DATALOGISKT TÄNKANDE -
ARBETSGIVARENS PREFERENSER

Kandidatuppsats i Informatik

Rasmus Brants
Alexander Johansson

VT 2017:KANI01

HÖGSKOLAN
I BORÅS

Svensk titel: Datalogiskt tänkande – Arbetsgivarens preferenser

Engelsk titel: Computational Thinking- Employers preferences

Utgivningsår: 2017

Författare: Rasmus Brants & Alexander Johansson

Handledare: Carina Hallqvist

Abstract

Sweden and the rest of the world are constantly evolving in the digital aspect. Every year more technology is being implemented in all the fields which make the technology in control of all the working fields. These changes have made Sweden and other countries to start with programming in the elementary school. This is to develop the individual's computational thinking. In this thesis, computational thinking is researched as a property of an individual and how employers prioritize computational thinking at a potential employment within their business as a software developer. Through interviews and surveys the researches have collected data and constructed their own categories from programming and computational thinking which was used in comparison with collected data from different employers. This study has shown that employers actually prefer computational thinking over programming-skills which support the recent implementation of computational thinking.
This thesis is unique because none or very few previous researchers with a study revolving computational thinking where categorizes have been used to compare the two different fields. The researches strongly believe that academic stakeholders within computer science and organizations that uses software developing can have great use of this thesis' results. This study enlightens a unique result which most likely has not been conducted before. Potential consequences of this study are that the authors hope that computational thinking will acquire more attention. A clear knowledge grant of the previous research about computational thinking is being presented in this study which can affect the future research and implementation of the term computational thinking.

Keywords: Computational thinking, abstraction, programming languages, stereotypes, categorized analysis

Sammanfattning

Sverige och världen blir mer digitaliserat för varje år. Varje år implementeras ny teknik i alla branscher vilket innebär att tekniken kan komma att styra arbetsbranschen. Detta har gjort att Sverige och andra länder har börjat med programmering i grundskolan. Anledning till det är för att utveckla individens datalogiska tänkande. I den här studien undersöks datalogiskt tänkande som en egenskap som en individ innehar och hur arbetsgivaren inom mjukvaru-utveckling prioriterar datalogiskt tänkande vid en potentiell anställning som programmerare. Genom intervjuer och enkäter har forskarna samlat in data och konstruerat egna kategorier ifrån programmering och datalogiskt tänkande för att sedan jämföra dessa mot den insamlade data från olika arbetsgivare. Studien har visat att arbetsgivaren faktiskt prioriterar datalogiskt tänkande över programmeringskunskaper vilket stödjer den senaste implementationen av datalogiskt tänkande.
Den här studien är unik då det finns få eller ingen tidigare forskning om datalogiskt tänkande där en kategorisering har genomförts inom de två områdena. Författarna tror starkt på att akademiska intressenter inom data- och systemvetenskap men även företag som bedriver programmering kan ha stor nytta av att ta del av den här studien. Studien belyser ett unikt resultat som troligtvis inte genomförts tidigare. Möjliga positiva konsekvenser av den här studien är att författarna hoppas på att datalogiskt tänkande får mer uppmärksamhet av branschen. Ett tydligt kunskapsbidrag till den tidigare forskningen om datalogiskt tänkande presenteras i studien vilket kan påverka den framtida forskningen och implementationen av begreppet datalogiskt tänkande.

Nyckelord: Datalogiskt tänkande, abstraktion, programspråk, stereotyper, kategoriserad analys

Innehållsförteckning

Förord

Vi skulle vilja uttrycka ett stort tack till vår handledare Carina Hallqvist som hjälpt oss under hela processen och alltid styrt oss i rätt riktning.

Vi vill även tacka informanternas och respondenterna för deras tid och delaktighet i den här studien.

- *Rasmus Brants och Alexander Johansson*

1 Introduktion

Det svenska samhället blir mer och mer digitaliserat för varje år och utvecklas i en väldig fart (Entreprenör, 2015), vilket innebär att *datalogiskt tänkande* har kommit att bli en viktig egenskap att införskaffa sig. Datalogiskt tänkande definieras som en *problemlösningsprocess* för att beskriva, analysera och lösa problem med hjälp av datorer (Wing, 2008).

Karin Nygård (2015) beskriver att datalogiskt tänkande används vid programmering. Kod är det som skrivs för att få datorn att utföra något. Att skriva själva koden heter att koda. För att kunna utveckla applikationer av kod, krävs det mer av programutvecklaren än bara själva kodningen. Det behövs en planering och strukturering av koden för att se hur sambandet och arkitekturen i applikationen agerar. Att koda är en del av programmering, men programmeringen innefattar även problemlösning och algoritmkonstruktion. För att kunna programmera används datalogiskt tänkande enligt Nygård (2015). Det innebär att lösa komplexa problem genom att bryta ner dem till mindre bitar, skapa abstraktioner och leta mönster. Detta kan enligt Nygård (2015) tillämpas på all form av problemlösning, inte bara programmering.

Datalogiskt tänkande beräknas komma att vara en influens i varje arbetsområde, (Wing, 2008) vilket tros komma vara en ny utmaning inom vårt utbildningssystem. Detta har uppmärksammats i Sverige och datalogiskt tänkande har blivit en prioritet i ung ålder just på grund av hur viktig det är för framtiden (Pålsson, 2016). England anses ligga före i utvecklingen och har redan implementerat datalogiskt tänkande i de yngre skolåren sedan några år tillbaka (BareFootCas, 2016).

Datalogiskt tänkande kan komma att förbättra individers problemlösningsförmåga vilket är viktigt i dagens samhälle (Wing, 2008). Eftersom det redan har influerat arbeten och eftergymnasial utbildning, kommer datalogiskt tänkande vara viktigt att utveckla i Sveriges utbildningssystem, från lågstadiet till gymnasiet. Med andra ord skall det implementeras i tidig ålder (Wing, 2008). Sverige har tagit efter andra länder som införskaffat det, och enligt Pålsson (2015) har programmering inkluderats i läroplanen i 16 europeiska länder på en nationell, regional eller lokal nivå.

Det är viktigt att de yngre generationerna förstår att den digitala världen är uppbyggd av kod (Nygård, 2015). Eftersom datalogiskt tänkande är ett relativt nytt begrepp som inte tog fart förrän Wing (2006) skrev sin första artikel om ämnet, vilket har gjort att begreppet datalogiskt tänkande inte uppmärksammats i en större utsträckning.

En person som även är skicklig i datalogiskt tänkande har många eftertraktade egenskaper. Den främsta egenskapen är att individen anskaffar sig verktygen för att kunna lära sig nya programspråk och ramverk när situationen uppkommer. Det är en livslång utbildning och det lärs inte ut detaljer som i en utbildning inom ett visst programspråk. Kreativitet och problemlösning är två viktiga kriterier som datalogiskt tänkande lägger mycket vikt på (Lee, 2008)

Anledningen till att datalogiskt tänkande kommit att få mycket uppmärksamhet är inte bara för att samhället digitaliseras (Entreprenör, 2015), utan också för att utbildningen inom programmering har varit långsam med att övergå från modellen som fokuserar mest på programmeringen till den modellen som är mer generell och lättare att förstå för samtliga

inblandade (Henderson, 2009). Den modellen som inriktar sig på programmering och programspråken i sig har utvecklats otroligt mycket och fortsätter att utvecklas varje dag.

Det finns lite eller ingen tidigare forskning om den jämförelsen mellan datalogiskt tänkande och kompetens inom ett specifikt programspråk, speciellt inte i en större utsträckning. Där ses en möjlighet att upplysa samtliga intressenter om förhållandet mellan dem och lägga en grund för framtida forskning inom området.

1.1 Problemdiskussion

Efter Skolverkets implementation av programmering i grundskolan, blir det tydligt att det inte är det specifika programspråket som är i fokus, utan fokus ligger på att utveckla det datalogiska tänkandet. Som Lee (2008) beskriver är själva målet med utbildningar inom informatik att ge studenten verktygen för att utveckla förmågan att lära sig nya programspråk och inte det specifika programspråket som utbildningen undervisas i. Det är kreativiteten och förståelsen som skall utvecklas (Abelli, 2004).

Enligt Naces (2015) undersökning om vad arbetsgivare faktiskt letar efter vid arbetsintervjuer är problemlösning, förmåga att samarbeta i grupp, kommunikation och initiativ är exempel på några av de förmågor som en arbetssökande förväntas att ha. Problemlösning är alltså ett område som är i fokus hos arbetsgivaren. Datalogiskt tänkande och problemlösning går hand i hand och borde därför uppmärksammas mer inom det data- och systemvetenskapliga området.

Butcher och Tuttle (2016) skriver att i USA är de mest eftertraktade programspråk på Wall-Street är C# och Java. Även här är det i liten grad fokus på individens kvalitéer utöver erfarenhet och kunskapen om programspråket. Det är en återkommande observation och kan förvisso dementeras efter intervjuer då analytiska förmågar sällan kan visualiseras på papper utan måste visas genom olika tester.

Datalogiskt tänkande är något som inte endast programmerare har nytta av, det är något som varje individ borde eftersträva (Hunt & Riley 2014). Att lära sig lösa problem utifrån aspekten att en dator ska kunna göra det, är något som de flesta borde lära sig menar Hunt och Riley (2014). Hunt och Riley (2014) skriver även att alla problem i världen utanför datorer kan lösas med ett datalogiskt tänkande eftersom ifall en dator kan hitta en lösning borde människor även kunna göra det. Ett datalogiskt tänkande bör vara en kvalité hos en individ som är en prioritering vid anställningar, om programspråkutlärning är något arbetsgivaren kan tänka sig att investera i. De anser även att datalogiskt tänkande är något som en individ måste ha för att kunna utvecklas.

Även om datalogiskt tänkande är någonting som används varje dag av människor, utan att de tänker på det, kan det komma att uppmärksammas mer och göras till en kvalifikation under arbetsintervjuer eller liknande scenarion. Som tidigare nämnts så kommer programmering att införas i låg- och mellanstadier i skolor i Sverige (Skolverket, 2016). Det är en faktor som gör att människor i låg ålder kan utveckla sitt datalogiska tänkande vilket leder till att en större utveckling existerar inom området som kan utnyttjas i senare åldrar.

Ur de här punkterna ovan har fokus lagts på problem angående datalogiskt tänkande och de krav av arbetsgivare som finns på programspråkskunskaper. Det leder fram till en diskussion om vad arbetsgivare faktiskt letar efter när de letar efter en ny anställd till företaget som programmerare.

1.2 Problemformulering och frågeställning

Problemet som identifierats i ovanstående problemdiskussion är att det inte finns mycket information i Sverige om vilken betydelse datalogiskt tänkande har för en eventuell anställning, men även bristande eller icke existerande information om vilka personlighetsdrag som föredras i sagd situation. Den generella informationen brukar ange vilket programspråk företaget verkar i och att kunskap inom givet programspråk är att föredra.

Utifrån denna problemformulering har följande övergripande forskningsfråga formulerats:

- *Vilka personliga egenskaper och kunskaper efterfrågas vid en anställning?*

Mer specifikt kommer studien att fokusera på följande frågeställning:

- *Vad hos individen söker arbetsgivare främst efter - ett datalogiskt tänkande eller efterfrågas erfarenheten och kunskapen inom ett specifikt programspråk?*

Utfallet av given frågeställning bör resultera i större förståelse angående relationen mellan datalogiskt tänkande och kunskap inom ett specifikt programspråk. Det kommer också resultera i en kartläggning av diverser egenskaper och dess relevans vid en anställning.

1.3 Syfte

Syftet med uppsatsen är att utföra en jämförande studie mellan datalogiskt tänkande och specifika programspråk, med inslag av personliga egenskaper, som resulterar i en större förståelse inom området. Syftet är även att hitta de eventuella stereotyper och egenskaper som kan återfinnas i programspråk och datalogiskt tänkande för att identifiera de mest eftersökta av arbetsgivarna. Målet med studien är även att generera ny kunskap som andra forskare kan ta lärdom av och sedan göra ytterligare forskning på området.

1.4 Målgrupp

Den här studien riktar sig mot akademiska intressenter med data- och systemvetenskaplig bakgrund. Den riktar sig även mot mjukvaruföretag vilka kan ha användning av studiens resultat vid eventuella rekryteringsprocesser.

1.5 Avgränsningar för studien

De avgränsningar som gjorts i studien är att data endast är insamlad från Västra Götalands län, med majoritet i Borås Stad. Motiveringen till denna avgränsning är eftersom det är begränsade resurser vilket gör insamling av data från andra län än Västra Götaland problematiskt. Studien avgränsar även att endast samla in data från seniora mjukvaru-utvecklare och rekryterare av utvecklare.

Företagsstorleken är en aspekt som kan vara diskuterbar när det gäller avgränsningar. Huvudsyftet för studien är att ta fram de prioriteringar som finns hos arbetsgivare. Vilket innebär att det krävs att företaget aktivt rekryterar nya utvecklare till deras verksamhet. Dock har ingen avgränsning gjorts för detta utan användningen av mindre företag har gjorts ändå men endast med seniora program-utvecklare som har lång erfarenhet av branschen. Deras åsikt värderas då som pålitlig och insiktsfull i det här ämnet.

2 Teoretiskt ramverk

Det teoretiska ramverket fokuserar både på individers egenskaper inom datalogiskt tänkande och på deras kunskap om specifika programmeringsspråk, vilka används som grund för intervjuer och enkäter. Utifrån nedanstående definitioner har ett flertal kategorier tagits ut inom respektive område. Det har även skapats stereotyper utifrån den tidigare forskningen som presenteras nedan. Dessa egenskaper och stereotyper sammanfattas i metodkapitlet och ligger till grund för operationaliseringen av studiens analytiska ramverk.

2.1 Grunderna inom datalogiskt tänkande

Problemlösning är byggstenen i datalogiskt tänkande som hela begreppet utgår ifrån. Emanuelsson, Ryding och Johansson (1991) beskriver problemlösning som den processen att lösa problem utifrån givna förutsättningar eller instruktioner.

Det främsta målet med implementationen av datalogiskt tänkande är att få elever att börja använda datalogiskt tänkande för att kunna bli mer kreativa och utveckla program på ett effektivt och hållbart sätt (Government, 2014). Eleverna ska förstå principerna och ramverken som krävs för att utforma applikationer och använda sitt datalogiska tänkande för att lösa problem på ett systematiskt och smart sätt.

Datalogiskt tänkande innebär att lösa problem, designa system och förstå mänskligt beteende genom att utnyttja de grundläggande begreppen inom datavetenskap. Ett värdefullt perspektiv av datalogiskt tänkande, specifikt för gymnasieskolan, presenteras i en kurs om datalogiskt tänkande, där de fokuserar på tillämpningen av datalogiskt tänkande och är baserad på de stora idéerna inom datoranvändning (Grover & Pea, 2012).

De sju idéerna är:

1. Beräkning är en kreativ mänsklig process
2. Abstraktion reducerar information och detaljerar fokusen till de relevanta koncepten för att förstå och lösa problem.
3. Data och information underlättar tillverkandet av kunskap
4. Algoritmer är verktyg för utvecklingen och uttryckningen av lösningar till datalogiska problem.
5. Programmering är en kreativ process som producerar datalogiska artefakter.
6. Digitala enheter, system och de nätverk som sammankopplar dem möjliggör och främjar beräkningsmetoder för att lösa problem.
7. Beräkning aktiverar innovation i andra branscher, som inkluderar vetenskap, social vetenskap, konst, medicin, teknik och affärer.

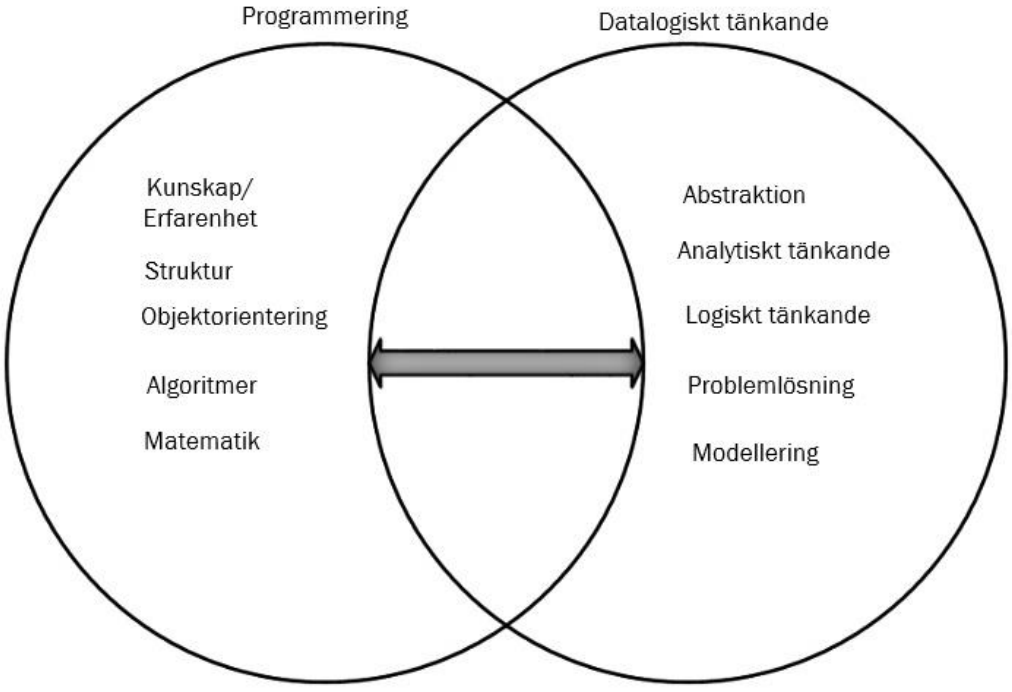
Dessa sju idéer beskriver datalogiskt tänkande på ett abstrakt och datalogiskt plan. Grover och Pea (2012) menar att beräkning är en kreativ aktivitet för individer och underlättas av abstraktion för att fokusera på de koncept som är relevanta. Det hjälper att förstå och lösa problem. Data och information underlättar att skapa kunskap inom området. Algoritmer är verktyg som gör det möjligt att utveckla lösningar vid datorbaserade problem. Programmering är den process som producerar artefakter och möjliggör innovation även i andra områden än just programmering. Det öppnar upp ett nytt tankesätt att reflektera över problem och

problemlösning för andra vardagliga problem och kan vara en god kunskap att ha menar Grover och Pea (2012).

