implementerat och körs i produktion skulle jag säga. Som man förr eller senare kommer lida av. Egentligen har man ett use-case som kanske byter scope eller att man medvetet eller omedvetet missar cases eller edge cases i sitt scope. Som gör att ens lösning inte håller. Lösningen kan hålla till mindre eller större grad som typ att hårdkoda konstant är så tydligt teknisk skuld, det är någonting som kommer behöva ändras på. Men så kan man tro att man är väldigt dynamisk i sin lösning men så missar man vissa krav eller att det är för generiskt skulle det också kunna vara som gör att det nästan blir svårt att upprätthålla de viktiga use-case kraven. Lite spontana tankar på vad teknisk skuld innebär när man sitter och jobbar skulle jag säga.
13	F	Är teknisk skuld något ni är påverkade av?
14	R3	Det är lite svårt att säga att vi inte har någonting i produktionen än. Vi är snart där och jag skulle säga så ja, det är klart man får alltid

		göra genvägar när man sitter och bygger saker, speciellt när man är små. Då försöker man ju bevisa sitt case skulle jag säga. Jag tror det är svårt att ha kod som är 100% perfekt eller att man inte har någon skuld på den. Jag tror det men jag skulle inte kunna säga allt, det finns saker som är gömda och vissa som är medvetna.
15	F	Det här medvetna och omedvetna är rätt så intressant. Hur tror ni att ni kommer kunna hantera det sen?
16	R3	Ja det medvetna är ju sånt som man får lyfta. Det medvetna är tydligare och så ska man lösa problem och så inser man att kanske inte vet hela scopet av problemet eller att lösa hela scopet är alldeles för tidskrävande eller svårt, så man försöker lösa mindre delar av det. Där tror jag det är viktigt med kommunikation, att man berättar för andra om det. Så att andra vet om den tekniska skulden, för det kan vara att alla resonerar och andra vet om den liksom. Det är svårare när folk ska resonera kring teknisk skuld när man inte berättar det, då är liksom en medveten skuld som man själv kanske sitter på men när andra ska resonera kring vad man kan göra med systemet så resonerar de fel för att de har fel fakta. Men sen finns det helt totalt omedvetna där man tror att man gör rätt men det är inte rätt, det kommer inte hålla liksom.
17	F	Har ni några planer eller strategier för att hantera den? Några uttänkta redan nu?
18	R3	Ja alltså när man sitter och jobbar med det och vet om att det kommer att bli en skuld och man berättar det. Det ligger ju in i den framtida plan att när vi kommer till den här punkten säg att vi har en slags preview/release till exempel då säger vi ja men scopet till preview/releasen är att kunna göra A,B,C sen kanske A,B,C kan göras väldigt hackigt men så länge vi vet om att hackigt så kan vi säga att från previewn så måste vi faktiskt ta tid för att lösa de här sakerna och det kan vara saker som vi vet hur vi löser så att det är inte så de är omedvetna lösningar. Sen kan det finnas de som man inte vet hur man löser vilket är svårare att estimera. Vi behöver göra det här, just nu fungerar det på det hackiga sättet men vi vet inte den riktiga lösningen på det här och vi måste lägga tid, det blir väldigt mycket research. Det är väl så vi kan resonera kring de medvetna taktiska skulderna, de omedvetna är svåra att säga någonting om. Man kan komma in på morgonen och ja, fin dag och sen så sätter man sig ner och “åh nej” och så kan man sitta där i flera veckor liksom och det är någonting som är ur ens kontroll.
19	F	Hur ni någon plan för när ni måste betala skulden? Märker ni att problemet blir så stort så att ni måste ta tag i det och kan inte undvika det längre?
20	R3	Ja. Jag skulle säga att det är ganska självklart. Oftast så brukar den smärtgränsen eller toleransen, avväga ifall man måste lösa eller inte. Är det att vi inte klarar av att bygga saker längre på ett vettigt sätt då brukar det kännas som en bra måttstock för när man behöver lösa saker. Nu är det smärtsamt nog att vi inte kan vara produktiva längre i och med att man hela tiden slåss mot funktionalitet och implementerad funktionalitet, man har bara så mycket tid, speciellt när man är

		små så man slåss alltid mot att var ska man lägga mitt fokus. Så just den här smärtgränsen, när smärtan uppstår och går över att till att det inte håller längre. Det kan ju antingen vara i produktion eller när man sitter och utvecklar, just utvecklar upplevelsen för att jag ska kunna köra hela min kodmiljö måste jag drar igång 15 tjänster på datorn eller någonting. Det pallar inte datorn längre, då är det smärta okej och det måste lösas på något sätt. Planmässigt är det väldigt oplanerat, nu känner jag smärta och nu måste jag lösa det här men som sagt, de här sakerna som man kanske flaggar för och faktiskt säga vi har. Vi planerar vad vi ska ha för featurereleases eller att ska ha de här sakerna ungefär klara till det här datumet, man inser ungefär med erfarenhet hur mycket man hinner med hur mycket man behöver hacka ihop saker och det ser man ju närmre man kommer sitt planerade tidsmål. Därifrån får man också ha i bakhuvudet hur löser vi det här? och när löser vi det här? hur viktigt är det här? Förutse vissa smärtor som databas backuper, alltså det är smärtsamt ifall vi inte har det och lanserar. Så det är kanske något som vi behöver fixa och vet hur vi löser, ska vi lägga tid på det nu eller sen, det är väl planen skulle jag säga, det är en ganska informell plan. Det är inte så att alla är kanske inte nedskrivna i något tidsschema eller kalender liksom här ska vi lösa de här. Det ligger väldigt mycket på oss huvuden skulle jag säga. och en delge kort alltså i gira liknande kort.
21	F	Det är inte så mycket formellt utan det är mer när man möter det? då tar man hand om det?
22	R3	Ja just det här med smärtan och de kanske mer planerade som vi vet om att vi inte har databas backuper säger vi och det här måste vi fixa men det är planerade åtgärder som vi vill göra om 5 veckor för att vi dem just nu men vi behöver om 5 veckor och vi vet hur vi gör det så att det är ganska lätt att estimera hur lång tid det tar. Det är inte heller något som är formellt nedskrivet men det är något som cirkulerar i konversation och i huvudet på folk och dyker upp lite i våra issues eller tasks men mer informellt och inte så att vi pratat om en teknisk skuld plan på det sättet, utan det är med AD-hoc.
23	F	Så även när man ska skjuta upp betalningar och så, om man funderar på det?
24	R3	Betalning är att man löser skulden på något sätt?
25	F	Ja
26	R3	Förlåt vad var frågan
27	F	När man kommer fram till att man kan skjuta upp betalningar? Det är inte heller informellt utan att man ser att det kommer?
28	R3	Ja det brukar dyka upp i diskussionen, hur mycket tid man har. Vad är det vi hinner med och vad är det mest kritiska. Där får man jobba med, vissa saker är konstanter till exempel vad vi måste hinna med, för vi har ju features och vi har tid och resurser, resurserna går ju inte att röra så mycket på. Tiden för oss just nu gå inte att röra så mycket på så då får vi tumma på features och försöka omsortera så bra vi kan för att nå det vi har sagt till ett visst datum men det är omöjligt att få alla tre, att lira konstanter.

29	F	Med tanke på det du sa, finns det svårigheter med hanteringen? beroende på er storlek?
30	R3	Ja, en svårighet att man faktiskt inte levererar. Kanske sa man att man skulle leverera och då finns det stakeholders och andra intressenter och själva imagen av företaget står ju på spel så det är väldigt nära businessen och extremt viktigt att trycka saker som folk ser värde i. Vilket gör att saker som utvecklare upplevelse som att det är lätt att slå upp en miljö eller att återställa en databas är inte lika prioriterade för att det är inte lika tydlig vinst för dem eftersom det är ingen som ser business caset i det, utan det är något som får halka efter lite och man får svettas lite mer som programmerare ibland och det är lite av charmen men kanske.
31	F	Finns det några fördelar som du kan se med att vara ett litet företag?
32	R3	Ja, det går väldigt snabbt att göra saker. Det är både för att vi ett så kallat green field, vi har inget som håller oss tillbaka egentligen. Vi får bygga hur vi vill och i vad vi vill, vilken teknik, på så sätt har vi väldigt mycket frihet och det går snabbt. När man är få blir det kanske att man har sitt område som man jobbar med och nu när vi bygger en mikrotjänst arkitektur vilket gör att jag oftast är inne mer i vissa tjänster än andra, så jag har lite mer ägandeskap över det, vilket gör att jag kan snabbare gå in och fixa någonting eftersom jag är mer medveten om hur den tjänsten fungerar, man krockar inte så ofta i GIT och får merge conflicts. Någonting som vi inte har just nu är någon rigorös review process där vi kollar varandras kod, man får själv ta ansvar och gå in och kolla vad folk gjort under dagen, men det är inte så att det behöver godkännas för att få in det. Sånt gör att det går mycket snabbare att utveckla, men självklart kanske man får tumma på konsekventhet i koden och buggar, best practices som andra fångar sånt som är bra med review-processer. Men det är samma sak, det är alltid balansen av att spela mot tid och resurser. Men det är går väldigt snabbt att bygga grejer jämfört ett större enterprise eller projekt där man är 100 personer.
33	F	Hade ni gjort något annorlunda med hantering av teknisk skuld om ni hade haft mer resurser?
34	R3	Ja alltså. Mer resurser då kan man till exempel skaffa mer utvecklare eller testare, någon testare i vårt fall, vi jobbar ju lite som testare också. Resurser hjälper ju alltid. Men det som är mest Return on invest är väl kanske att investera i automatisering av testning och sånt, det kräver inte lika många resurser, man kan testa de mest basala sakerna så att det fungerar, kärnfunktionaliteten det ger ganska mycket för har man ändå lite confidence att våga trycka fram features snabbbare för att man vet att grunden fungerar, så det är inte hela systemet kollapsar eller sådär. Det är svårt att veta exakt för om man visste vad man skulle gjort annorlunda om det var lite det som var frågan så är det svårt för att man tror att man alltid gör det bästa eller försöker göra sitt bästa. Men hittills om man tittar tillbaka har det känts som att jag känner att jag har haft ett rimligt angrepp på det, just med vilka features är viktigast för att businessen ska kunna bevisa use-case så just nu är det ganska enkelt att sortera det där i huvudet. Sen tror jag