Forskare arbetar mycket med beskrivningar av datalogiskt tänkande och värdet i abstraktion som hörnstenen inom datalogiskt tänkande och som särskiljer den från andra typer av tänkande är obestridd (Grover & Pea, 2012; Wing, 2008). Dessa element som presenteras nedan är numera brett accepterade att de innefattas i datalogiskt tänkande och som skapar basen som t.ex. kurser är baserade på och som stöttar inlärningen och utvecklingen:

• Abstraktion och generalisering av mönster(inklusive modeller och simulationer).
• Systematisk process av information.
• Symboliska system och representationer.
• Algoritmiska begrepp av flöde av kontroll.
• Nedbrytning av strukturerade problem (modellering).
• Iterativt, rekursivt och parallell-tänkande.
• Villkorlig logik
• Effektivitet och prestanda krav.
• Felsökning och systematiskt fel-hantering.

Programmering är inte bara en viktig egenskap inom datavetenskap generellt sett utan också ett verktyg för att stödja de kognitiva uppgifterna inom datalogiskt tänkande. Olika prestationer har gjorts som t.ex. CS Unplugged (2017) som introducerar dator koncept utan användningen av en dator, medan de använder sig av aktiviteter för att introducera barn till datavetenskap-området, dock kan dessa saker hålla dem borta från viktiga datalogiska erfarenheter som datalogiskt tänkande faktiskt erbjuder (Grover & Pea, 2012).



Figur 2.1 Datalogiskt tänkande & programmering. Baserad på: (Sneider et al, 2013)

Figur 2.1 beskriver förhållandet med datalogiskt tänkande och programmering samt vilka faktorer den bygger på. Sneider et al (2014) beskriver de variabler som utgör datalogiskt tänkande och ett matematiskt tänkande. I denna studie har denna modell avgränsats och utgår från det mer abstrakta perspektivet av datalogiska tänkandet och tar inte med som hänsyn till t.ex. spel och robotik. Sneider et al (2014) menar dock att faktorerna från det matematiska tänkandet tillsammans med det datalogiska tänkandet framkallar de egenskaper som är attraktiva inom IT. Problemlösning, modellering, analysering och statistik/sannolikhet är flera av de variabler som undersökts i denna studie.

Sneider et al (2012) menar att med mer datoranvändning och hur mer individerna lär sig om möjligheterna och begränsningarna kommer de ha en ökad förståelse av forskningsinriktade problem eller projekt som är beroende av datorer. Sneider et al (2012) har forskat kring datalogiskt tänkandet inom skolan och deras studie har visat att i vissa fall kommer det ökade datalogiska tänkandet leda till en motivering till studenter till att vilja studera inom STEM (sv. Vetenskap, Teknologi, Ingenjörsvetenskap, Matematik) (2017).

2.2 Koncept och tillvägagångssätt

Datalogiskt tänkande är samlingen av koncept och tillvägagångssätt för att lösa ett problem. För att beskriva helheten av datalogiskt tänkande delas datalogiskt tänkande in som två områden, nämligen koncept och tillvägagångssätt (Barefootcas, 2014).

Koncepten innebär att kunna se logiken i problemet och det logiska sättet att utföra uppgiften (Barefootcas, 2014). Algoritmer är ett koncept som används både hos individen och hos datorn för att utföra arbetsuppgifter. Algoritmen kan ses som en plan för hur lösningen ska utformas. Förmågan att se mönster i en logik eller helheten är viktigt eftersom förmågan att kunna upptäcka dem förenklar helheten. Utvärdering är koncepten som utgör grunden för alla koncept då individen måste kunna utvärdera problemet och lösningen för att hitta det effektivaste och optimalaste tillvägagångssättet (Barefootcas, 2014).

Vid användningen av koncepten skall problemlösning utnyttjas. Själva skapandet av lösningen görs av diverse tillvägagångssätt, exempelvis kodning av lösningen, vilket kräver en kunskap av det programspråk som används t.ex. C#, Java, C++ m.fl. En stor del av skapandet av en applikation eller problemlösningen är felsökning. I programmering uppstår det oftast många problem som är oförutsedda vilket kräver en felsökning och är en stor del av lärandet av både programmering och datalogiskt tänkande då det handlar om att analysera utfallen av problemet. Datorer och maskiner är maskinära, vilket betyder att de behöver tydliga och binära lösningar eftersom maskinen inte själv kan utforma sitt eget tänkande. Källkoden i ett program skrivs om av en kompilator till det binära språket som maskiner förstår och kan hantera. En kompilator som används är t.ex. Common Language Runtime (CLR) (Microsoft, 2015). Felsökningen kräver därför att individen ska kunna agera och tänka som en dator d.v.s. datalogiskt tänkande (Wing, 2008).

En viktig del inom datavetenskap är automationen av abstraktioner, således blir det också en vital del inom datalogiskt tänkande. Abstraktioner används för att definiera mönster, t.ex. att generalisera från specifika fall som det har uppstått i och det används också parametern som är aktiviteten att hitta parametrar för att ha möjlighet att göra en komplett specifikation av en modell eller ett objekt. Abstraktion används för att låta ett objekt stå för många, och att extrahera de grundläggande egenskaperna hos objekt och dölja det överflödiga eller

irrelevanta inom dem. Ett exempel på en abstraktion är en algoritm som är en abstraktion av en process som tar en input, exekverar ett antal steg, som sedan ger en output som skall uppfylla ett visst mål eller ett syfte (Wing, 2008).

2.3 Abstraktion

Ett viktigt begrepp som används inom datalogiskt tänkande är *abstraktion.* Abstraktion innebär att man frigör sig från konkreta detaljer och identifierar vilka detaljer som är viktiga och vilka detaljer som kan ignoreras. Det ligger i grund för datalogiskt tänkande (Wing, 2008). Det görs genom att skapa abstraktion-lager och sedan identifiera förhållanden mellan lagren. Dessa lager är våra mentala ”verktyg” för att sedan använda fysiska verktyg för att utföra aktionerna. Fysiska verktyg behöver inte betyda datorer utan kan likaväl betyda människor. Människor behandlar information vilket gör dem till en slags dator (Wing, 2008). Det är samarbetet mellan en dator och människa som skapar hög effektivitet.

– “Abstraktion är en process där vi identifierar de viktigaste aspekterna av ett fenomen och ignorerar detaljerna” (Ross, Goodenough & Irvine 1975, s17).

- ”Missförståndet av abstraktion är att det används som både en process och en entitet. Abstraktion som en process, aktiverar extraktionen av de viktigaste detaljerna av ett objekt, eller en grupp av objekt samtidigt som de mindre viktiga detaljerna ignoreras. Abstraktion som en entitet, aktiverar en modell, en vy eller en annan representation av ett verkligt objekt. Abstraktion är mest använd som en komplexitets-teknik. T.ex. hör vi ofta människor som säger saker som: ”ge mig bara höjdpunkterna” eller ”bara fakta tack”. Vad de personerna egentligen menar är abstraktioner” (Booch 1991, s521).

Abstraktion i det datalogiska tänkandet fokuserar främst på det Ross, Goodenough och Irvine (1975) beskriver, just att ha förmågan att identifiera det viktiga i fenomenet och att ignorera de detaljer som inte har en betydelsefull påverkan på processen. Booch (1991) beskriver abstraktion som en process men även som en entitet vilket kan relateras till programmering. Processen beskrivs ur det datalogiska tänkandets perspektiv och just att utesluta det irrelevanta som efterfrågas. När abstraktion beskrivs som en entitet relateras det till programmering då abstrakta klasser används för att beskriva ett objekt för t.ex. användaren men även exkludera det oväsentliga.

Ett vanligt objekt-orienterat begrepp som ofta missförstås och relateras till abstraktion är inkapsling. Berard (2006) menar att abstraktion och inkapsling är olika tekniker men högt relaterade. Inkapsling är tekniken att packetera information för att klargöra vilken information som är menad att vara synlig eller gömd (Snyder, 1986).

Abstraktion ger oss möjligheten att hantera komplexitet. Att arbeta med abstraktion ger möjligheten att arbeta med stor komplexitet utan att behöva hantera överflödig eller onödig information. Inom t.ex. programmering används abstraktion då en mjukvaru-utvecklare inte behöver tänka på underliggande teknik som t.ex. nätverk eller hårdvara utan de endast fokuserar på kodning vilket underlättar arbetet (Wing, 2008).

De abstraktioner som görs inom datalogiskt tänkande är oftast mer innehållsrik och komplexa än de som görs inom matematik och fysik. Det datalogiska tänkandets abstraktion har oftast inte nytta av t.ex. algebraiska egenskaper av matematiska abstraktioner som exempelvis

nummer av den fysiska världen. Wing (2008) menar att när det arbetas med sådana abstraktioner är det kritiskt att definiera ”rätt” abstraktion. Abstraktionsprocessen, som innebär att det bestäms vilka detaljer som måste understrykas och vilka detaljer som kan ignoreras, ligger till grund för datalogiskt tänkande.

Den abstraktionsprocess som beskrivs innebär att lager introduceras. Inom programmering arbetas det med minst två, men oftast fler, abstraktionslager. Välgjorda och definierade gränssnitt är ett viktigt steg i att bygga stora och komplexa system. T.ex. en användare ska inte behöva veta detaljerna om komponentens implementation för att kunna använda det. Det grundläggande inom datalogiskt tänkande och abstraktion är att definiera abstraktioner, att arbeta med flera abstraktionslager och förstå relationerna mellan de olika lagren. Abstraktioner är de mentala verktyg av datoranvändning. Styrkan av dessa verktyg förstärks med våra fysiska verktyg, som är datorer eller andra mekaniska verktyg. De tekniska verktygen är automatiseringen av dessa abstraktioner (Wing, 2008). När arbetet med lager av abstraktioner utförs behövs oftast relationerna mellan lagren sättas i fokus. I programmering och utveckling kan det definieras i funktioner och moduler men i det datalogiska tänkandet är det psykologiskt och ett logiskt tankesätt som styr lager-abstraktionen (Wing, 2008).

2.4 Analytiskt tänkande

Datalogiskt tänkande kan ses som ett slags analytiskt tänkande. Det är en integrering av tre andra typer av tänk, det matematiska, tekniska och vetenskapliga. Det finns likheter mellan det matematiska tankesättet och det datalogiska tankesättet, framförallt hur människor hanterar problemlösning. Det tekniska tankesättet innehåller likheter om hur människor designar och utvärderar ett komplext system utifrån kunskapen som erhållits. Slutligen innehåller datalogiskt tänkande en del av det vetenskapliga tankesättet som består av intelligens och mänsklig interaktion (Wing, 2008). För att sätta det i kontext kan exempel på datalogiskt tänkande vara att laga mat utifrån ett recept eller bygga ett bord utifrån givna instruktioner (Henderson, 2009).

Datalogiskt tänkande är som tidigare nämnt ett slags analytiskt tänkande där definitionen är att det är en problemlösningsprocess för att beskriva, analysera och lösa problem för att kunna ta hjälp av datorer. Det är då aktiviteten att formulera ett problem för att sedan finna lösningen med hjälp av datorer (Wang, 2016). Det går att använda sig av datalogiskt tänkande även om inte en dator används utan en människa kan utföra samma arbete. Datalogiskt tänkande är ett samlingsord för att bryta ner problem i mindre hanterbara problem som tillsammans utgör lösningen för helheten.

2.5 Programmeringskunskaper

Beskrivningen av de mest effektiva programspråken är en komplicerad uppgift. Det beror på många olika faktorer som t.ex. personlig erfarenheter eller egna uppfattningar om programspråk. Dock finns flera studier gjorda om de programspråk som har den mest effektiva prestandan vilket presenteras nedan. I den här studien används endast back-end (sv. Maskin-nära) programspråk. Fourment och Gillings (2008) beskriver de mest använda och effektivaste språken som följande; *C#, C++, C, Java, Perl* och *Python*.

Samtliga programspråk förutom C är objekt-orienterade programspråk. Med objekt-orienterade programspråk menas att språkets ramverk baseras på användning utav objektorienterade principer och objekt (Microsoft, 2010). Objekt är en självständig entitet/modell som kan användas för att visualisera och representera verkligheten. Ett exempel är ett objekt som heter ”Hund”. Hunden har i verkligheten en rad egenskaper som t.ex. ålder, kön och färg. Dessa kan visualiseras och representeras i objekt som attribut av objektet. Objekt kan tilldelas värden och agera annorlunda beroende på vilka parametrar/argument som den tar emot och skall returnera.

Några generella egenskaper som är grundprincipen för ett objekt-orienterat programspråk är: inkapsling, arv, polymorfism och dynamiska bindningar (Berard, 2006; Microsoft, 2010).
Som tidigare beskrivit är inkapsling strukturen på ett objekt för att dölja/visa information från andra objekt. Inkapsling är nära relaterat till abstraktion som finns inom objekt-orienterade(OO)-språk men även i datalogiskt tänkande. Arv är förmågan av objekt att ärva egenskaper och metoder från ett annat objekt (Microsoft, 2010). Polymorfism är förmågan att returnera och behandla objekt olika beroende på vilken in-parameter som tas emot. Dynamisk bindning menas när ett objekt anropas utav en metod eller ett annat objekt, är det inte starkt associerat med sitt ursprungliga objekt utan är dynamiskt bundet (Craig, 2008).

Algoritmer är en viktig process i programmering (Blass & Gurevic, 2003). Algoritmer menas som vägen att lösa ett problem, steg för steg till att det går att utföra uppgiften och lösa problemet. Algoritmer ger inte lösningen utan den måste även exekveras och köras i programmet men det är den som gör det möjligt att lösa problemet. Datalogiskt tänkande är starkt kopplat till algoritmer där problem bryts ner och delproblemen löses steg för steg (Wing, 2014).

Samtliga programspråk vid programmering utgår dem oftast ur problemlösning (Luhandjula & Rangoaga, 2014). Det finns en funktion att implementera eller ett system att utveckla. Alla delar/funktioner i systemet skall implementeras och kan ses som delproblem för att lösa det stora problemet d.v.s. systemet.

2.6 Datalogiskt tänkande vid programmering

Wang (2016) har utformat en mall för vilka egenskaper inom datalogiskt tänkande som testas vid ett specifikt fall inom ett programmeringsscenario. Han går igenom många olika fall och visar ett brett perspektiv av datalogiskt tänkande och vad det faktiskt innebär och vilken effekt det har på arbetet. Till exempel när ett flödesschema startar och ska skapas, då hjälper det datalogiska tänkandet till med att vara redo på oförutsägbara problem när uppgifter utförs, och att göra uppgifter i rätt ordning och att säkerställa att allt är korrekt innan arbetet tas vidare in i nästa fas eller uppgift.

Wang (2016) beskriver dessa kategorier som faktorer som beskrivs nedan. Wang (2016) menar att ett problem löses genom att bryta ner det i mindre delproblem, och varje delproblem kan brytas ner på samma sätt och till slut är problemen lätta att lösa vilket gör att arbetet rör sig framåt.

Flera viktiga perspektiv av datalogiskt tänkande skildras i artikeln från Wang (2016) och var det faktiskt kan utnyttjas på det mest effektiva sättet. Till exempel när tabeller ska användas och det skall effektiviseras, då hjälper det datalogiska tänkandet till med att utvärdera olika

alternativ. Ingen funktion i ett system skall ses som fast utan tankegångarna skall snarare vara: vad händer om det är ändrat? Det är då tankegångarna blir flexiblare och det skapas en djupare förståelse och kanske tillochmed finner en lösning som inte har visat sig tidigare.

Även inom abstraktion finns det egenskaper i artikeln som är upplysta, och som förklarar vid vilket tillfälle det optimalt bör användas. Vid just det här segmentet argumenterar författaren på detaljer och att det är viktigt att lägga tid på det (Wang, 2016). Tidigare i texten beskrivs vikten av en abstraktion och att det består av att söka ut viktiga delar och ignorera irrelevanta delar. Men det betyder inte att viktiga delar inte kan vara detaljer, vilket gör abstraktion som en svår egenskap att skaffa sig då personen bör ha både detaljer och helheten i åtanke när den använder sig av abstraktion på ett visst område eller ett visst program. Wang (2016) förstärker detta i citatet nedan.

– ”Det är de små, små detaljerna som gör det” (Wang, S, P 2016, s76).

Analytiskt tänkande återfinns även i Wang (2016) där det beskrivs att de skall skapa en kraftig cykel. Med det menar dem att de skall ta fördel av ny kunskap, nya verktyg och applicera dem överallt (Wang, 2016). Med ett välutvecklat analytiskt tänkande kan de använda sig av nya kunskaper och verktyg på områden som det inte har använts på tidigare för att få ut saker och ting som de inte såg tidigare.

3 Forskningsansats och metod

Recker (2013) menar att en induktiv forskningsansats involverar att basera generella slutsatser ifrån specifik insamlad data. I denna studie används en induktiv ansats eftersom slutsatser baseras på det data som samlades in. Eftersom det finns få eller ingen tidigare kategorisering av datalogiskt tänkande, framställdes egna kategorier utifrån tidigare forskning och därmed konstruerades ett eget analytiskt ramverk som jämfördes med den insamlade empirin.

I den här studien har en *mixad metod* (eng. Mixed Method) använts, vilket innebär en kombination av en kvalitativ och en kvantitativ metod. En kvalitativ ansats fokuserar på att samla in kvalitativ data i ett visst sammanhang, och en kvantitativ ansats fokuserar på mätbar data för att effektivt kunna representera och generalisera den på ett tydligt sätt (Recker, 2013). Anledningen till att en blandad metod används i det här arbetet är för att expandera forskningsområdet och för att använda båda insamlingskomponenterna för att samla in mer data, t.ex. med hjälp av både intervjuer och enkäter, vilket Recker (2013) rekommenderar om en blandad metod skall användas. Den kvantitativa ansatsen har även använts för att verifiera och styrka data som erhållits från den kvalitativa delen av studien. Den kvalitativa metod som använts är semi-strukturerade intervjuer, vilka valdes för att kunna ställa följdfrågor samt anpassa frågorna efter informantens eventuella position i frågan. Den kvantitativa metoden som användes var en enkät där generella frågor om ämnet ställdes för att tydligare kunna klargöra resultatet och jämföra det med det kvalitativa resultatet i en större utsträckning. Metoder och verktyg kommer att beskrivas och förklaras utförligare nedan.

3.1 Urval av informanter och respondenter

Urvalet av målgrupp som undersökts i studien är verksamheter från stora till små som har en avdelning tillägnad för informationsteknologi och utveckling. Enkäter och intervjuer har genomförts med informanter som antingen är seniora mjukvaru-utvecklare eller rekryterare av mjukvaru-utvecklare som har god insikt i programutveckling. Eftersom fokus i studien är på rekryteringen av programmerare, var det nödvändigt att de seniora programmerare som antingen besvarat enkäten eller varit delaktiga i intervjun, hade någon delaktighet i rekryteringsprocessen hos verksamheten.

Enkäten skickades till *högre* chefer eller ansvariga inom en verksamhet, som de själva besvarar eller vidarebefordrar till de respondenter som är relevanta för enkäten.

Informant	Roll	Företag	Tidsåtgång
1	Samordnare för utvecklare/rekryterare	700 antal anställda	24 min
2	Ansvarig utvecklare/rekryterare	10 antal anställda	35 min
3	Ansvarig utvecklare/projektledare	10 antal anställda	37 min
4	Ansvarig utvecklare	2 antal anställda	27 min
5	Rekryterare av utvecklare	170 antal anställda	26 min
6	Rekryterare av utvecklare	7000 antal anställda	28 min

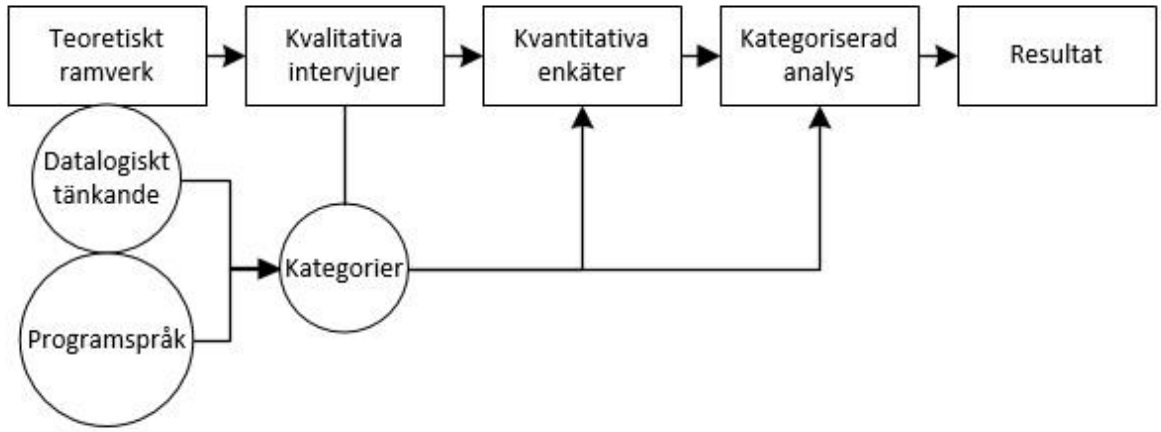
Tabell 3.1 Företagens storlek

Tabell 3.1 beskriver informantens arbetsroll, storlek på företag där informanten var anställd samt tidsåtgången för respektive intervju. Som tidigare beskrivit är företagsstorlekarna medvetet valda för att kunna identifiera om företagsstorleken har en påverkan på dess åsikt inom de konstruerade kategorierna. Uppdelning av tre företag av mindre storlek och tre företag av större storlek ger tillräcklig information för att identifiera mönster och hur storlek påverkar åsikterna.

De två informanter som inte hade rollen som rekryterare har dock en insikt i eller en åsikt angående processen av rekrytering av utvecklare. Informant 4 som endast har två anställda hade dock ingen kontinuerlig rekryteringsprocess utan reflekterade mer över informantens tidigare erfarenhet och önskemål om utvecklare men informantens åsikt är ändå högst relevant för studien eftersom arbetserfarenheten var över 40 år i utvecklingsbranschen.

3.2 Design av studien

Informationsinsamlingen bestod av två olika delar, en del av enkäter och en del som består av semistrukturerade intervjuer. Studiens övergripande design illustreras i figur 3.2.



Figur 3.2 Studiens design

Det teoretiska ramverket påbörjades som en förstudie där teori om ämnet studerades för att ge en bättre inblick och professionell utgångspunkt om datalogiskt tänkande. Ur det teoretiska ramverket definierades två områden, *datalogiskt tänkande* och *programmeringskunskaper*, samt deras kategorier som tillhör området.

Den första delen består av semistrukturerade intervjuer, där utgångspunkten var de två områdena (Datalogiskt tänkande och programmeringskunskap) med respektive konstruerade kategorier. Det konstruerades frågor som berör dessa områden som sedan använts i jämförelsen i resultatdelen av studien.

Den andra delen av informationsinsamlingen består av en enkät där ett tydligt resultat representeras som svarar på simpla frågor och som skall skapa en generell uppfattning om kunskapen eller åsikter om frågor som vi har inkluderat. Kvantitativ undersökning brukar oftast generera ett nummerbaserat resultat som är en kraftfull representation på hur ett fenomen fungerar (Recker, 2013). Det viktigaste inom en kvantitativ studie är mätningen, det

är det viktigaste verktyget för att resultatet ska bli mätbart men även att resultatet skall vara trovärdigt och överförbart.

Resultatet på den andra delen har samlats in med hjälp av en enkät som skickas ut via e-mail till företag som agerar inom den informations-teknologiska branschen. Med frågor som är enkelt att sammanställa och genererar ett tydligt resultat skapas ett tydligt underlag. Det underlaget har sedan kombinerats med det djupare materialet från den andra delen av undersökningen som analyseras och används för att förklara det empiriska materialet som enkäten erhåller.

Intervjuer utgjorde en central del i studien och det var det mest effektiva verktyget för att få fram det underlag och information som behövts. Det är en metod som kan ge resultat om strukturen på den är utformad på rätt sätt. Då strukturen var en någorlunda fri och semi-strukturerad intervju samt att den utgjordes av konkreta och tydliga frågor gjorde att informationsinsamlingen gav ett bra resultat. En blandning av kvalitativ och kvantitativ ansats gav viktig information som både var djup och bred men även mätbar (Recker, 2013). Antal potentiella informanter var inte förbestämt innan studien utan det bestämdes när en avgränsning gjorts och den nödvändiga informationen var insamlad.

Intervjuerna har varit strukturerade som upptäckande (exploratory*)* och beskrivande *(*descriptive), och har genomförts ansikte-mot-ansikte (face-to-face*)* (Recker, 2013). En utforskande intervju används för att definiera frågor och för att utveckla nya teorier, vilket är målet med studien. Men intervjuerna kommer även att vara beskrivande, då det krävs att det skapas en rik förklaring av fenomenen och hur de uppfattas av individer. Vilket skapar en generell uppfattning som sedan hjälper till att förstå fenomenet på ett bättre sätt och skapar grund för analysen. Fältarbetet genomfördes på informanternas arbetsplatser. En av de kännetecken som en kvalitativ metod medför är att intervjun görs i dess naturliga miljö, i det här fallet på deras arbetsplats (Recker, 2013). Det var den intervjuplatsen som gav mest pålitlig information.

Semistrukturerade intervjuer genomfördes först för att samla in empiri med utgångspunkt ur de kategorier som framtagits ur det teoretiska ramverket. Enkäter skickades sedan ut för att verifiera och identifiera mönster den insamlade datan från semistrukturerade intervjuerna.

Recker (2013) rekommenderar dock att faserna genomförs på ett annorlunda sätt. Recker (2013) menar att enkäter bör genomföras först och sedan jämföra det med kvalitativa intervjuer för att validera enkätsresultatet. Den här studien använder sig istället av att först genomföra kvalitativa intervjuer för att sedan validera intervjuerna med hjälp av enkätssvaren, eftersom de har en lägre trovärdighet då det är svårt att styra vilka som faktiskt besvarar enkäterna, då de enkelt kan vidarebefordra enkäten till *fel* person inom en verksamhet.

När empirin var insamlad och färdigställd genomfördes en analys som har en utgångspunkt i de kategorier som framtagits ur det teoretiska ramverket. Resultatet av studien har då grundats sig i det teoretiska ramverket och utökats med den empiri-insamling och den kategoriserade analysen.

3.3 Datainsamling

3.3.1 Intervju

Vid utformningen av intervjun följde och utformades arbetet efter intervjuforskningens sju stadier av Kvale (2014). De sju stadierna är Tematisering, Planering, Intervju, utskrift, Analys, Verifiering och Rapportering.

Tematiseringen är att formulera undersökningens syfte och att göra en beskrivning av ämnet. Här definieras *varför* och *vad*. Planeringen är att planera för samtliga sju stadier av undersökningen. Intervjustadiet är när intervjuer genomförs enligt en intervjuguide och frågor. Utskrift innebär en överföring från talspråk till skriftspråk. Analysstadiet består av att välja analysmetod utifrån syftet och ämnet. Verifiering är att intervjuernas reliabilitet och validitet och till sist Rapporteringsstadiet där resultatet av analysen rapporteras och leder till en läsbar produkt (Kvale, 2014).

Strukturen var en semistrukturerad forskningsintervju, och användningen av metoden är som Kvale (2014) beskriver, ett mänskligt samspel där kunskap utvecklas genom en dialog. När intervjun tog plats, var det viktigt att den intervjuade skapade tillit till intervjuaren och kände sig trygg och kunde tala fritt om sina känslor och åsikter. Där låg det stort ansvar på intervjuaren att berätta tydligt om ämnet men också att skapa en atmosfär där svaren blir mer än en artig konversation eller utbyte av åsikter (Kvale, 2014).

Vid början på intervjun introducerades arbetet och författarna. Därefter fick informanten berätta om sig själv, vilket gjorde att informanten kände sig mer bekväm och berättade mer om erfarenheterna. När informanten förstod arbetets mål och struktur, blev det lättare för individen att veta vilken information som krävdes och vilken information som inte var nödvändig. Om inte arbetet förklaras, kan det samlas in information om fel saker och inte ha tillräckligt med information för att utföra en analys som grundas på konkret och en stor del information.

Det finns flera olika typer av intervjuer, och de kan se väldigt olika ut i t.ex. aspekterna, strukturen och öppenheten. Intervjun strukturerades som en semistrukturerad, där valet togs att berätta syftet med intervjun i början av intervjun. Anledningen till det var för att ämnet datalogiskt tänkande är relativt nytt och då även dess innehållande begrepp. Förklaring av dem var viktigt för att de skulle förstå syftet med studien och ge information utifrån det. Öppna frågor med mycket utrymme gavs, på så sätt kan individen tala fritt och sedan låta intervjuarna styra personen i rätt riktning med färdiga frågor. Med stort fokus på följdfrågor men också en checklista som användes på intervjun, där det säkerhetsställs att tillräcklig information samlas in inom respektive kategori för att göra en analys och sedan ett resultat.

Eftersom en kategoriserad analys har gjorts utifrån empirin har inte informanterna varit informerade under intervjun vad kategorierna är och därför ställdes frågorna implicit utifrån kategorierna i det konstruerade analys- och begreppsramverk.

De analysmetoder som använt är *meningskoncentrat* och *kategorisering* för att analysera den data som samlats in. Den analysmetod som valdes var viktig för att den valda metoden kommer att styra sammanställningen av intervjuguiden, intervjuprocessen och utskriften (Kvale, 2014). Intervjuerna spelades in och transkriberades utan någon modifiering. Med hjälp av de två analysmetoderna meningskoncentrat och kategorisering sammanställdes empirin i resultatkapitlet. Primärt användes meningskoncentrering på intervjuerna där meningar förkortades och endast det väsentliga tas med och omformuleras (Kvale, 2014). Därefter användes kategorisering, informationen från meningskoncentratet analyserades och

kategoriserades utifrån analys- och begreppsramverket som förklaras och visualiseras senare i texten.

Meningskoncentrering betyder att meningar som uttryckts av informanten formuleras mer koncist (Kvale, 2014). Långa meningar pressas samman, och endast det väsentliga ur samtliga uttalanden tas med och omformuleras. Meningskategorisering innebär att intervjun delas in i kategorier, som till exempel om ett fenomen förekommer eller inte. Då intervjuerna grundas på ett antal kategorier, ses det förmånligt att dela in intervjun i sagda kategorier för att sedan analysera uttalandena del för del. Sedan finns det en annan form av intervjuanalys där det används ad hoc, där materialet läses igenom och skapar en generell uppfattning om intervjun och sedan gå tillbaka till specifika avsnitt och implementera den metoden som det avsnittet kräver (Kvale, 2014).

3.3.2 Enkät

För att realisera enkäten och presentera data har Webbenkäter (2017) och Infogram (2017) använts.

Efter att den kvalitativa intervjun gjorts har även en kvantitativ metod använts genom att skapa enkäter och skicka ut till individer med samma bakgrund som informanterna, mjukvaru-utvecklare och rekryterare. Anledning till det valet är att information behövs från ett stort antal individer med direkta frågor för att ytterligare styrka intervjuanalysen och finna samband vilket förstärker reliabiliteten med forskningsuppsatsen som en helhet. Enkätmaterialet sammanställs med hjälp av statistiska tekniker och andra kvantitativa tillvägagångssätt (Recker, 2013).

Den här metoden användes för utforskningskontext och svarar på frågor som varför och hur. Syftet med enkäten är att göra en *upptäckande* enkät då studien kräver både kvalitativ empiri men även en bredare och generell empiri där ett bredare och applicerbart resultat kan uppnås (Recker, 2013). Mätbar data var viktigt att samla in, som är enkel att jämföra med den kvalitativa empirin som är insamlad. Enkäten bidrar även till överförbarheten till framtida forskare med hjälp av den mätbara datan.

En del frågor i enkäten var enklare ja eller nej-frågor då en del information var nödvändig att veta för att kunna utvärdera resultatet i enkäten. Som Bryman och Bell (2011) beskriver, används stängda frågor med flera svarsalternativ i skalor. Detta för att inte bli instängda i ja eller nej-frågor och att kunna utvärdera och analysera svaren från varje besvarad enkät. Men även att kunna sammanställa och se potentiella kopplingar mellan svar från respondenterna. Nackdelar med denna typ av undersökning är som Bryman och Bell (2011) menar att det finns rum för misstolkningar av svar. Därför formulerades frågorna noggrant för att eliminera misstolkningar och för att få en trovärdig empiri.

Vissa frågor i enkäten har dock utelämnats som en öppen fråga för att respondenten kan svara hur de vill eftersom det är en utvärdering av datalogiskt tänkande beroende på individen. Det finns flera olika insikter om hur mycket individen reflekterar över datalogiskt tänkande och därför togs beslutet att inte ha slutna svar utan öppna kommentarsfält. Dessa svar sammanställs senare med hjälp av *kodning* (Recker, 2013) som redovisas i resultatkapitlet. Kodning användes genom att svaren analyserades och nyckelord kunde extraheras.

Eftersom enkäten var utformad enligt en kvantitativ och kvalitativ design ses enkäten som en blandning av kvalitativ och kvantitativ empirisk undersökning. Detta för att en del kategorier inte kan besvaras med mätbar data utan kräver ett kvalitativt svar.

3.4 Studiens grundläggande analys- och begreppsramverk

I den här studien har datalogiskt tänkande och programmeringskunskap kategoriserats till sju kategorier, som är beskriva nedan. Dessa kategorier har konstruerats och sorterats utifrån den insamlade teorin i det teoretiska ramverket. Kategorierna användes för att sortera in testfall och litteratur mot vilken kategori de igenkänns i för att skapa en tydligare och bättre analys. Stereotyper har även konstruerats som innefattar några av de existerande kategorierna. Detta illustreras i tabell 3.3.

Kategorier / Stereotyper	Kunskap/Erfarenhet om språk	Strukturerat arbetssätt	Objektorienterad ansats	Helhetslösningar	Analytiskt/logiskt tänkande	Abstraktion	Problemlösning
Datalogiskt tänkande					X	X	X
Programmering	X	X	X				X
Mix	X	X	X	X	X	X	X

Tabell 3.3 Analys- och begreppsramverk

3.4.1 Kategorier

Kategorierna har tagits fram från den tidigare forskningen kring datalogiskt tänkande och programmering. En del av kategorierna har definierats själva av författarna för att tydliggöra användandet av kategorierna i den här studien. De kategorier som används ansågs vara de viktigaste och tydligaste för att kunna samla in empiri och besvara forskningsfrågan.

Kunskap/erfarenhet om programspråk är den kategori som handlar om den råa kunskapen om ett visst programspråk t.ex. C# och C++. Det handlar om hur individen har kunskap om syntax, regler och funktioner inom det ramverk som det används i. Det kan innefatta programspråk, ramverk, arkitekturer, tillvägagångssätt av design osv.