		att det kommer att bli svårare och svårare ju längre fram man kommer till att man faktiskt börjar ha mer i produktion och blir mer en seriös produkt, det blir mer features och komplexiteten ökar liksom, man vill kanske lansera i flera olika regioner i världen, då kommer det bli mycket svårare att hantera teknisk skuld. Det som är bra med få är att alla vet om den tekniska skulden, den medvetna alltså, den omedvetna den är alltid ett wild card men den medvetna har i alla fall någon som har koll på. Jag har ju väldigt bra koll på vad är lite hackigt så att ifall någon säger att det här vill vi göra så kan jag säga att ja det kan vi göra men då måste vi fixa A eller B eller C. I och med att vi är så få är vi nästan alltid på alla diskussioner så att det gör ju också så att man kan röra sig fort för att alla är experter så att säga på systemet som man bygger är i rummet, vilket inte alltid är fallet när man bygger större system med större team.
35	F	Det blir mindre specialiserade då?
36	R3	Ja exakt. Någon som kanske inte har jobbat så länge på projektet, cirkulation på folk, hur fungerar den här grejen som byggdes för att halvår sedan, nu är den personen inte kvar. De hade någon överlämning men ja det här med överlämningar är också svårt, det brukar oftast vara något som hamnar i något dokument någonstans och sen är det också oförståeligt. Det är svårt med sånt. Så att alla som har byggt… Allt som är byggt är byggt av folk som är i rummet, så det är lättare att ta beslut på det sättet.
37	F	Sen undrar vi om du har något mer du vill tillägga?
38	R3	Nä, egentligen inte. Det enda jag kan säga att teknisk skuld är något som finns, oavsett hur man jobbar så kommer det att finnas till större eller mindre grad. Det är väldigt många som kastar det som ett smutsigt ord men jag känner själv att jag är ganska pragmatisk i det, ja vi tillåter lite teknisk skuld så länge vi är medvetna om det. Det är extra viktigt när man sitter där och gör beslutet och skriver in sin tekniska skuld i produkten eller lösningen så kan det vara bra att slå en extra tanke liksom. Tänk så hur skulle jag lösa det i framtiden? Är det så att jag inte vet hur jag löser det är det kanske lite dummare att göra snabba lösningen för det kan vara så att man bygger in funktionalitet som egentligen är omöjlig att uppnå. Men ifall man bygger något snabbt för att för att man måste ha ut snabbt men vet hur man skulle lösa det på så att säga på riktigt sen så skulle jag se det som att det är ganska låg risk att få in den tekniska skulden då. Just när man kastar in okända variabler som att man inte vet om man kan lösa det och ja som sagt omedvetna, det är svårare att hantera.
39	F	Så det är viktigt att.. Man pratar ofta att det är en hävstång teknisk skuld.. Att det mer liknas som ett finansiellt lån. Det är viktigt att det inte blir för stort då?
40	R3	Ja precis, det är någonting man vill ha. Annars kanske man inte pushar features och inte testar vad man kan göra med produkten liksom. Och det märker man på kanske, nu är det lite antagande men man märker det på större företag som Google eller Facebook. Ja tror Zuckerberg har sagt väl sagt “break things fast”, det är väldigt teknisk

		skuldigt att ha sönder saker snabbt, då pushar man features och allt kanske inte testas vid en teknisk skuld så man får liksom lösa sen men man fick bevisa ett koncept kanske, vilket kan vara värt mycket mer pengar och det värdet är så mycket större än vad det kostar att fixa skulden sen, det är någon slags ekonomisk balansgång där.
41	F	Om det inte var något mer så tackar vi för oss!