Strukturerat arbetssätt är en egenskap som är viktig för mjukvaru-utvecklare (Andrews, 1991). En egen definition konstrueras i den här studien. Med strukturerat arbetssätt menar vi hur en mjukvaru-utvecklare är strukturerad och planerad i sitt arbetssätt genom att planera sitt tillvägagångssätt vid programmering. Källkoden skrivs på ett sätt att andra intressenter har lätt att förstå vad allting gör. Motsatsen till strukturerad, enligt oss, är t.ex. att en utvecklare skriver källkod endast för att utföra en viss uppgift, utan att tänka på vad som händer efter. Det enda viktiga är då att få det att fungera och har ingen tanke på hur det ser ut för programmet och andra intressenter.

Den objektorienterade ansatsen bygger på principen att utveckla applikationer vars områden innefattar OO(Objektorientering), OOA(Objektorienterad analys), OOD(Objektorienterad design) och OOP(Objektorienterad programmering) (Microsoft, 2010; Microsoft, 2017; Lee, 2014). Eftersom programmeraren är i fokus i studien riktas mest principer på OOP och de interna principer inom programmering som t.ex. abstraktion, inkapsling och arv (Berard, 2006).

Helhetslösningar är ett begrepp som kan relateras till många av kategorierna. En applikation består till en stor grad av programmeringslösningar vilket löser vissa problem i applikationen. Lösningar kan dock vara begränsade till en funktion och påverka andra funktioner negativt vilket bryter mot den objektorienterade ansatsen som t.ex. låg koppling. Helhetslösningar är då begreppet där designen av lösningar är anpassade till omvärlden och de andra modulerna i programmet utan att möjligtvis förstöra eller förhindra andra funktioner från att utföra deras uppgifter.

Analytiskt eller logiskt tänkande är egenskaper där en individ analyserar situationen utifrån de möjliga utfall som kan finnas vid vissa problem. Man analyserar och konstruerar alternativa lösningar och överväger vilka beslut man bör ta för att uppnå den bästa lösningen för samtliga situationer (Wing, 2008). Analytiskt tänkande och logiskt tänkande är nära relaterat eftersom de båda grundas på den logiska strukturen på ett problem eller lösning för att finna den som är mest lämpad för situationen.

Abstraktion är ett begrepp som finns i både programmering och datalogiskt tänkande. I programmering handlar det om att dölja komplexiteten från användaren och bara visa det nödvändiga i applikationen (Berard, 2006; Hunt & Riley, 2014). I datalogiskt tänkande är abstraktion enligt Wing (2008) att individen ska fokusera på det som är viktigast i uppgiften och sätta det andra ur fokus för att få fram det bästa resultatet.

Problemlösning är ett begrepp som även här återfinns i programmering och datalogiskt tänkande. Problemlösning i programmering är egentligen grundprincipen i programmering eftersom man hela tiden löser problem. Att visa en kunds information är ett problem som löses med kod i en applikation men även att lösa eventuella fel i programmet. Problemlösning i datalogiskt tänkande enligt Wing (2008) och Grover och Pea (2012) är nedbrytning av det komplexa problemet till mindre hanterbara delar för att minska risken för komplicerad problemlösning och kunna skapa en strukturerad och planerad problemlösningsstruktur.

3.4.2 Stereotyper

De stereotyper som konstruerats är datalogiskt tänkande, programmering och mix (tabell 3.3). De stereotyperna har tagits fram utifrån ramverket och även dess kategorier som respektive stereotyp innehar.

Det datalogiska tänkandet har som grund i problemlösning på en abstrakt nivå. Stereotypen hanterar komplexa problem genom att bryta ner det till delproblem och löser dem på ett planerat och strukturerat arbetssätt som är enkelt att följa och tyda (Wing, 2008; Grover & Pea, 2012). Stereotypen fokuserar på det som är viktigast och utesluter det onödiga. Stereotypen hittar mönster i problemlösningen och arbetssätten för att kunna tydligare se hur problem och arbetsuppgifter hänger samman och den övergripande arkitekturen.

Den programmerings-baserade stereotypen har grund i programmerings principer och kunskaper. Stereotypen har kunskap om programmering generellt och programspråkets syntax och regler. Den följer de valda ramverken och den objektorienterade ansatsen återfinns i arbetet. Den har god kunskap om hur man strukturerar arbetet på ett sätt som stödjer återanvändning och effektivitet i programmeringen(Microsoft, 2010; Microsoft, 2017; Berard, 2006).

Den mix-baserade stereotypen är den som har kunskap inom samtliga variabler. Programmering stöds utifrån den problemlösning som utförs med hjälp av det datalogiska tänkandet. Det leder till att programmeringen ur ett verksamhetsperspektiv blir en effektivt och hållbar lösning som kan återanvändas och uppföljas i verksamheten utan större problem. Det logiska och analytiska tänkandet är som grund för de beslut som tas i programmering för att välja den rätta vägen att ta. Källkod som program-utvecklaren skriver är strukturerad och lättläst vilket underlättar mycket för intressenter som ska ta del av källkoden senare och som sedan underlättar förändringsbarheten inom produkten.

3.5 Validitet och reliabilitet

Som Recker (2013) beskriver är validitet en viktig faktor i en forskningsstudie. För att uppnå en hög validitet har vi som urvalet understryker endast samlat empiri från personer som antingen rekryterar mjukvaru-utvecklare eller är seniora utvecklare med flera års erfarenhet i branschen och har en åsikt om rekrytering av utvecklare.

Bryman och Bell (2011) menar att validiteten i intervjuer kan påverka resultatet eftersom det är svårt att genomföra intervjuer med exakt likadan struktur varje gång. Eftersom vi fick anpassa de semi-strukturerade intervjuerna för varje informant vi pratade med kan då potentiellt resultat utelämnas eller inte observeras. Försök gjordes att strukturera samtliga intervjuer på liknande sätt för att få ett ungefärligt likadant resultat från samtliga informanter, det gäller även enkätsundersökningen.

Företag krävde att få en kopia av studien, vilket skapar krav på författarna att skapa ett utförligt och pålitligt arbete, och även att informationen stämmer om de på något sätt kan identifiera sitt företag utifrån storleken på det. Det ställer stora krav på att informationsinsamlingen är korrekt. I och med att intervjuerna spelas in, så samlas det in exakt vad informanten sa, och kan enkelt verifiera informationen. Det skall inte finnas rum för misstolkningar, vilket ställer krav på oss som forskare att vara tydlig i beskrivningar till den grad att det inte skall finnas rum för tolkning.

3.6 Etiska aspekter

Recker (2013) menar att de etiska aspekterna är vitala för en forskningsrapport. Det får inte under några omständigheter förekomma vanskötsel av de etiska aspekterna. Angående det etiska perspektivet och integriteten hos informanter och respondenter, har valet tagits att låta dem vara anonyma i den här studien. Verksamheterna som de arbetar på är även de anonyma. Anledningen till att låta all empiri vara anonym är dels för att få ett trovärdigt resultat men även att låta informanterna berätta vad de verkligen tycker om frågan, utan att bli igenkända i studien av andra intressenter. Genom att låta informanter vara anonyma, blir resultatet trovärdigare eftersom informanterna känner sig tryggare i situationen.

Som Shamoo och Resnik (2009) påpekar finns flera etiska aspekter att reflektera över vid utförandet av forskning. I den här studien har alla aspekter som Shamoo och Resnik (2009) värderats och använts som t.ex. sekretess, integritet av informanterna och objektivitet.

Vid intervjutillfället säkerställdes det att individen accepterade att intervjun spelades in, och att det användes till arbetet. Det klargjordes i ett tidigt skede att varken namn eller företag

kommer användas, vilket gjorde att de kan öppna upp sig och inte tänka på att det kan påverka namnet eller företag på ett negativt eller positivt sätt.

Vetenskapsrådet (2017) menar att oredlighet i forskning inte får uppkomma. I den här studien har det inte medvetet gjorts några oredliga beslut som påverkat studiens resultat. För att öka pålitligheten och stödja argumentet för oredlighet uppmanas andra forskare att genomföra en liknande studie.

4 Resultat och kategoriserad analys

Sex intervjuer genomfördes för att ha möjlighet att jämföra datalogiskt tänkande och de framtagna kategorierna från programspråken. Det formuleras olika frågor om analytiskt tänkande, abstraktion och logiskt på datalogiskt tänkande och frågor om kodning och algoritmer på programspråket. Frågorna som följde var sedan jämförande frågor mellan de båda fenomenen och även beskrivande frågor som informanterna fick förklara deras åsikt om områdena och vilket de föredrog i vilken position. De viktigaste resultaten är sammanfattade nedan per kategori från tabell 3.3, följt av en övergripande sammanfattning av respektive kategori. Under intervjufasen i studien har det även observerats att personliga egenskaper är en stor del av rekryteringsprocessen och i urvalet av kandidater. Därför läggs en kategori till utanför programspråk och datalogiskt tänkande som endast fokuserar på personliga egenskaper.

4.1 Personliga egenskaper

Med personliga egenskaper avgränsades dessa till endast egenskaper till individen som människa. Detta berör alltså inte programmeringsegenskaper utan egenskaper som t.ex. social, kreativ m.fl.

En faktor som inte tagits med som design av stereotyperna men som fått hög prioritering bland samtliga informanter är den sociala aspekten av en individ. Att ha en förmåga att samarbeta och kommunicera med andra på företaget ses som mycket värdefullt och anses vara ett krav om inte programmeringskunskaperna är extremt höga vilket då kan reglera kravet om det kan ge ett bra resultat. Det fanns exempel där individer arbetar för självständigt och inte samarbetar vilket ofta resulterar i överarbetade lösningar med onödiga funktioner och liknande. Andra egenskaper som tas upp är koncentrationsförmågan, där förklaringen lyder att det är viktigt att individen fokuserar på det väsentliga, vilket har en direkt koppling till *abstraktion* som är ett grundbegrepp inom datalogiskt tänkande.

Det som nämns bland samtliga informanter är vikten av samarbete och förmågan att arbeta i grupp. Speciellt i de större företagen som arbetar endast i grupper, där prioriteras den sociala förmågan mycket högt och är i princip ett krav för att bli anställd. Finns förmågan att kommunicera och arbeta tillsammans blir det oftast en individ som gör lösningar som bara den själv förstår och försvårar samarbete med andra individer som arbetar inom samma projekt eller liknande.

Vid frågan om egenskaper som undviks, där dyker den asociala förmågan fram som ett vanligt svar. Men det dyker även upp egenskaper inom programspråken, att de ska vara villiga att lära sig nya programspråk, en informant säger att fundamentalisterna är problematiska att ha att göra med, då de inte är villiga att förändra och anpassa sig efter den verksamheten som de arbetar i. Det är en viktig förmåga för samtliga att ha viljan att lära sig och anpassa sig. Då skillnaderna mellan programspråken börjar bli mindre, blir möjligheten att lära sig andra programspråk större vilket gynnar verksamheterna och möjligheten att anställa nya individer som lätt kan lära sig det programspråk som verksamheten innehar.

Egenskaper som också är att det finns ett driv framåt, och att det tas initiativ på egen hand och att dem inte väntar på att få uppgifter utan på egen hand utvärderar på vad som behöver göras och sen försöker hitta lösningar på de problemen. Det nämns också att förmågan att ta till sig

ny kunskap, kan vara både kunskap som i att lära sig ett nytt programspråk, projektkunskap eller annan kunskap som kan kopplas till både kunskap inom programmering och kunskap inom t.ex. sociala samarbeten.

De personliga egenskaperna är viktiga för samtliga informanter på grund av den anledningen att om en individ till exempel har samarbetssvårigheter och arbetar endast på egen hand, kan den göra för mycket eller för lite. Båda scenariona innebär att verksamheten förlorar pengar som nämns ofta att det är pengar som styr. Det är därför det är viktigt med kommunikation och samarbete så att de kan säga till när de är lite fel ute eller inte är riktigt färdiga eller har gjort för mycket.

	Personliga egenskaper	Företagsstorlek
Informant 1	x	700 antal anställda
Informant 2	x	10 antal anställda
Informant 3	x	10 antal anställda
Informant 4	x	2 antal anställda
Informant 5	x	170 antal anställda
Informant 6	x	7000 antal anställda

Tabell 4.1 Prioritet av personliga egenskaper

Tabell 4.1 illustrerar informanternas prioritering angående personliga egenskaper som en kategori som efterfrågas vid en anställning. Samtliga informanter prioriterar personliga egenskaper som illustreras genom *x*.

4.2 Programmeringsegenskaper

4.2.1 Programmeringskunskap

Inom kunskapen om programspråken förespråkar samtliga för att de använder sig av att lära ut de olika programspråken. Programspråkskraven är mest preferenser eller önskemål och att finns det någon kandidat som kan de språken har de större möjlighet att bli anställd än om de inte har någon erfarenhet. Men även liknande programspråk kan vara av stort värde. En informant på ett av de större företagen nämner att objekt-orienterade programspråk skiljer sig endast i syntaxen som t.ex. C#, C++ och Java vilket de är öppna för att lära ut syntaxen men bara ifall kandidaterna innehar kunskap om den objektorienterade ansatsen. En annan informant har förståelse att vissa krav de har på tekniker och programspråk inte utbildas längre och därför måste de ställa krav på liknande programspråk som fortfarande lärs ut i utbildningar inom området.

Flera av informanterna som specificerar de programspråk och tekniker som krävs för tjänsten använder även detta som ett urvalsverktyg. Är det flera ansökningar utesluter informanterna de kandidater som inte uppfyller de önskemål som arbetsgivaren ställer på kandidaten. Men som DN (2017) skriver är det i dagens marknad en brist på programmerare vilket leder till att dessa önskemål som arbetsgivaren ställer på kandidaterna inte är helt avgörande. Flera informanter menar att de hellre testar kandidaten för att se hur bra personen är på programmering generellt utifrån ett teoretiskt perspektiv och ser en nytta i att lära upp

kandidaten i ett nytt programspråk eftersom personen kanske redan har kunskapen om programmering generellt. En informant menade att kraven i ansökningarna är deras egna *önskemål* i en drömvärld men det är ingenting man förväntar att en kandidat uppfyller. Informanten menar att det är meriterande och underlättar arbetet ifall kandidaten har kunskap om flera tekniker men absolut ingenting nödvändigt.

En av informanterna ställde inga krav på olika programspråk då individen arbetade inom Access (Microsoft, 2017), och då ställde personen större krav på problemlösningsförmågor och förmågan att hitta bästa lösningen utifrån de resurser som getts. Informanten menade då att syntax och regler är irrelevant vid rekryteringen eftersom det som är utav mest värde är kunskapen om branschen de ska agera i, och problemlösningsförmågan.

Samtliga personer nämner att syntaxer och algoritmer kommer med mer erfarenhet. 80 % av informanterna menar att syntax är det enklaste att lära sig inom programmering, men att själva problemlösningsförmågan och strukturering av kod är det som tar tid och kanske inte alltid går att lära ut. Övning ger färdighet och det gäller även inom det här området menar informanterna. Det är därför som problemlösning generellt sett prioriteras för det är sådana egenskaper som är svårare att lära sig med tiden, även om det såklart kan utvecklas men vissa individer har lättare för det än andra vilket gör det till en önskad egenskap hos en individ.

65 % av informanterna berättade att vid varje rekryteringsprocess, har dem någon typ av programmeringstest. De har programmeringstestet för att kunna utvärdera kandidatens problemlösningsförmåga, programmeringskunskap och det analytiska tänkandet. De exempel på olika test som de gav kandidaterna kan vara från att implementera en funktion till att utnyttja objektorienterade principer. En av informanterna nämnde dock att en av de viktigaste faktorerna som gör att rekryteraren överväger att anställa kandidaten är just den objektorienterade ansatsen. Är fallet att kandidaten inte har kunskap om objektorientering finns det ingen möjlighet att gå vidare med processen. Informanten menade även att det finns flera programmerare med 10 års erfarenhet som kan flera programspråk och utnyttja flera tekniker, men ändå faller på att de inte har den problemlösningsförmåga och objektorienterad ansats som de eftersöker.

Samtliga informanter som använder sig utav ett programmeringstest vid rekryteringen menar att själva utförandet av testet inte är det viktigaste. Det är uppföljningen som har mest inflytande i beslutet om kandidaten. Efter programmeringstestet får kandidaten chansen att diskutera lösningen med rekryteraren. Flera informanter menar att det är vid den tidpunkten de faktiskt får veta vad kandidat har för kunskap om programmering. De får argumentera för lösningen och förklara alternativa lösningar för rekryteraren och svara på frågor om just varför beslutet togs att göra på just det sättet.

En av informanterna menar att programmeringstest är något som de inte vill använda. Med tanke på dagens nuvarande brist på programmerare (DN, 2017) vill inte den ansvarige rekryteraren *skrämma iväg* potentiella kandidater. De menar att branschen behöver nyexaminerade utvecklare och ett programmeringstest kan möjligtvis göra att kandidaterna blir osäkra på deras kompetens. Informanten nämner dock att i framtiden är programmeringstester någonting som de vill använda sig av då det tidigare skett en del misstag gällande rekryteringen eftersom vissa kandidater inte alls hade de kunskaper som de utgav sig att ha vid rekryteringsprocessen.

Företaget med mest antal anställda använde sig inte av ett programmeringstest. De baserade urvalet av kandidater baserat på inskickat CV med de tidigare kompetenser och personlighetstester. Där är det indelat i ett personlighetstest för att se vilken typ av person man är, och ett logiskt test som utgår från att hitta logiska lösningar på ett problem. Informanten beskriver dock att programmeringstest har gynnat företagen och skulle rekommendera sitt nuvarande arbete att faktiskt använda sig av det.