Appendix 5: Intervju 4

Tid: 8 maj 2019 kl. 10.00

Plats: Respondentens arbetsplats

Deltagande: Christopher Andersson, Martin Larsson & Felix Lidforsen

Respondent: Babak Vahidi

Författare/Forskare = F
Respondent = R4

Nummer	Person	Fråga/svar
1	F	Vad heter du?
2	R4	Babak
3	F	Vad är din roll på företaget?
4	R4	Jag är CTO
5	F	Hur ser en dag ut på ditt jobb?
6	R4	De är oftast inte jättelika varandra. Jag försöker vara med på en daily standup med våra utvecklare och synkronisera arbetet mellan dem och oss lite, och även se om det är några frågor jag behöver ta itu med under dagen. Sen har jag väldigt mycket fokus på projektledning, vi har ganska många kunder och ganska många projekt pågående samtidigt så jag brukar kolla på det också. Sen försöker jag få lite tid över för att fundera på de strategiska frågorna som hur vi ska organisera oss, rekryteringar och sådana grejer.
7	F	Skulle du kunna berätta lite om din bakgrund?
8	R4	Jag pluggade IT på Chalmers med fokus på artificiell intelligens, och sen jobbade jag som AI-utvecklare i några år, på mindre bolag, och sen utvecklades jag från utvecklarrollen till först produktchef, produktägare och sedan operativ chef osv. Då tyckte jag att jag gillar de ekonomifrågorna, ledningsfrågorna ganska mycket så efter att ha jobbat med de i några år så tog jag tjänstledigt och läste på Lunds universitet. Där läste jag en master som heter ”Corporate entrepreneurship and innovation management”, och sen kom jag tillbaka till arbetslivet och började gå tillbaka till teknikdelen och jobbade mycket som CTO. De senaste tre jobben har varit CTO på mindre företag.
9	F	Är du medveten om vad teknisk skuld är?
10	R4	Jag vet inte om jag har läst någon definition av det, men jag tror att jag vet.
11	F	Hur skulle du definiera det?
12	R4	Det är avvägningar man gör, eller tradeoffs, för att snabbare kunna implementera något för att nå sina deadlines, som man vet att någon gång i framtiden kommer man få fixa det här.
13	F	Är ert företag påverkat av teknisk skuld?

14	R4	Jag frågade våra utvecklare det här och de svarade ”Nej vi gör allting rätt från början”. Men det är klart att som uppstartsföretag har man inte så mycket resurser och då blir det väldigt många såna här frågor där man får avväga om man ska göra något snabbt, vilket man gör ofta. Så vi är inte, vi har väldigt mycket teknisk skuld också.
15	F	Men du var lite inne på det, de här avvägningarna, på vilket sätt?
16	R4	Det kan vara att man har, när man ska sitta och koda så vet man till exempel att man skulle behöva en databas för att hantera vissa uppgifter men så tycker man att det tar för lång tid att göra eller att det är för mycket arbete för en liten grej. Då hårdkodar man in vissa saker i koden och sen vet man att i framtiden så kommer vi få ha en databas men nu väljer vi denna lösningen för att det går mycket snabbare. Det är ett exempel, men många gånger är det så att man får avväga.
17	F	Är det ofta på det sättet som teknisk skuld uppkommer, alltså medvetna beslut? Som det med databasen du nämnde.
18	R4	Fast om det inte är medvetet skulle jag säga att det inte är teknisk skuld utan, ja det var en bra fråga jag har inte tänkt på det. Jag vet faktiskt inte, för man benämner ju det teknisk skuld även om man hittar det sen och det inte varit en medveten avvägning. Jag har på något sätt funderat på teknisk skuld som en medveten handling.
19	F	Har företaget några utmaningar gällande teknisk skuld?
20	R4	De utmaningar som finns är såklart att veta när man ska låta något vara och ta den tekniska skulden. Lite också när man ska fixa den, betala den. Sen är det kommunikationsutmaningar med resten av organisationen och förklara varför man sysslar med att betala av en skuld just nu istället för att utveckla en ny feature eller liknande.
21	F	Har ni några medvetna strategier för att hantera den tekniska skulden?
22	R4	Vi har sagt, vår organisation har ett produktteam som består av mig själv och en designer och marknadschefen och operativa chefen som bestämmer lite hur produkten ska utvecklas och därmed också vad utvecklarna ska sitta och koda. Men vi har sagt att produktgruppen får bestämma hälften av vad utvecklarna ska ägna sig åt och resten är upp till det tekniska teamet själv, och av det arbetet som de bestämmer själva tar vi upp väldigt mycket av den tekniska skulden. Det är lite vår strategi, att dela upp tiden mellan produktteam och teknikteam.
23	F	Hur kommer ni fram till att en skuld ska betalas?
24	R4	Det är väl ofta att det kommer en bugg, som kanske inte alltid beror på den tekniska skulden, men för att fixa buggen måste man kanske hantera skulden också. Då diskuterar man och kanske kommer fram till att man kan lika gärna göra något annat samtidigt. Det är nästan alltid så att det fixas i samband med att det kommer en bugg.
25	F	Så det är lite informellt, när det dyker så hanterar man det?
26	R4	Jag vet att vissa företag har temaveckor då de bara sitter och fixar teknisk skuld men det har inte vi.
27	F	Händer det att ni väljer att skjuta upp en betalning?
28	R4	Det händer. Dessutom händer det väl att vi bara betalar en del av skulden och inte fixar allting. Detta av samma anledning egentligen,

		att det innebär en för stor kostnad att fixa allt. Men det händer ganska ofta.
29	F	Vilka anledningar finns det då för att skjuta upp det?
30	R4	Att man tycker att den tekniska skulden inte har en för stor påverkan på verksamheten, eller produkten, eller kunderna alternativt att det är för mycket jobb att fixa den tekniska skulden jämfört med vad man vinner.
31	F	Tror du att det finns några svårigheter med hantering av teknisk skuld med hänsyn till företagets storlek?
32	R4	Ja men det tror jag. Jag tror att i mindre företag med färre utvecklare har man mindre resurser att lägga tid på sånt där och på att utreda. Då blir det lite mer skuld jämfört med när jag jobbat i större företag där man haft hundratals utvecklare och projektledare och liknande med analysmänniskor och allt möjligt. Då har man haft mer tid att behandla de här frågorna men då har utvecklingstiden å andra sidan lidit av det. Jag tror det är lite svårare i mindre företag.
33	F	Om vi vänder på det, finns det några fördelar?
34	R4	Ja, fördelen är att i större företag kan det ibland bli så att man måste fixa till det här, eller om man jobbar på konsultföretag och man gör något åt kunden vill man på något vis täcka sin rygg och inte åka på att det här gjorde ni dåligt. Då fixar man såna saker som man vet att man egentligen inte behövt göra, eller till exempel om man har tio saker som man måste göra en avvägning om man vill ta en teknisk skuld eller inte. Då kanske en av dem faktiskt har en impact men de andra nio har inte det. I vårt företag kanske man valt att skjuta upp alla tio och tänker att det blir en mindre kostnad sen medan ett större företag tänker att man måste fixa alla tio för att en sak kan få stor impact i deras fall. Då måste man skjuta upp den.
35	F	Om ni hade haft mer resurser, hade ni gjort något annorlunda?
36	R4	Vi skulle utrett vissa frågor mer, nu diskuterar vi väldigt informellt. Det hade kanske varit mer fokus på att faktiskt utreda vissa implementeringar jämfört med andra. Sen skulle vi såklart ha haft mer tid att lägga på att betala av skulden.
37	F	Finns det något mer du vill tillägga?
38	R4	Angående teknisk skuld?
39	F	Eller generellt om du känner att något saknas?
40	R4	Nej, alla utvecklare känns som att de är väldigt medvetna om vad teknisk skuld är. Vi försöker också prata om motsatsen, teknisk ”wealth” vi har inget bättre ord, vad det är och hur man kan jobba med det. Det är det inte så många utvecklare som är medvetna om.
41	F	Ja begreppet är lite negativt betonat.
42	R4	Teknisk skuld menar du? Ja, och det ska det inte vara egentligen.
43	F	Man brukar ju likna det vid ett finansiellt lån; vilket nytt företag tar inte ett finansiellt lån?
44	R4	Nej jag tycker inte det är negativt faktiskt alls, och du har rätt att det har lite negativ klang. Men som litet företag och satsa på att göra allting rätt från början är väldigt dåligt och dyrt. Då riskerar man verksamheten på andra sätt. Det var det jag hade och säga.
45	F	Stort tack!