Informanten antyder på att programmeringsbranschen har ett stort problem gällande rekryteringen eftersom det finns många olika programspråk och tekniker. Det är väldigt svårt att hitta en kandidat som är riktigt duktig på två eller flera tekniker. Det är oftast att man är duktig på en teknik och mindre duktig på de andra tekniker som eftersöks menar informanten. Det påpekas också att mjukvaru-utvecklare är ofta indelade i front-end/back-end där språk och tekniker blir uppdelade och man helt enkelt får specialisera sig på ett tillvägagångssätt. Det dem värderar högst är just kunskapen om programspråken men även det analytiska tänkandet och problemlösningsförmågan menar informanten. De programmeringskrav som ställs i ansökningarna är inte helt nödvändiga. Är individen en duktig utvecklare ska ett byte av programspråk eller lära sig nya tekniker inte vara något problem.

	Kunskap/Erfarenhet om språk	Företagsstorlek
Informant 1	x	700 antal anställda
Informant 2	x	10 antal anställda
Informant 3		10 antal anställda
Informant 4	x	2 antal anställda
Informant 5	x	170 antal anställda
Informant 6	x	7000 antal anställda

Tabell 4.2 Prioritering av kunskap/erfarenhet av programspråk

Sammanfattningsvis, som tabell 4.2 förtydligar hade samtliga informanter, förutom en av de små företagen, samma perspektiv angående kunskaperna och erfarenheten från det programspråk som eftersöktes.

4.2.2 Objektorienterad ansats

Flera av informanterna antydde att objektorientering är en väldigt viktig faktor inom programmering och kan ses som ett tankesätt som en individ har. Flera informanter berättade att det nästan är omöjligt att klara arbetsuppgifterna om de principerna inte följs. En informant menade att det finns knappt någonting kvar som inte är objektorienterat längre och det är därför objektorientering ofta är ett krav i ansökningar om att den egenskapen ska behärskas. Informanterna menade även att det är objektorienteringen som är det viktigaste och syntax/algoritmer är någonting som kan läras med tiden.

66 % av informanterna är eniga vid att den objektorienterade ansatsen är ett krav som måste uppfyllas av varje kandidat. De menar att det inte finns någon annan väg eller metodik att ta när digitaliseringen har nått den nivå som den är på idag. De flesta programspråk som används i branschen är objektorienterade, framförallt i applikationsutveckling menar

informanterna. Samtliga informanter menar dock att objektorienteringen inte är ett formellt krav i en ansökan. Det brukar oftast ingå indirekt i annonsen under programspråket som eftersöks.

En informant pratade specifikt om objektorienterad ansats som en egenskap som måste finnas hos de mjukvaru-utvecklare som blir anställda hos deras företag. Har inte kandidaten den grundläggande teorin och kunskapen om objektorientering är den inget alternativ för dem. Informanten antyder att syntax och regler faktorer som går att lära sig snabbt med tiden men den objektorienterade ansatsen är något som kräver mycket tid eftersom det är ett ramverk att arbeta utifrån hela tiden.

Nästa kategori som presenteras är strukturerat arbetssätt vilket på en grad sammanknyts av objektorientering menar flera informanter. Genom att programmera utifrån den objektorienterade ansatsen behövs det en struktur på koden som stödjer de principer som objektorienteringen kräver. Av de sex informanterna är det två som har objektorientering som största fokus vid rekrytering av utvecklare. Det är den programmeringsegenskap som värdesätts högst av rekryterarna. De beskriver att inom deras system måste koden skrivas utifrån deras riktlinjer och med deras riktlinjer menar dem objektorientering vilket gör det till en egenskap/kunskap som individen kan inneha.

Av de informanter vars utvecklingsmiljö använder ett objektorienterat programspråk som t.ex. C#, Java och C++ menar att objektorienteringen följer med i kunskapen och erfarenhet från användningen av programspråket. En av informanterna menade att ifall en kandidat uppger att hen kan programmera i Java är det självklart att kandidaten förstår den objektorienterade ansatsen. En annan informant menar dock att detta inte är fallet. Som föregående kategori beskrev fanns det kandidater som hade haft 10 års erfarenhet i ett objektorienterat programspråk men ändå aldrig använt sig utav de objektorienterade principerna. Ur ett forskningsperspektiv är det svårt att skilja på objektorienteringen och kunskapen om ett programspråk eftersom de går hand i hand och det finns inget sätt att samla in empiri om de två områdena separat.

	Objektorienterad ansats	Företagsstorlek
Informant 1	x	700 antal anställda
Informant 2	x	10 antal anställda
Informant 3		10 antal anställda
Informant 4		2 antal anställda
Informant 5	x	170 antal anställda
Informant 6	x	7000 antal anställda

Tabell 4.3 Prioritering av objektorientering

Tabell 4.3 förtydligar resultatet av empirin från samtliga intervjuer om hur informanterna prioriterade den objektorienterade ansatsen hos de kandidater som är i rekryteringsprocessen.

4.2.3 Strukturerat arbetssätt

Flera informanter lägger stor vikt på välstrukturerad kod, och menar att med kod som inte är strukturerad på ett korrekt och tydligt sätt skapar många risker. T.ex. om personen skriver kod som endast personen förstår, blir det svårt att fortsätta på arbetet av andra utvecklare när individen är frånvarande. Men det är också svårt att se vad personen har gjort och varför den gjort som på det sättet, om en annan individ behöver tolka den andras arbete om det ska integreras tillsammans. Välstrukturerad kod och simpla lösningar är något som uppskattas bland informanterna och det reflekteras konstant om kodens betydelse och görs även parallellen att kod är vital för en programmerare, och utan förmågan att skriva bra och tydlig kod, så ifrågasätts personens karriärsval.

På de största företagen förstärks vikten av en välstrukturerad kod. Då samarbeten sker med flera globala stora företag där det är livsviktigt för dem att koden är skalbar och testbar, annars förlorar de stora mängder pengar. Konceptet att ”bara det funkar så är det tillräckligt”, fungerar inte utan samtliga projekt som skapas måste ha hög kvalité och fungera till fullo annars kan det få förödande konsekvenser. Däremot för de mindre företagen är det viktigare att det fungerar, 50 % av informanterna menar att om funktionerna är där samt att tidsbudgeten och kostnadsbudgeten hålls är bristfällig kod acceptabelt. Det betyder inte att de inte har fokus på koden utan det är mer ett val som görs där leverans i tid är prioriterat.

De menar också att kommentarerna är viktiga för att förstå *varför* koden ser ut som den gör, inte specifikt vad som händer. Programmerare har samtliga kunskap om vad som händer i koden, men det är specifikationen på anledningen som är viktig och är avgörande för nästa person som skall ta över. Det är på grund av det här som det är viktigt att programmerare är involverade i processen och förstår helheten för att ha en god uppfattning som gör det möjligt att förklara besluten.

Att skriva kod som bara en individ förstår och arbeta som en individualist försvårar förmågan för andra att fortsätta på arbetet och det innebär att det förloras tid som kunde lagts på samma projekt och underlättat för samtliga och även verksamheten i sig. 80 % av personerna hade en fullständig prioritering på sociala och samarbetsvilliga personer, och personer som samarbetar och liknande skapar en kod som är lättare för andra att fortsätta på, och individualisterna skapar kod som endast dem förstår med lite förståelse för att andra individer skall arbeta med samma material.

Flera individer trycker på vikten med att göra det rätt från början och att vara förberedd. Ett exempel är att göra flödesschema så att arbetarna från början vet vad som ska göras och i vilken ordning. När systemet väl implementeras blir det problematiskt att förändra kod och det blir högre underhållningsgrad vilket resulterar i att kostnaden stiger. Då samtliga företag håller med om att pengar är det viktigaste så blir förberedelserna innan projekten sätts igång mycket viktiga och kan påverka resultatet positivt.

En informant menar dock att beroende på syftet kan struktureringen av koden se annorlunda ut. Om en prototyp ska utvecklas behöver inte nödvändigtvis strukturen på koden vara den mest optimala och bästa eftersom det enda syftet med projektet är att visa en funktion. Men om det är en produkt som ska levereras, och en kund som ska ta över produkten vid leveransen måste strukturen vara helt problemfri och lättförståeligt. Det måste vara en helt felfri kod som är lättläst för att möjliggöra vidareutveckling av kunden eller någon annan utvecklare.

	Strukturerat arbetssätt	Företagsstorlek
Informant 1	x	700 antal anställda
Informant 2	x *	10 antal anställda
Informant 3	x	10 antal anställda
Informant 4		2 antal anställda
Informant 5	x	170 antal anställda
Informant 6	x *	7000 antal anställda
	* = Beror på situation	

Tabell 4.4 Prioritering av strukturerat arbetssätt

Tabell 4.4 förtydligar data från informanterna. Med (*) menas att informanterna kan acceptera ett ostrukturerat arbetssätt beroende på faktorer som t.ex. tid och kostnad d.v.s. att det beror på vilken situation arbetet är i.

4.2.4 Helhetslösningar

På den här punkten var det varierande svar från de mindre och större företagen. I de större företagen var det inget stort krav på att ha helhetslösningstänk i en större utsträckning. Båda företagen av den större storleken menade att det är viktigt att de förstår vad de ska göra och hur de ska integreras med andra, men ett större tänk än det är inte nödvändigt för det stora företaget. Men för det mindre företaget sågs det som en självklarhet att personen var involverad i samtliga delar för att få större förståelse för uppgiften. Det är storleken som är faktorn och i större företag läggs helhetslösningstänket på projektledaren eller liknande, som klarar av att lösa problemet utan att det påverkar kodningen.

Programmeraren som fokuserar mycket på helhetslösningen i ett stort projekt kan tappa fokus och skapa onödiga funktioner eller förivra sig i detaljer som inte har betydelse. Men däremot i de mindre företagen anses det att det är viktigt att individen har helhetstänket i åtanke menar en informant. Anledningen är att i mindre företag ges det mindre projekt vilket innebär att samma roll som projektledaren har vid stora projekt, ges till varje individ. Det innebär att programmeraren behöver ta större ansvar och ha samtliga aspekter i åtanke när individen skapar funktionen eller det som skall göras. Det är viktigt att samarbetet inom företaget existerar, t.ex. att programmerare och säljare har en kommunikation. Finns inte den kommunikationen blir det svårt att anpassa sig efter varandra och samarbeta. Många företag ger programmerare enorma friheter, vilket också har genererat ett bra resultat. Det visar att fria individer som kommunicerar med varandra och samarbetar är den bästa metoden inom sådana branscher och verksamheter.

Informanten från det stora företaget menar att det skiljer sig på helhetstänket mellan olika roller. Det ger exempel att det måste finnas individer med specifika kvalitéer som har stora kunskaper inom ett specifikt område, och de individerna behöver inte ha ett helhetstänk enligt dem. Men de resterande bör samtliga ha ett helhetstänk och ha helheten i åtanke när de arbetar. Anledningen till det är deras konstanta arbetsform som är i grupper med samarbete i fokus. Med mer perspektiv och kunskaper som kommer de tillsammans fram till lösningar som ger ett bättre resultat än om få individer sköter samtliga dimensioner av verksamheten.

Helhetstänket inom mindre företag blir då annorlunda eftersom att de är få personer och då krävs det att samtliga har ett helhetstänk för deras arbetsområden är bredare.

Om individer inte blir involverade kan det resultera i passivitet, enligt en av informanterna, då de väntar på uppgifter istället för att ta initiativ själva och använda den egna problemlösningsförmågan. Däremot om de blir involverade och förstår problemet kan de på egen hand försöka lösa problemen tillsammans, istället för att vänta på att de som är ansvariga ska ge dem order om vad som ska göras. Det ger en otrolig frihet inom verksamheten som genererar ett bra resultat enligt en informant.

”Vi är alla medansvariga för hur resultat går i hamn.” Informant 3.

En informant menar att alla mjukvaru-utvecklare bör vara involverade i verksamhetsperspektivet vid sina lösningar. Att bara bli tillsagd att utveckla vissa funktioner eller algoritmer är ingenting som skapar motivation och bra kod.

	Helhetslösningar	Företagsstorlek
Informant 1		700 antal anställda
Informant 2		10 antal anställda
Informant 3	x	10 antal anställda
Informant 4	x	2 antal anställda
Informant 5	*	170 antal anställda
Informant 6	*	7000 antal anställda
	* = *Beror på situation*	

Tabell 4.5 Helhetslösningar

Tabell 4.5 beskriver informanternas prioritering om helhetslösningar. Med (*) menas att informanterna prioriterade kategorin utifrån vilken situation projektet var i. Vissa projekt behövs inte helhetslösningar utan utvecklaren ska bara utveckla en funktion och ingenting annat.

4.3 Datalogiskt tänkande

4.3.1 Analytiskt tänkande/logiskt tänkande

Analytiskt tänkande var viktigt för samtliga intervjuade där problemlösningsförmågan och förmågan att bryta ner problem sågs som de mest vitala delarna av det den egenskapen. Ett av företagen nämner att utan den förmågan blir det svårt för dem att arbeta självständigt och det kräver mycket hjälp från andra, vilket i slutändan blir tidskrävande och inte hållbart i längden eftersom kostnaden blir för hög. Det händer ofta att saker byggs som inte behövdes eller gjorde saker bättre än det var tvunget att vara. Det kan även vara så att det skall användas i endast sex månader och då är detaljerna ännu mer onödiga än med ett permanent system. Det är viktigt att förstå samtliga scenarion inom uppgiften för att sedan välja det mest optimala. Utan den kunskapen blir det problematiskt att analysera och att göra ett val blir svårt.

”Alla vägar bär till Rom, men det är olika långt dit.” Informant 4.

Logiskt tänkande är också en viktig faktor, när vi bad dem att förklara vad de vill ha av en anställd, beskrev dem många faktorer som var nära kopplade till logiskt tänkande och då också datalogiskt tänkande även om de inte var medvetna om vad datalogiskt tänkande är för något och hur det kan hjälpa i t.ex. problemlösningen. Analytiskt och logiskt tänkande är en egenskap som kan vara väldigt svår att se på en individ under en rekryteringsprocess. Det är någonting som kan igenkännas efter en tids arbete menar samtliga informanter. Det var två företag som utmärkte sig gällande det analytiska och logiska tänkandet. De använder sig av logiska tester under rekryteringsprocessen och enligt dem får de en god inblick i hur individen använder förmågan att analysera och hitta logiska lösningar till de problem som uppstår.

En av informanterna berättade att den egenskap som de värderade högst var analytiskt tänkande. Det var väldigt viktigt att en mjukvaru-utvecklare har förmågan att utvärdera alla möjliga utfall av ens handlingar och se logiken i det man gör. Därför har företaget inte något programmeringstest utan endast personlighetstester och logiska tester. Det logiska testet var en matris där den nästkommande matrisen i ordningen som har en bakomliggande logisk regel ska hittas. De menade att utifrån det här testet kan man få en tydlig inblick i hur individen är som utvecklare eftersom mycket av programmering bygger just på ett logiskt flöde och problemlösning i utvecklingen.
Under rekryteringsprocessen använder de sig även av *kompetensbaserad* intervju där istället för att kandidaterna ska svara på frågor, ska dem ge konkreta exempel där de använt sig av den kompetens som företag söker. De påpekar att deras intervjuer ger dem en väldigt god bild över kandidaten och dess analytiska förmåga. De har en vision om att en individ med analytiskt och logisk förmåga är naturligt talangfulla utvecklare.

Utifrån den kunskap som en av informanterna hade om datalogiskt tänkande, kopplade informanten analytiskt och logiskt tänkande till datalogiskt tänkande som en grundfaktor. Han menar att analytiskt tänkande handlar just om att bryta ner komplexa situationer och studera de utfall och möjligheter som finns med problemen och arbeta på den mest logiska och bästa lösningen. Informanten menar att en person som konstant behöver hjälp med problemlösning, har förmodligen inte en talang för analytiskt tänkande och datalogiskt tänkande. Det leder till en ganska icke effektiv utvecklare eftersom mycket inom utveckling bygger på problemlösning och analysering.

En informant som använde sig av ett utförligt programmeringstest under rekryteringsprocessen berättade om deras tolkning av analytiskt tänkande. De har skapat ett test som tar flera timmar att genomföra. Informanten menar att genom att utföra testet måste kandidaten använda sitt analytiska tänkande eftersom det finns flera olika lösningar på det problem som kandidaten blivit tilldelad. Informanten menar att det är väldigt svårt att upptäcka de egenskaper som är abstrakta genom att bara undersöka ett CV. Han menar att testet vinklar individens förmåga att använda sitt analytiska och logiska tänkande och sedan förklara vilka andra lösningar det finns till problemet. Informanten menar även att genom deras test är det ett medvetet motiv att göra provtiden för kort. Kandidaterna ska alltså inte hinna med att bygga klart lösningen. Informanten påpekar att det är vid diskussionen efteråt som dessa egenskaper upptäcks, som t.ex. analytiskt tänkande, eftersom då får kandidaten möjligheten att förklara hur man tänkte och agerade. Men även de andra situationer som kunde ha uppstått ifall ett annat tillvägagångssätt hade använts.

En annan informant på ett av de mindre företagen hävdar att analytiskt och logiskt tänkande är egenskaper som är högst relaterat till erfarenhet inom utveckling. Genom att vara erfaren sker det analytiska tänkandet via reflexer och det är ingenting som reflekteras över medvetet. Informanten påpekar att under deras programmeringstester kan man snabbt filtrera ut kandidaterna eftersom det analytiska tänkandet är kopplat till den mängd av tid som behövs för att genomföra en uppgift. Vidare leder tid till en kostnad i projektet vilket är negativt i ett litet företag då det oftast genomför endast ett projekt åt gången. Informanten menar att dessa egenskaper dock är ingenting som dem lägger allt för stort fokus på i en rekryteringsprocess eftersom han menar att de egenskaperna kommer med erfarenheten. Erfarenheten kan endast bli längre om chansen ges att börja någonstans.