Appendix 6: Intervju 5

Tid: 9 maj 2019 kl. 13.00

Plats: Respondentens arbetsplats

Deltagande: Christopher Andersson, Martin Larsson & Felix Lidforsen

Respondent: Anonym

Författare/Forskare = F
Respondent = R5

Nummer	Person	Fråga/svar
1	F	Vad heter du?
2	R5	*anonym*
3	F	Vad är din roll på företaget?
4	R5	Front end utvecklare
5	F	Skulle du kunna berätta lite om din bakgrund?
6	R5	Ja, jag pluggade systemvetenskap och gick ut 2015. Sedan dess har jag jobbat med front-end utveckling som konsult och nu på det här företaget är det är det första gången jag jobbar med en produkt.
7	F	Är du medveten om vad teknisk skuld är?
8	R5	Ja det skulle jag nog säga, det beror kanske lite på definitionen av teknisk skuld.
9	F	Hur skulle du definiera teknisk skuld?
10	R5	Det skulle jag säga som ett behov av att refaktorera kodbasen och möta nya omständigheter. Stämmer det med er definition?
11	F	Jo det kan man säga, väldigt kortfattat absolut. Är ert företag påverkat av teknisk skuld?
12	R5	Ja det skulle jag nog säga, jag tror att alla företag är det men att det är som en skala där man kan vara mer eller mindre påverkade av det, så absolut.
13	F	Vad avgör var på denna skalan som ett företag hamnar?
14	R5	Ja egentligen kan man väl säga att det är behovet av fakturering och hur mycket tid man lägger på att fakturera en kodbas.
15	F	Har företaget några speciella utmaningar gällande teknisk skuld?
16	R5	Jag vet inte men så som vi påverkas av teknisk skuld är att nu bygger vi ju en ny produkt, vi har inte byggt på den så länge heller så vi påverkas av teknisk skuld på så sätt att våra business krav inte riktigt är satta än så att vår kodbas som vi har behöver att vi jobbar med den och hantera den som något som förändras kontinuerligt. Så det kan vi på ett sätt säga är en teknisk skuld som vi har, när vi skriver kod en månad så kanske nästa månad läggs på att skriva om den för att vi insåg att de kunderna vi kommer att ha inte vill jobba på det här sättet. Så det är väl i den fasen vi är i just nu.

17	F	Har ni några medvetna strategier för att hantera den tekniska skulden?
18	R5	Jag skulle inte säga att vi har medvetna strategier direkt men jag tror att som en följd av hur företaget ser ut, att vi är ett litet företag med få utvecklare gör att vi kan väldigt snabbt hantera den. Med team med fler utvecklare så blir det betydligt svårare att hantera en teknisk skuld för du måste synca med alla kring hur man ska refaktorera kodbasen och det är lätt att man kan få krockar då om 2 personer jobbar på samma sak. Så just de här mindre företagen och mindre team gör att det blir lättare att hantera teknisk skuld. Jag ska också säga hur duktiga programmerarna är som man jobbar med gör väldigt stor skillnad.
19	F	Kan du ge ett exempel på hur det kan variera, baserat på kompetensen hos utvecklarna?
20	R5	Ja för dels så är det hur man jobbar med teknisk skuld när den väl uppkommit alltså bara ren förståelse av kodbasen, hur man ska ändra om saker. Men vad som kanske är ännu viktigare är hur man förebygger teknisk skuld. Att man kan se ungefär hur man vill strukturera upp koden för att mycket handlar om att, eller jag tror man kan se på teknisk skuld på två sätt. Så det ena är att som jag pratade om innan att om business kraven förändras så kommer också kodbasen behöva förändras då har man en teknisk skuld. Men det är också det som man förutsåg att man skulle behöva göra är fel, så vi kan ju nu när vi sitter och bestämmer hur ska vi strukturera upp vår applikation så behöver vi göra vissa antaganden om framtiden, hur vi bäst bör göra det här och där så spelar erfarenhet in mycket tror jag och kunna förutsäga vilka svårigheter kommer vi ha i framtiden? Och förebygga det.
21	F	Så erfarenheten är viktig där?
22	R5	Ja precis det tror jag verkligen.
23	F	När man har en teknisk skuld hur beslutar man att man måste betala den eller åtgärda den?
24	R5	Det är dumt att betala en teknisk skuld innan man har behovet av det för då finns det en risk att man bara skapar ännu mer teknisk skuld så att det bästa är nog att man väntar och ser ”okej nu lider vi av det här så pass mycket” så att nu måste vi faktiskt göra något åt det och det är något man hela tiden kontinuerligt måste utvärdera ”okej måste vi åtgärda det här nu eller kan det vänta” och när kostnaden blir för hög att fortsätta jobba med något som faktiskt är en skuld då är det dags att göra något åt det. Och det kan vara en väldigt svår avvägning att göra, där kommer också erfarenheten in igen. Det är bara erfarenheten som säger ”okej nu är det faktiskt läge att spendera några dagar eller en vecka på att faktiskt göra om det här så vi sparar in tid i framtiden.
25	F	Om det händer att ni väljer att skjuta upp en teknisk skuld, vad är det som motiverar det?
26	R5	Då tror jag att det finns två anledningar, antingen är det då att man tar beslutet aktivt att ”okej vi lider inte av det här så mycket just nu så vi kan fortsätta ha det som det är. Men den andra grejen är också att när

		man har krav utifrån, det kanske är mer applicerbart om man är konsult, då måste ju kunden också gå med på att man spenderar den här tiden på att åtgärda en teknisk skuld vilket de inte alltid är så pigga på att göra. Då blir det på något sätt en, ja då får man de kraven på sig utifrån att man ska fortsätta att jobba med den här tekniska skulden.
27	F	Är det huvudsakligen i din erfarenhet den stora anledningen att man skjuter upp något, krav utifrån liksom?
28	R5	Ja, ja precis. Hade man haft obegränsat med tid och resurser så hade man ju åtgärdat det direkt antagligen men det är ju då någon annan som beslutar om vad som ska göras och då kanske de prioriterar annorlunda än vad utvecklarna gör.
29	F	Tror du att det är kunskapsskillnad mellan personer som tar beslut och utvecklarna?
30	R5	Ja definitivt, det är nog den största anledningen till att det är så. Många gånger kan ju de som tar de här besluten vara så insatta i hur det faktiskt fungerar med programmering och så.
31	F	Finns det några svårigheter med hantering av teknisk skuld med hänsyn till företagets storlek?
32	R5	Ja det va det vi var inne på tidigare då att det är mycket lättare att vara rörligt och flexibel kring hur man hanterar teknisk skuld när man är mindre. Det blir mindre att synca mellan programmerarna och alla som är inblandade och jobbar med koden, så definitivt. Ju större teamet är desto svårare blir det att jobba med det.
33	F	Men finns det några nackdelar med att vara ett litet företag också? Även om du har enkelheten i strukturen och bland programmerarna men det kanske inte bara är positivt.
34	R5	Nä det skulle vara en generell sak isåfall. Det skulle vara att det blir mindre gjort, man har ju mindre kapacitet, man är färre människor så det är ju en av fördelarna om man har ett större team så kan man oftast leverera features snabbare men man bygger också upp sin tekniska skuld snabbare.
35	F	Om ni hade haft mer resurser tillgängliga hade ni hanterat den tekniska skulden annorlunda?
36	R5	Ja alltså man måste nog ha lite mer, då är det nog viktigare. Vad sade du? ”Om ni hade haft mer resurser?”
37	F	Ja
38	R5	Hade vi haft mer resurser så hade vi haft mer människor som jobbade på det och då hade vi nog behövt ha betydligt mer struktur kring hur vi jobbar med teknisk skuld. För det kan jag säga, nu när man är ett litet team så kan det bli ganska informellt. Att man bara beslutar sig för att förändringar snabbt, men ju större team man är och ju mer resurser man har ju mer behov finns det kring struktur. Man kanske planerar in det här ”om tre veckor då börjar vi angripa det här problemet med teknisk skuld” då har vi tid där och så blir det kanske mer uppstyrt och exakt hur ska vi jobba med det här. Ju mer resurser ju större behov av att strukturera upp arbetet med teknisk skuld.
39	F	Finns det något mer du vill tillägga som vi har missat?
40	R5	Jag kan väl säga det här med, det jag sade innan. Antagligen de två största orsakerna till teknisk skuld är att de förutsägelserna man

		gjorde om framtiden blev fel eller att business kraven förändras så helt plötsligt står man där med features som är föråldrade. När det kommer till att business kraven förändras så är det väldigt viktigt med kommunikationen mellan de som ställer de här kraven, det som kommer från business-sidan. Att man verkligen förstår varandra så man kan förebygga att det uppkommer en teknisk skuld och sen då för att förhindra att man gör fel förutsägelser då är det väldigt viktigt att ha personer som faktiskt är kunniga och kompetenta som faktiskt kan förutsäga varifrån de kommer så det är väl de två sakerna som antagligen är viktigast, kommunikation och kompetens

Referenser

Alves, N. S. R., Mendes, T. S., de Mendonça, M. G., Spínola, R. O., Shull, F. & Seaman, C. (2016).Identification and Management of Technical Debt: A Systematic Mapping Study, *Information and Software Technology*, vol. 70, no. 100-121

Ashford, W. (2019). *Facebook Security Policy and Practices Unfit, Say Infosec Pros* [Online]. Available online: https://www.computerweekly.com/news/252460044/Facebook-security-policy-and-practices-unfit-say-infosec-pros?fbclid=IwAR23vl_v1rCctl_B_tbj6j6w_U1uTTCDd4MTW8D29D7X16o-U7k_GYhaWfk [Accessed 11th April 2019].

Besker, T., Martini, A. & Bosch, J. (2018a). Technical Debt Cripples Software Developer Productivity: A Longitudinal Study on Developers' Daily Software Development Work. ACM.

Besker, T., Martini, A., Lokuge, R. E., Blincoe, K. & Bosch, J. (2018b). Embracing Technical Debt, from a Startup Company Perspective. IEEE.