	Analytiskt tänkande/Logiskt tänkande	Företagsstorlek
Informant 1	x	700 antal anställda
Informant 2	*	10 antal anställda
Informant 3	x	10 antal anställda
Informant 4	x	2 antal anställda
Informant 5	*	170 antal anställda
Informant 6	*	7000 antal anställda
	*= Beror på situation	

Tabell 4.6 Prioritering av analytiskt/logiskt tänkande

Tabell 4.6 illustrerar informanternas prioritering vad gäller analytiskt och logiskt tänkande. *X* betyder att informanterna prioriterar dessa egenskaper vid en anställning. Med (*) menas med att beroende på vilken situation och tjänst som eftersöks kan prioritering av analytiskt och logiskt tänkande variera.

4.3.2 Abstraktion

Inom detta område var företag överens om att det är en egenskap som är viktigt inom programmering. Det är fördelaktigt att inneha egenskapen att kunna avgränsa sig och fokusera på det viktiga och ignorera irrelevanta detaljer. En del viktiga aspekter som de grundar det på är tid och kostnad. Samtliga informanter nämner att kan de göra någon detalj bättre utan att det påverkar tid eller kostnad så gör dem det för att kunden ska bli nöjdare.

“Det viktigaste är offerten, allt annat får ses som bonus” Informant 2.

Tidsfaktorn är återkommande bland samtliga programmerare och har stor påverkan på resultatet och hur mycket funktioner eller detaljer som kan inkluderas i arbetet. Då ledtiden vid många projekt är statisk och inte kan ändras, är tiden en indikation på hur mycket som kan göras. Om det inte görs tillräckligt eller görs för mycket kommer det att kosta mycket, då tid och kostnad är direkt relaterade i företagsverksamheter antyder en informant.

Det nämns att avgränsning är mycket viktigt, och förmågan att kunna ignorera onödiga detaljer och fokusera på det relevanta. Förmågan att avgöra vad som är viktigt är otroligt värdefullt säger en informant från det största företaget. Ifall den förmågan är bristfällig kan det läggas fokus och resurser på en funktion som inte behövs. Det innebär att resurserna försvinner men som istället kunde investeras i de viktiga funktionerna. En av informanterna

beskriver hur under hans erfarenhet från branschen försökte han lösa ett problem varav han stötte på mer problem under tiden och försökte lösa dem också. Det innebar att han lade tid på onödiga detaljer för att lösa samtliga problem. Det är svårt att hitta en balans mellan viktiga detaljer och irrelevanta detaljer, men finner en individ den egenskapen, då är den värd mycket.

Abstraktion beror också på vilken typ av funktion som skall utvecklas. Ifall det endast behövs utvecklas en prototyp eller en funktion som ska användas en kort tid, är detaljer inte av värde då det endast behöver fungera för att verka som ett underlag för tester i ett senare skede. Men däremot när det ska utvecklas ett helt system som är övergripande och kräver att allting skall fungera tillsammans, då är det viktigt med detaljer eftersom små fel i detaljerna kan påverka hela systemet. Abstraktion har en liknande betydelse som strukturerat arbetssätt då prioriteringen av denna kategori beror mycket på kontexten och det sammanhang som det verkar i.

Det nämns av det största företaget att för detaljer kan det vara svårt att identifiera vilka som är viktiga och vilka som är irrelevanta för projektet. Det nämns också att det är svårt att finna sådana, och flera ger exempel på duktiga programmerare som har en tendens att lägga tid på irrelevanta detaljer vilket innebär mer kostnad för projektet än vad som budgeterats. Att ha den egenskapen att kunna gå in i viktiga detaljer, och sedan gå ut till helheten och inte tappa fokus är en otroligt värdefull egenskap som hjälper individen att identifiera relevanta områden.

	Abstraktion	Företagsstorlek
Informant 1	x	700 antal anställda
Informant 2	x	10 antal anställda
Informant 3	*	10 antal anställda
Informant 4	x	2 antal anställda
Informant 5	x	170 antal anställda
Informant 6	x	7000 antal anställda
	*= Beror på situation	

Tabell 4.7 Prioritering av abstraktion

Tabell 4.7 beskriver informanternas prioritering av abstraktion som en egenskap hos en kandidat. Fem av sex informanter antyder att abstraktion är en egenskap som föredras hos en kandidat. En av informanterna menar att beroende på situation kan egenskapen variera i prioritet vilket illustreras med *.

4.3.3 Problemlösning

I den här kategorin var de flesta informanter överens om att det krävs problemlösning till en viss grad men det är inte krav att de ska lösa alla problem. En del informanter menade att majoriteten av utvecklarna helst ska koda och inte vara med i det större perspektivet, medan en del andra informanter menade att det är viktigt att de är med i processen så att de förstår kontexten i problemet och kan förstå hur allting ska samverkas. Den faktorn som pekades ut är storleken på företaget. Ifall företaget är större bör ett mer omfattande ansvar ges för att

förstå helheten runt problemet och hur det skall integreras med andra delar inom samma system. Problemlösning är en stor del för samtliga och varenda person menar att problemlösningsförmågan är högt prioriterad bland kandidater.

Flera av informanterna prioriterade problemlösningsförmågan högre än själva programspråket, på grund av att programspråk är något som kan läras upp med tid och problemlösningsförmågan är svår att läras ut under en kort tid. Det sades även att en bra programmerare har automatiskt en bra problemlösningsförmåga, och att utan en god problemlösningsförmåga kan inte programmeraren anses som duktig för att konstant uppkommer problem. Finns inte möjligheterna eller kunskaperna att lösa dem är det negativt.

Samtliga informanter menar att mjukvaru-utvecklare ska vara involverade i problemlösningsprocessen för att ge fler perspektiv och olika vinklar till att lösa problemet. De får även mer insyn i hela verksamheten och får en större helhetsbild än om de bara programmerade. En del informanter påpekade att om programmeraren endast ska programmera utan att tänka på helheten och vara en del av processen kan det arbetet göras av någon annan för billigare pris. Det nämns dock att om utvecklare ges för mycket frihet, kan det hända att individen lägger fokus på fel uppgifter, och då krävs det att individen blir tillsagd och fokuserar på rätt saker igen. Blir de involverad i processen utvecklas även helhetstänket som tidigare nämnts, som är mycket nyttigt för verksamheten och även för individen.

Problemlösningsförmågan tar rekryterarna reda på innan de anställs i form av intervjuer eller programmeringstest. Resultatet i testerna ligger sedan som grund för beslutet som tas om en eventuell anställning. Det förekom också diskussioner om testet där de får förklara för att ytterligare verifiera deras problemlösningsförmåga, vilket styrker validiteten för deras tester som utförs.

Samtliga informanter menade att problemlösningsförmågan är en prioritet under en rekryteringsprocess och att det är viktigt för en programmerare att inneha den egenskapen. Det dyker ständigt upp nya problem och finns inte förmågan att lösa dem utan att de ständigt behöver hjälp kommer det att ta lång tid, vilket innebär att även kostnaden stiger. Att skriva kod kan i princip alla programmerare som har erfarenhet men att lösa problem löpande och ta hänsyn till verksamheten och hitta den bästa lösningen, den egenskapen är sällsynt i branschen idag menar en av informanterna.

Det fanns informanter som menade att problemlösningsförmågan behöver inte gälla på samtliga områden, är det större problem som påverkar hela verksamheten kan det vara bra att hyra in till exempel en verksamhetsanalytiker som sköter den uppgiften. Men inom just programmeringen är det snarare ett krav att individen kan lösa det problem som dyker upp inom kodning. Ifall inte den egenskapen finns blir det svårt att lyckas inom området där det ständigt finns tidskrav och budgetar som måste hållas.

	Problemlösning	Företagsstorlek
Informant 1	x	700 antal anställda
Informant 2	x	10 antal anställda
Informant 3	x	10 antal anställda
Informant 4	x	2 antal anställda
Informant 5	x	170 antal anställda
Informant 6	x	7000 antal anställda
	*= Beror på situation	

Tabell 4.8 Prioritering av problemlösning

Tabell 4.8 illustrerar att problemlösning är en egenskap som prioriteras av samtliga informanter vid en potentiell rekrytering.

4.4 Dataanalys av enkät

Antalet respondenter som genomförde enkäten var 21 personer. I enkätundersökningen har ett par generella frågor formulerats för att få en uppfattning om vad respondenterna har för tidigare kunskap om datalogiskt tänkande. Det ger en tydligare bild över hur responsen från enkäten kan generaliseras och appliceras på resultatet från de semistrukturerade intervjuerna. Delar av enkäten bestod av kvalitativa öppna frågor med svar som respondenten själv skrivit. Enkäten skickades ut till företag som har systemutveckling som ett område i sina verksamheter, och till personer med samma roller som i den kvalitativa undersökningen, nämligen rekryterare och utvecklare. Verktyget som använts var Webbenkäter (2017) vilket användes för att skapa frågorna och dels för att sammanställa svaren.

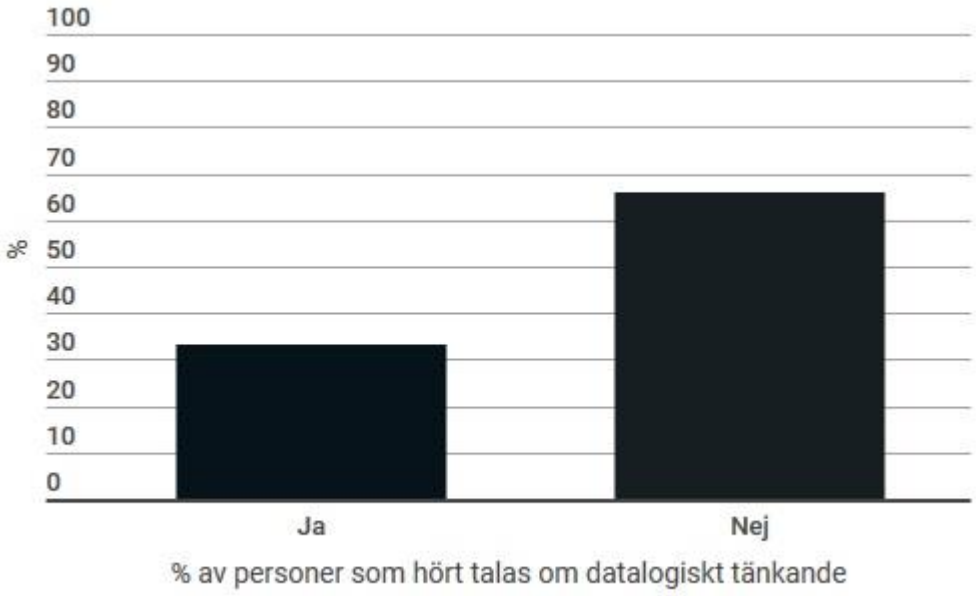


Diagram 4.1 Inledande enkätfråga

Som diagram 4.1 visar var det endast 35 % av respondenterna som någon gång innan den här enkätsundersökningen hört talats om begreppet datalogiskt tänkande. Eftersom frågorna baseras utifrån det datalogiska perspektivet är detta ett intressant resultat eftersom det betyder att respondenterna inte *vet* vad den övergripande agendan är. Det ger också ett perspektiv i hur många av respondenterna som är insatta i området.

Väldigt viktigt	Viktigt	Beror på situation	Möjligtvis	Ingen alls	
47,62	38,1	14,29	0	0	%

Tabell 4.9 Värdering av problemlösning

Tabell 4.9 beskriver respondenternas svar till vilken grad problemlösningsförmågan hos en individ påverkar chansen till en eventuell rekrytering. Nästan hälften, 47,62 % svarar att det är väldigt viktigt med att använda problemlösningsförmågan under processen. Detta stämmer överens med den insamlade datan från de kvalitativa intervjuerna vilket stärker datan. Det är ett tydligt resultat med strax under 86 % som värderar problemlösningsförmåga högt.

Diagram 4.2 Krav på programspråk

	Väldigt stor nytta	Stor nytta	Beror på kandidat	Möjligtvis	Ingen alls	
	19,05	42,86	23,81	9,52	4,76	%

Tabell 4.10 Värdering av nyttan att lära upp en individ

Diagram 4.2 och tabell 4.10 beskriver korelationen mellan två frågor som angavs i enkäten. Det var 76 % som menade att i deras beskrivning av tjänsten som eftersöks har dem krav på programspråk som kandidaten måste ha kunskap och erfarenhet av. Tabell 4.10 beskriver i vilken utsträckning som arbetsgivaren ser nytta i att lära upp individers i programspråk som de inte hanterat innan. 43 % ~ anser det finns framgång i att göra det. Dock är det nästan 35 % som anser att det beror på vilken kandidat det handlar om eller att det inte finns någon anledning till att göra det. Kopplingen är att de har krav, men att de fungerar mer som en preferens, och finns det möjlighet att lära upp en kandidat inom ett liknande programspråk, då utnyttjas den möjligheten oftare än inte.

	Väldigt viktigt	Viktigt	Beror på situation	Möjligtvis	Ingen alls	
	57,14	38,10	4,76	0	0	%

Tabell 4.11 Strukturerat arbetssätt

Tabell 4.11 beskriver empirin från respondenterna ifall ett strukturerat arbetssätt eftersöks hos en individ. Med strukturerat arbetssätt som tidigare beskrivits menas en källkod som är lättläst, kommenterad och förståelig för alla intressenter som koden berör. ~57 % menar att det är väldigt viktigt hos den potentiella kandidaten men även hos det dagliga arbetet som utvecklare. Utfallet stämmer överens med resultatet från den kvalitativa undersökningen, vilket innebär att åsikterna är enade vilket stärker den kvalitativa datan.

	Väldigt viktigt	Viktigt	Beror på situation	Möjligtvis	Ingen alls	
	57,14	33,33	9,52	0	0	%

Tabell 4.12 Personliga egenskaper

Tabell 4.12 beskriver respondenternas svar om hur en kandidats personliga egenskaper påverkar rekryteringsprocessen. Med personliga egenskaper riktas det mot en social egenskap och att kunna kommunicera med andra kollegor på ett effektivt sätt. De personliga egenskaperna är något som prioriteras högt hos samtliga, då samtliga respondenter säger att personliga egenskaper spelar roll i rekryteringsprocessen och har en avgörande roll i valet av kandidat.

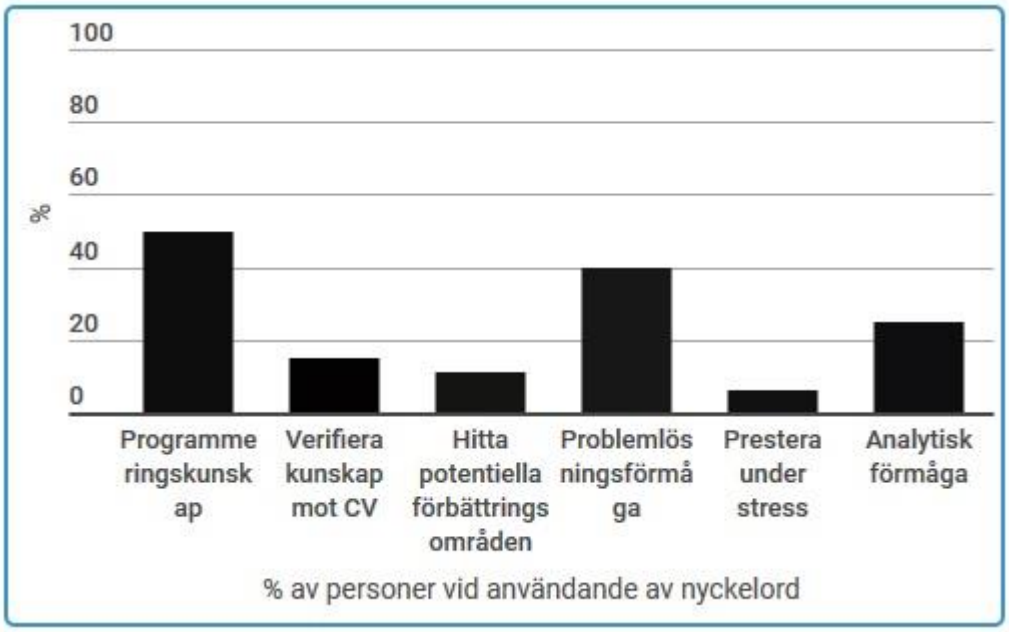


Diagram 4.3 Nyckelord för programmeringstesters påverkan vid en rekryteringsprocess

Kodning (Recker, 2013) har använts för att tyda de kvalitativa svaren från enkäten. Diagram 4.3 beskriver hur många procent av respondenterna som använts sig av nyckelorden från den genomförda analysen av svaren. Eftersom respondenter använde sig av flera nyckelord representerar procentsatsen andelen av nyckelord som använts av flera respondenter. Angående frågan vad tester har för inverkan och hur resultatet kan ge en inblick i kandidaten, beskriver 50 % att genom programmeringstest får rekryteraren en god inblick av kandidatens faktiska programmeringskunskaper. 40 % av respondenterna nämnde att det är också för att upptäcka kandidatens förmåga av problemlösning och hur kandidaten reflekterar över tillvägagångssättet för problemet.