Cunningham, W. (1992). The Wycash Portfolio Management System. *Addendum to the proceedings on Object-oriented programming systems, languages, and applications (Addendum).* Vancouver, British Columbia, Canada: ACM.

Curtis, B., Sappidi, J. & Szynkarski, A. (2012). Estimating the Size, Cost, and Types of Technical Debt. IEEE.

DevIQ. (2019). *Technical Debt All Things in Moderation.* [Online]. Available online: https://deviq.com/technical-debt/ [Accessed April 18th 2019]

Ekonomifakta. (2018). *Företagens Storlek* [Online]. Available online: https://www.ekonomifakta.se/fakta/foretagande/naringslivet/naringslivets-struktur/?fbclid=IwAR3wot6xibU3xAWEk3QASovcYK9-asmJPZOcH529i_-X_Gxxx1atKGz668A [Accessed 10th April 2019].

Europeiska kommissionen. (2015). *Användarhandledning Om Definitionen Av Smf-Företag* [Online]. Available online: https://www.vinnova.se/globalassets/dokument/eu-definition-smf.pdf?fbclid=IwAR0rW6X-ros5JbBGvJB0Ve7QVdla7PAfLxDzYYpL1YCnc-WxAvRpxuUgwvc [Accessed 10th April 2019].

Fowler, M. (2003). *Technicaldebt* [Online]. Available online: https://martinfowler.com/bliki/TechnicalDebt.html [Accessed 12th April 2019].

Fowler, M. (2009). *Technicaldebtquadrant* [Online]. Available online: https://martinfowler.com/bliki/TechnicalDebtQuadrant.html [Accessed 10th April 2019].

Jacobsen, D. I. (2002). Vad, Hur, Varför? Om Metodval I Företagsekonomi Och Andra Samhällsvetenskapliga Ämnen: Studentlitteratur.

Kim, D., Hong, J.-E., Yoon, I. & Lee, S.-H. (2018). Code Refactoring Techniques for Reducing Energy Consumption in Embedded Computing Environment.

Kruchten, P. (2012). Strategic Management of Technical Debt: Tutorial Synopsis. IEEE.

Kruchten, P., Nord, R. L., Ozkaya, I. & Falessi, D. (2013).Technical Debt: Towards a Crisper Definition Report on the 4th International Workshop on Managing Technical Debt %J Sigsoft Softw. Eng. Notes, vol. 38, no. 5**,** pp 51-54

Li, Z., Avgeriou, P. & Liang, P. (2015).A Systematic Mapping Study on Technical Debt and Its Management, *The Journal of Systems & Software*, vol. 101, no. 193-220

Majchrowski, A., Ponsard, C., Saadaoui, S., Flamand, J. & Deprez, J. C. (2016).Software Development Practices in Small Entities: An Iso29110-Based Survey, *Journal of Software: Evolution and Process,* vol. 28, no. 11**,** pp 990-999

Maldonado, E. D. S., Abdalkareem, R., Shibab, E. & Serebrenik, A. (2017). An Empirical Study on the Removal of Self-Admitted Technical Debt. IEEE.

Martini, A. & Bosch, J. (2016). An Empirically Developed Method to Aid Decisions on Architectural Technical Debt Refactoring: Anacondebt. *Proceedings of the 38th International Conference on Software Engineering Companion.* Austin, Texas: ACM.

Oates, B. J. (2006).Researching Information Systems and Computing. London: Sage., vol. no.

Ponsard, C. & Deprez, J.-C. (2017). Helping Smes to Better Develop Software: Experience Report and Challenges Ahead. ACM.

Richardson, I. & von Wangeheim, C. G. (2007).Why Are Small Software Organizations Different?, *IEEE Software,* vol. 24, no. 1**,** pp 18-22

Sierra, G., Tahmid, A., Shibab, E. & Tsantalis, N. (2019). Is Self-Admitted Technical Debt a Good Indicator of Architectural Divergences? : IEEE.

Stripe. (2018). *The Developer Coefficient* [Online]. Available online: https://stripe.com/files/reports/the-developer-coefficient.pdf?fbclid=IwAR1WorQOlbU-Lp9S4Sm5jdgmCZqFiua74Fs3ytwC0H5N6lxP3w8TLUN8DHg [Accessed 11th April 2019].

Tom, E., Aurum, A. & Vidgen, R. (2013).An Exploration of Technical Debt, *The Journal of Systems & Software,* vol. 86, no. 1498-1516

Trumler, W. & Paulisch, F. (2016). How “Specification by Example” and Test-Driven Development Help to Avoid Technial Debt. IEEE.

Yli-Huumo, J., Maglyas, A. & Smolander, K. (2016).How Do Software Development Teams Manage Technical Debt? – an Empirical Study, *The Journal of Systems & Software,* vol. 120, no. 195-218