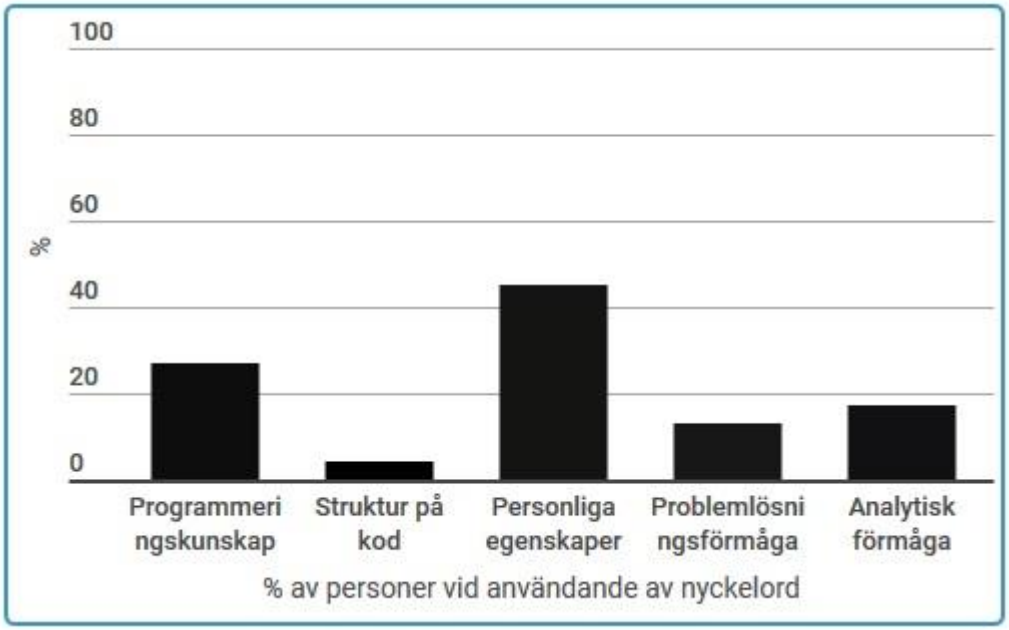


Diagram 4.4 Prioritering av egenskaper

Diagram 4.4 illustrerar respondenternas användande av nyckelord vid frågan om vilken egenskap de värderar mest hos en utvecklare vid rekryteringsprocessen. 45 % av respondenterna nämner personliga egenskaper som en av de viktigaste egenskaperna hos en utvecklare. Med personliga egenskaper menas social, kreativ, sammarbetsvillig, utåtriktad, arbetsmoral och positiv är de personliga egenskaper som respondenterna beskrev som egenskaper. Det här ger en bra översikt av rekryteringen och vad som prioriteras.

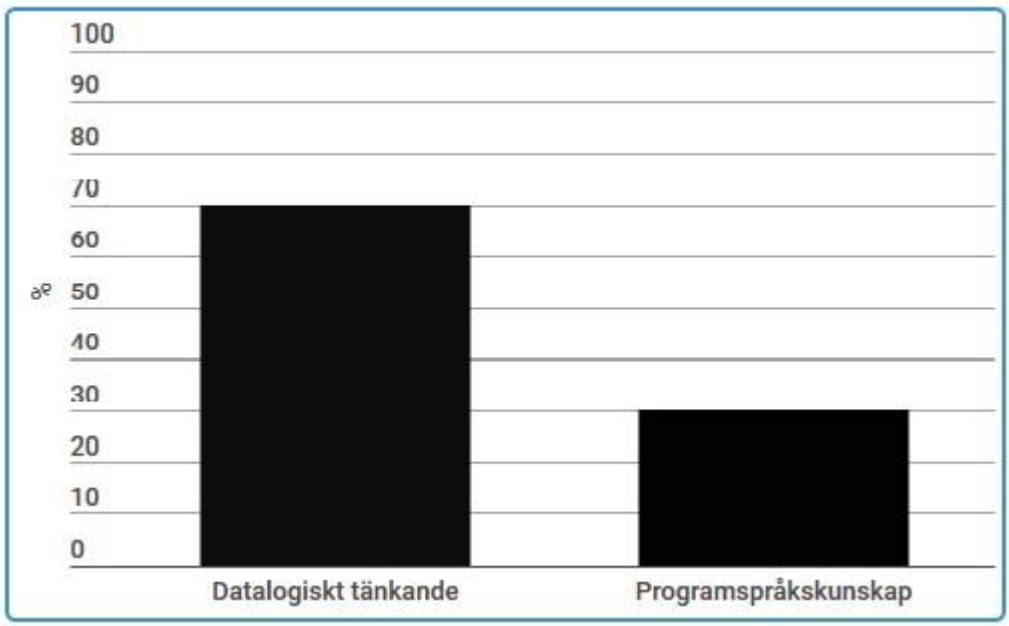


Diagram 4.5 Selektion av datalogiskt tänkande och programspråkskunskap

En fråga i enkäten som respondenterna fick möjlighet att svara på var *vilken individ föredrar du?* Till denna fråga fick respondenterna två val, datalogiskt tänkande eller programspråkskunskap. Som diagram 4.5 illustrerar var det 70 % av respondenterna som hellre valde en individ med gott datalogiskt tänkande och bristande programspråkskunskaper istället för råa programspråkskunskaper och ett bristande datalogiskt tänkande.

5 Diskussion

5.1 Informanternas och respondenternas omedvetna prioritering

Wing (2008), Nygård (2016) och BareFootcas (2008) har tillsammans en tydlig bild över konceptet datalogiskt tänkande. I den här studien har konceptet utformats utifrån dessa definitioner från tidigare forskning kring ämnet. Problemlösning och abstraktion är två nära relaterade begrepp inom datalogiskt tänkande vilket tillsammans utgör den principiella grunden. Att skilja dessa begrepp ifrån programmering är en abstrakt och otydlig avgränsning att göra. Dock kan ett intressant fynd tas upp vid en jämförande studie som denna. 65 % (figur 4.4.1) av samtliga respondenter och 80 % av informanterna medgav att innan den här studien, har de inte hört talats om begreppet datalogiskt tänkande utan de kunde endast gissa utifrån namnet vad det var för något.

Eftersom denna låga frekvens av kunskap om datalogiskt tänkande hos informanterna och respondenterna är det intressant att nästan majoriteten av resultatet från intervjuerna och enkäten relateras till de byggstenar och kategorier ifrån datalogiskt tänkande är någonting som prioriteras. Flera av nyckelord som problemlösning, abstraktion, analytiskt eller logiskt tänkande och helhetslösningar är egenskaper hos en individ som är bland de högst prioriterade egenskaperna vid en rekryteringsprocess hos en potentiell kandidat. Rekryterarna berättar sin egen version av datalogiskt tänkande, fast utan namn när de beskriver de egenskaper som eftersöks. Flera av informanterna och respondenterna förespråkar att dessa kategorier som tagits fram i metodkapitlet är faktiskt de egenskaper som de eftersöker tillsammans med de personliga egenskaperna, t.ex. social, kreativ etc. Den här studien har visat att det finns många faktorer som spelar in i urvalet av kandidater och till slut handlar det om personliga preferenser. Vissa informanter väljer den första kandidaten som har bredare kunskaper inom endast programmering, och en annan informant väljer den individ med god problemlösningsförmåga och som kan diskutera sitt tankesätt och möjligheter utan att behöva göra en praktisk lösning i kod. Den praktiska lösningen är något som anses vara möjligt att lära sig med tiden.

Datalogiskt tänkande och programmeringsegenskaper är två områden som är svåra att skilja och göra det till ett förståeligt ämne. Att arbeta som programmerare kräver ett visst datalogiskt tänkande (Wing, 2008) men det finns skillnader hur skicklig en individ är på just den egenskapen. Men eftersom datalogiskt tänkande kan användas även utanför den digitala världen, (Wing, 2008) är det en egenskap att sträva efter. Detta gör datalogiskt tänkande till en egenskap som blir viktig att utveckla och det är en av anledningarna till att Skolverket (2016) och England har implementerat det i grundskolan.

5.2 Personliga egenskaper – stor påverkan

Arbetsförmedlingen (2017) menar att arbetsgivaren bryr sig om en individs personliga egenskaper lika mycket som en dess kompetens rörande den eftersökta tjänsten. Personliga egenskaper sågs av i princip samtliga informanter och respondenter som en viktig egenskap och som i en viss utsträckning ansågs vara ett krav för anställning inom verksamheten. Sedan prioriterades en av de två delarna, programspråkskunskaper eller datalogiskt tänkande. Det som påverkade det här valet var storlek och vilket typ av struktur det var på företaget. Kraven på datalogiskt tänkande minskar och ökar med företagets storlek. Personliga egenskaper var viktiga för samtliga för arbetsmiljön, men ännu viktigare för företag som arbetade i projektform, där samarbete och kommunikation är viktigt. Ett av de mindre företagen nämnde att han föredrog en som var duktig på att programmera, och sedan litade på diverse ledare som skulle lyckas styra honom i rätt riktning. Utöver det talade samtliga högt om datalogiskt tänkande i andra hand efter personliga egenskaper.

Det blev genast tydligt att personliga egenskaper har hög prioritet bland rekryterare. Varje individ, både från den kvantitativa och kvalitativa undersökningen, menade att personliga egenskaper har en viktig roll i rekryteringsprocessen. Till den utsträckning att det var den enda egenskap som analyserades grundligt innan rekrytering. Det lades senare till att de inte gjorde grundliga test för att inte skrämma iväg potentiella kandidater när det redan var så få ansökningar, men det är anmärkningsvärt att det är strukturerat på ett sådant sätt. I ett sådant tillfälle hjälper ett CV till med att låta rekryterarna förstå nivån på programmeringen, och om den överensstämmer med kraven kan sedan de personliga egenskaperna identifieras och sedan ta ett beslut. Det förvånade oss som forskare att de personliga egenskaperna har en större roll i en teknologisk bransch där resultatet innefattar en stor del av helheten.

Den stereotyp som framförallt informanterna framställer som en utvecklare är antisociala och icke samarbetsvilliga individer. Det är sådana typer av individer som samtliga undviker, enligt intervjuerna. När samhället och verksamheter konstant utvecklas krävs det andra typer av utvecklare, med andra typer av egenskaper, där den sociala egenskapen är högt prioriterad. Det var få informanter eller respondenter som svarade annorlunda och prioriterade programmeringskunskap. Det är en intressant punkt som uppkom efter undersökningsfasen att verksamheter ändrar sitt fokus till ett mer individbaserat fokus, istället för ett kunskapsfokuserat rekryteringsbehov. Det var en generell åsikt från författarna, som baserats på utbildningen och efter granskade jobbannonser, att programmeringskunskaper var högsta prioritet. Men med tanke på att arbetsformerna är i projektform och att programspråkens gränser mellan varandra minskar, ökar prioriteringen på det datalogiska tänkandet och individens personliga egenskaper.

5.3 Kategoriernas betydelser

5.3.1 Problemlösning

Nygård (2015) menar att programmering innebär att bryta ner komplexa problem och konstruera algoritmer. Hon uttrycker problemlösning som en viktig egenskap hos en programmerare för att kunna utföra arbetet effektivt. Problemlösning var den egenskap som ofta nämndes och hade högst prioritet bland egenskaper hos en potentiell kandidat. Det var viktigt att den har möjligheten att arbeta självständigt och kunna lösa problem som dyker upp inom verksamheten. Ifall individen har dålig problemlösningsförmåga krävs det att individen styrs och kräver resurser som kan läggas på aktiviteter som gynnar verksamheten. Datalogiskt tänkande är en problemlösningsprocess (Wing, 2008), vilket självfallet kopplar samman de två områdena problemlösning och datalogiskt tänkande. Fem av sex av informanterna föredrog en datalogiskt talangfull individ framför en med goda programkunskaper med bristande datalogiskt tänkande, och på samma fråga föredrog 70 % av respondenterna en individ med datalogiskt tänkande. På den första frågan på enkäten svarade 65 % att de inte hade hört begreppet tidigare. Slutsatsen som dras är att många rekryterare söker efter datalogiskt tänkande, men vet inte vad begreppet innebär och hur test skapas för att få reda på den informationen.

5.3.2 Programspråk

Inför intervjuerna hade vi förutsatta meningar om att programspråk var det viktigaste kravet att uppfylla vid en potentiell anställning. Men efter undersökningarna har den åsikten förändrats och att det ses som en preferens och liknande programspråk accepteras av arbetsgivarna. Det beror dels på att det är brist på mjukvaru-utvecklare generellt sett, men också att språken börjar likna varandra mer och mer, vilket gör att kraven på programspråk blir lägre. Det enda mönstret som uppfattades var den objektorienterade ansatsen, där det sades att i princip allt är objektorienterat och att det är en viktig utveckling som bör följas för att konkurrera om arbeten inom branschen. Även strukturering av kod var av stor betydelse, då specifikt för stora företag, där de riskerar förluster om den inte är tillräcklig för att flera ska arbeta med samma kod. Båda dessa faktorer lärs genom erfarenhet och kommer efter ett tag att bemästras, på en kortare tid än datalogiskt tänkande, vilket gör programspråk enklare att bemästra. Programspråk borde därför prioriteras lägre än t.ex. datalogiskt tänkande på grund av svårighetsgraden att bemästra. Det överensstämmer med empirin och verifierar våra tankegånger om arbetsgivarens prioriteringar och referenser.

Respondenterna och informanterna svarade samtliga att de ser värde i att lära upp individer om kandidaten kan hantera liknande programspråk eller har tillräckliga kvalifikationer inom problemlösning. Det är viktigt att vara flexibel i och med att skillnaderna mellan programspråken minskar vilket gör det enklare att lära sig ett programspråk som är likt det som programspråk som behärskas. Upplärningsprocessens tid beror självfallet på individens egenskaper och kunskaper. De personliga egenskaper som efterfrågas som t.ex. koncentrationsförmåga och ambitionsnivå påverkar hur snabbt och effektivt ett nytt programspråk kan hanteras. Även programkunskaper påverkar inlärningsprocessen och är en faktor som bör tas i åtanke vid anställning. Det förvånade oss att se så många av liknande åsikt att upplärning används. Vi trodde att kraven var mer statiska, men i och med att programspråken blir mer lika varandra och att datalogiskt tänkande blir viktigt, anser vi att upplärning blir ett viktigt verktyg att använda.

5.3.3 Helhetslösningar

Flera företag hade åsikten att utvecklarna ska vara involverade i utvecklingsprocessen, för att förstå sammanhanget de verkar i och få bättre helhetssyn på uppgiften som skall göras. Den påverkande faktorn som vi finner är storleken på företagen och således deras projekt. Ett mindre företag, som har mindre projekt, innebär att det inte existerar många funktioner som inte kräver att individen förstår verksamheten. Det betyder att i princip samtliga funktioner eller detaljer kräver att individen har kunskap om sammanhanget för att ha möjlighet att skapa det som krävs för projektet. På stora företag ökar antalet funktioner och då även antalet funktioner som inte kräver någon kunskap om verksamheten. Även om samtliga påpekade att en involvering hjälper till med det analytiska tänkandet och att hitta det utfall som ger bäst resultat, för att kunskapen om området är större. Det leder också till att problemlösningsförmågan underlättas, den blir inte möjligtvis bättre hos individen, men det underlättar i beslutsfattandet.

5.3.4 Analytiskt & logiskt tänkande

Den här kategorin är svår att få fram konkret data eller information inom som är användbar när begreppet i sig är abstrakt och det är svårt att upptäcka hos en individ. Men eftersom datalogiskt tänkande är ett slags analytiskt tänkande i grunden(Wing, 2008), är de självfallet starkt kopplade. Det utvecklas genom erfarenhet enligt några, och några menade att det är en talang som är svår att utveckla. Samtliga sa att det var en viktig egenskap att erhålla, och att det gick hand i hand med problemlösning. Även den här datalogiska egenskapen prioriterades högt hos individerna. Den här kategorin tillsammans med problemlösning är möjlig att finna genom tester till en viss utsträckning, där är OPQ-tester(Testhuset, 2008) och liknande tester ett verktyg för att identifiera dessa egenskaper och hur utvecklade de är. Analytiskt tänkande är helt klart en egenskap som prioriteras då det existerar inom programspråkkunskaper också, vilket gör det till en eftertraktad kunskap att erhålla som en kandidat.

5.3.5 Abstraktion

Som Booch (1991) beskriver är abstraktion ett ytterst komplext område där många misstolkningar kan göras. Booch (1991) menar att abstraktion betyder att visa den fakta som efterfrågas och gömma det onödiga. Abstraktion var svårast att finna konkret information om från respondenterna och informanterna på grund av att området är så abstrakt. Det är ett stort område som avgränsades till att betyda avgränsning inom de båda områdena. Det var viktigt att göra rätt aktiviteter vid rätt tidpunkt. Resultatet visade att tid och kostnad är två faktorer som spelar en stor roll inom området. Det beror på i vilken kontext det talas om. De ser trots allt värde i detaljer om det inte påverkar de två faktorerna, vilket är en egenskap som finns inom abstraktion att förstå vilka funktioner där detaljer kan förfinas, och vilka funktioner som inte behövs.

Förmågan att se detaljer, men också kunna gå tillbaka helheten utan att tappa koncentrationen på de viktigaste delarna på detaljer och helheten, är en viktig egenskap att erhålla enligt samtliga informanter. Abstraktion används för att hantera komplexitet (Wing, 2008), och därför används det mer och är viktigare inom större företag med mer komplexa projekt och system. Inom mindre företag är projekten simplare, och då bör kraven på abstraktion sänkas. Datalogiskt tänkande består av olika delar där alla delar beror på olika situationsfaktorer. Storleken på verksamheter är en stor faktor som påverkar budgeten, komplexiteten och kraven i allmänhet höjs. Det är viktigt i samtliga sammanhang med datalogiskt tänkande men just i stora verksamheter höjs kraven på hur mycket det skall användas.

Sammanfattningsvis var det flera kategorier som prioriterades mer än andra. Det datalogiska tänkandets kategorier var dem som eftersträvats mest. I den här studien låg fokus på datalogiskt tänkande och att jämföra dem med programmeringskategorierna, vilket var vår grundidé. Resultatet visar att kategorierna från datalogiskt tänkande faktiskt är dem som prioriteras mest av arbetsgivarna inom avgränsningen. Programspråkets kategorier kunde hamna i andra hand då informanterna ansåg att det är någonting som en individ kan läras upp med.

5.4 Datalogiskt tänkande – ”allt” är relaterat

När det samlades in information om fenomenet datalogiskt tänkande var tanken att det var uppdelat på ett visst sätt, och således kunde en individ vara bättre på vissa områden än andra områden. Efter undersökningsfasen förstod vi att det inte är uppbyggt på det sättet. Det är omöjligt att ha en god analytisk förmåga och sedan bristfällig på problemlösning, eftersom en god analytisk förmåga leder till problemlösning. Det gjorde processen att diskutera de ingående kategorierna (tabell 3.3) komplicerat, då de är starkt sammankopplade och påverkar varandra. Abstraktion innebär att man frigör sig från konkreta detaljer och identifierar vilka detaljer som är viktiga och vilka detaljer vi kan ignorera (Wing, 2008), och för att lyckas med det krävs ett analytiskt tänkande, vilket är ett annat viktigt område inom datalogiskt tänkande. Det är ett av flera exempel inom området som gör att det är lämpligt att identifiera datalogiskt tänkande som en egenskap och inte en kombination av flera egenskaper.

Kategorierna ifrån datalogiskt tänkande innefattar en korrelation eftersom samtliga extraherats ifrån det övergripande området. Det intressanta med just det här resultatet är att faktiskt resultatet pekar på en uppdelning mellan dessa kategorier och vissa prioriteras mer än andra. Det svåra är dock att avgränsa dessa kategorier.

Skillnaden finns i att i programspråken existerar spetskompetens, vilket innebär en hög kunskapsnivå inom ett visst område som kan utnyttjas av verksamheter. Det innebär självfallet att det blir lättare att handplocka individer med de kunskaper som krävs för att utföra uppgiften som verksamheten behöver göra. De två områdena är väldigt olika och det är anledningen till att de kan ses på olika sätt. Inom programmering finns det många olika områden och kvalitéer och då skapas möjligheter att skaffa sig mycket kunskap inom ett visst område och är det ett komplicerat och eftertraktat område är det kunskap som blir viktig för en verksamhet.

5.5 De tre stereotyperna

Tre stereotyper ha tagits fram i metodkapitlet med syftet att placera in arbetsgivarnas önskemål i potentiella stereotyper. Dessa stereotyper har delade kategorier som utgår ifrån programspråkets perspektiv och det datalogiska tänkandet. Tabell 3.3 illustrerar dessa stereotyper och kategorier.

Forskningsfrågorna i den här studien innefattar ifall arbetsgivare har en potentiell stereotyp som de letar efter under en rekryteringsprocess, men även under det dagliga arbetet som utvecklare. Som resultatet från den kvalitativa och den kvantitativa metoden är det relativt tydligt att den stereotyp vars egenskaper uppfyller alla kategorier är den som eftersöks. Det är ett självklart resultat som kunnat förutses i ett tidigt skede men dessa egenskaper skiljer sig beroende på vilken prioritetering som arbetsgivaren har. Diagram 4.3 från enkätresultatet illustrerar respondenternas respons kring vilka egenskaper som eftersöks vid en rekrytering.

Individens kunskap och erfarenhet av programspråk är den egenskap som eftersöks mest av respondenterna vilket även överensstämmer med den kvalitativa empirin. Wang (2016) menar att programmering leder till ett datalogiskt tänkande vilket kan tolkas som att programmerare utvecklar det datalogiska tänkandet ju mer dem arbetar med det och därför är kunskapen och erfarenhet kring ett programspråk en egenskap som är eftertraktad.

Den faktor som bryter det här resultatet är just de personliga egenskaperna hos en individ. Alltså de egenskaperna utöver programmeringskunskaper och datalogiskt tänkande. Samtliga respondenter i enkätundersökningen använde sig av ett nyckelord tillhörande en personlig egenskap som t.ex. social, kreativ, självgående och sammarbetsvillig. Dessa egenskaper finns inte i modellen som vi framtagit tidigare i studien utan behandlas särskilt. Dessa egenskaper väger tyngt hos arbetsgivarens preferenser och kombineras med programmeringskunskaper och/eller datalogiskt tänkande.

Kategorier / Stereotyper	Kunskap/Erfarenhet om språk	Strukturerat arbetssätt	Objektorienterad ansats	Helhetslösningar	Analytiskt/logiskt tänkande	Abstraktion	Problemlösning
Datalogiskt tänkande					x	x	x
Programmering	x	x	x				x
Mix	x	x	x	x	x	x	x

Tabell 5.1 Stereotyper och kategorier

Stereotypen som arbetsgivaren prioriterar kan relateras främst till mix stereotypen i tabell 5.1. Alla informanter och respondenter vill i princip ha alla egenskaper som de kan få och inte förhålla sig till just datalogiskt tänkande eller programmeringskunskap. Den mixbaserade stereotypen skiftar dock från arbetsgivare till arbetsgivare. Vissa föredrar abstraktion och problemlösning som är de två stora och nära relaterade egenskaperna i datalogiskt tänkande (Wing, 2008) men även ha den grundläggande programmeringskunskapen hos en individ utöver de personliga preferenser som arbetsgivaren har. Detta gör att utifrån den modell som vi utformat i studien inte direkt kan generaliseras till just en specifik stereotyp eftersom t.ex. en informant i den kvalitativa metoden prioriterade kunskap, strukturerat arbetssätt och de datalogiska egenskaperna men utelämnade den objektorienterade ansatsen.

Det finns alltså ingen specifik stereotyp som är eftersökt utav de tre som tagits fram, utan det är en kombination av arbetsgivarens personliga preferenser som styr vilka egenskaper och kategorier som prioriteras. Mix stereotypen är dock den stereotyp som kan relateras närmast till den insamlade empirin eftersom de flesta informanter och respondenter blandar egenskaperna med ett stort inflytande av de personliga sociala egenskaperna.

En viktig detalj som är värd att nämna som ett resultat i studien är att i enkätsundersökningen fick respondenterna möjlighet att välja mellan en individ med gott datalogiskt tänkande eller en individ med goda programmeringskunskaper men ett sämre datalogiskt tänkande. 70 % svarade den förstnämnda vilket kan leda till en slutsats där respondenterna faktiskt prioriterar ett datalogiskt tänkande hos en individ utan att de faktiskt är medvetna om varken begreppet eller deras omedvetna prioritering.

6 Slutsats

Utifrån resultatet i studien kan slutsatsen dras att personliga egenskaper anses vara de egenskaper som är mest prioriterade vid en potentiell anställning. Arbetsgivaren fokuserar på den typ av individ som den möts av och sedan prioriterar de tekniska egenskaper individen har, eller datalogiskt tänkande. Det betyder att det datalogiska tänkandet eller programmeringskunskaperna kommer i andra hand efter individens personliga egenskaper.

Den stereotyp som eftersöks är en mix mellan de kategorier från datalogiskt tänkande och programspråken men som är styrda av personliga egenskaper eftersom resultatet menar att arbetsgivarna värdesätter upplärning av en individ i efterfrågat programspråk.

De egenskaper som eftersöks av arbetsgivarna är främst de datalogiska kategorierna. Resultatet tyder på att de egenskaper från programmeringen är egenskaper som kan läras ut av verksamheten medan de datalogiska egenskaperna är svårare att lära upp. Resultatet visar att datalogiskt tänkande prioriteras högre, på grund av att datalogiskt tänkande är något en individ har talang för och är svårare att bli upplärd inom.

De egenskaper som eftersöks mest är framförallt problemlösning, både generell hos individen men även inom programmering.

Baserat på resultatet i den här studien kan följande övergripande forskningsfråga besvaras:

- *Vilka personliga egenskaper och kunskaper efterfrågas vid en anställning?*

Utifrån den datainsamling och genomförd diskussion som gjorts, kan en slutsats formuleras som att det är datalogiskt tänkande som är den egenskap som prioriteras framför kunskapen inom ett specifikt programspråk.

6.1 Metodologisk reflektion

Recker (2013) beskriver att blandade metoder stärker studiens generaliserbarhet. Eftersom en blandad metod har valts går det även att diskutera flera kvalitétsfaktorer som t.ex. validitet, reliabilitet, generaliserbarhet och vidareförbarhet.

Generaliserbarheten i den här studien är varierande då studien presenterar en utsträckning av insamlad empiri som ger en grund för att dra slutsatser inom de avgränsningar som gjorts. Inom intervjuerna är generaliserbarheten hög då antalet informanter är tillräckligt många för att samla in tillräckligt med information för studien. Generaliserbarheten i enkäten är däremot tvivelaktig på grund av den låga svarsfrekvensen. Det är anledningen till att majoriteten av den mätbara empirin användes endast som underlag för att stärka empirin från intervjuerna. Dock tillsammans som en blandad metod ges intressenter möjlighet för generalisering av studiens innehåll.

Som Recker (2013) beskriver är vidareförbarhet(eng. Transferability) kvalitétsfaktorn som beskriver ifall studiens resultat kan användas av andra intressenter. Vidareförbarheten av denna studie ger ett generellt värde eftersom resultatet presenteras tydligt och besvarar forskningsfrågorna enligt de avgränsningar som gjorts. Därför är vidareförbarheten god inom avgränsningarna i studien. Resultatet är inte applicerbart eller överförbart till andra studier som inte har med rekrytering att göra eftersom fokus i studien låg på just rekrytering.

Vidareförbarheten i studien kan variera eftersom val av geografiskt område kan påverka resultatet, vilket innebär att resultatet kan vara specifikt för avgränsningen, som är Västra Götaland. Därför kan inte en garanti ges på att det blir samma resultat på andra platser i Sverige. Recker (2013) antyder att forskarna bör argumentera och ge en rik bakgrund över det område som studien applicerats på för att ge en god vidareförbarhet vilket den här studien har gjort med detaljerade beskrivningar och avgränsningar för att tydliggöra studiens relevans.

Reproducerbarheten i den här studien går även att diskuteras. Som Leek och Peng, R (2015) antyder går det aldrig att garantera för en god reproducerbarhet. Den här studien främjar en god reproducerbarhet men kan självklart inte garanteras eftersom den inom avgränsningarna genomförts i Västra Götaland och därför kan inte reproducerbarheten fastställas i andra län och områden.

Enkätsundersökningen hade som syfte att stötta den kvalitativa insamlingen av empiri. Recker (2013) och Bryman och Bell (2011) menar att det bör vara tvärtom, först enkät och sedan styrka enkäten med intervju. Eftersom vi hade fokus på just djupa intervjuer som fokuserade på en rekryterares åsikt och procedurer tyckte vi det var effektivare och pålitligare att låta enkäten stötta upp intervjuerna för att få fram en mer generell åsikt.

Enkäter som förmedlas via internet innebär dock en risk. Vi som forskare kan inte ha full kontroll på vilka respondenter som faktiskt besvarar enkäten. Vi kan endast styra den första kontakten med en potentiell respondent och som vi sedan förlitar oss på att vidarebefordra enkäten till relevanta personer inom rekrytering av mjukvaru-utvecklare eller seniora utvecklare. Eftersom denna risk finns, genomförde vi först intervjuerna för att få en djupare och insiktsfullare empiri för att kunna validera enkätsundersökningen mer effektivt.

Svarsfrekvensen i enkäten var låg. Ungefär 80 mail skickades ut varav 21 respondenter har besvarat enkäten. Detta anser vi vara en låg svarsfrekvens men då studien cirkulerar kring den kvalitativa empirin anser vi att det är acceptabelt med 21 besvarade enkäter med kvalitativa och kvantitativa svar som kan jämföras och relateras med empirin från intervjuerna. Då tidsperspektivet kring studien är relativt kort fanns det inte möjlighet att vänta på fler svar på enkäten.

En del kategorier i studien har fördelar och nackdelar. Vissa kategorier är svåra att avgränsa och jämföra mot varandra eftersom det mesta är relativt abstrakt och leder då till en analys som blir en aning ostrukturerad. Datalogiskt tänkande har sina kategorier och programspråk har även sina unika kategorier. Det är en korrelation mellan de två områdena eftersom den ena behöver den andra. Studien hade kunnat optimeras och möjligtvis ändrat en del kategorier för att göra det tydligare men vi tycker ändå studien har uppfyllt sitt syfte.

Metodvalet gav tillräckligt med information för att genomföra studien, även om antalet kunde höjas både på den kvantitativa och den kvalitativa för att öka validiteten och eventuellt få bredare svar och mer information. Men den grundliga informationen som behövdes för att besvara forskningsfrågorna erhölls och kunde analyseras på ett fördelaktigt sätt. Även om ämnet är väldigt abstrakt lyckades vi med öppna frågor och med precisa följdfrågor precisera deras åsikter och preferenser. Det var av yttersta vikt för att lyckas samla in information inom varje kategori (figur 5.1) för att kartlägga informanternas åsikter om dem och sedan visualisera informanternas olika preferenser.

Tidsfaktorn var en av anledningarna till att antalet respondenter inte var så högt, och svarsfrekvensen var låg och eftersom vi avgränsade till Borås och Västra Götaland fanns det ett begränsat antal företag som kunde besvara frågorna, och därför blev det svårt att få in mer svar. För att effektivisera det här kan enkäter skickas ut i större utsträckning till hela Sverige för att få mer svar, vilket gör hela studien bredare och mer perspektiv fångas och kanske till

och med kan finna skillnader eller liknelser mellan olika områden i Sverige eller det land som studien utförs i.

Den här studien var problematisk att genomföra i och med att begreppet är relativt nytt och att majoriteten av informanterna och respondenterna inte har hört om det innan, vilket gör att svaren blir impulsiva och spontana istället för genomtänkta svar som grundas på deras kunskap inom området. När begreppet blir mer och mer känt efter att det implementeras på fler arbetsplatser, kommer fler individer bli involverade och få kunskap inom området. Det gör att studier som görs i ett senare skede kan få mer genomtänkta svar och kunskap som skapats utifrån att praktiskt och teoretiskt sätta sig in i området.

Om denna studie skulle genomföras på ett annorlunda sätt hade den kvalitativa delen gjorts i en större utsträckning och bredd genom att fler informanter intervjuats. Det mest optimala enligt oss skulle vara att samla in empiri från ett högt antal informanter och sedan endast basera studien på den empirin. Enkäten var i det här fallet en lösning som gav bredare data för att stärka de intervjuer som genomförts.

Andra möjliga förändringar hade varit att bredda avgränsningen till hela Sverige, det hade gjort det lättare att få fler svar på enkäten, då antalet företag som svarade hade ökat, och gett ett bredare resultat och mer information. Det hade inneburit att intervjuerna också gjorts i hela Sverige för att få en validitet på informationen. Det är inte genomförbart med den budget som tilldelats, men framtida forskning borde bredda avgränsningar och eventuellt identifiera skillnader mellan olika platser i Sverige.

6.2 Framtida forskning

Vi hoppas på att denna studie kan leda vidare till en forskning inom datalogiskt tänkande som är ett relativt outforskat område men som tros komma att bli ett viktigt begrepp. Vår studie kan ge en god grund för både Skolverket och andra intressenter inom utvecklingsbranschen men även andra områden där datalogiskt tänkande kan appliceras. Speciellt eftersom vårt resultat visar tydligt att rekryterare av mjukvaru-utvecklare prioriterar just datalogiskt tänkande som en egenskap, utan att ha någon tidigare kunskap om begreppet.

Datalogiskt tänkande är enligt oss ett begrepp som kommer få mer betydelse i framtiden, allt eftersom Skolverket implementerar det i Sverige, och att andra länder i Europa också implementerar det, vilket gör att fler människor får kunskap om det. Denna studie kan vara en god grund för att få en uppfattning om vad det handlar om.

Förhoppningsvis gör vår studie att datalogiskt tänkande uppmärksammas mer och att djupare forskning kan göras inom området, som kommer bli en viktig egenskap att erhålla i framtiden.

Källförteckning

Abelli, B. (2004). *Programmeringens grunder: med exempel i C#*. Studentlitteratur:Lund:Sverige.

Arbetsförmedlingen. (2017). *Dina personliga egenskaper*[Online]
https://www.arbetsformedlingen.se/For-arbetssokande/Tips-och-rad/Personligt-brev/Personliga-egenskaper.html
[Använd:2017-05-04]

Barefootcas. (2016). *Computational Thinking*[Online]
http://barefootcas.org.uk/barefoot-primary-computing-resources/concepts/computational-thinking/
[Använd: 2017-04-01]

Berard, V, E. (2006). *Abstraction, Encapsulation and Information Hiding*[Online]
https://web.archive.org/web/20071214085409/http://www.itmweb.com/essay550.htm
[Använd: 2017-05-10]

Blass, A. Gurevic, Y. (2003). Algorithms: A quest for absolute definitions. *Bulletin of the European Association for Theoretical Computer Science*, s.81. Microsoft:Storbrittanien

Booch, G. (1991). *Object-oriented Design with Applications*. Menlo Park:California

Bryman, A. Bell, E. (2011). *Business Research Methods.* OUP:Oxford

Craig, I, D. (2007). *Object-oriented Languages: Interpratation.* Springer-Verlag:London

CS Unplugged, (2017), *Principles*[Online]
http://csunplugged.org/principles/
[Använd: 2017-05-10]

DN(Dagens Nyheter). (2017). *Skenande brist på programmerare*[Online]
http://www.dn.se/ekonomi/skenande-brist-pa-programmerare/
[Använd: 2017-04-21]

Emanuelsson, G. Johansson, B. Ryding, R. (1991). *Problemlösning.* Lund:Sverige:Studentlitteratur.

Fourment, M. Gillings, M, R, (2008), *A Comparison of Common Programming Languages Used in Bioinformatics.* BMC Bioinformatics

Gov.uk. (2016). *National curriculum in England Computing Programmes Of Study*[Online]

https://www.gov.uk/government/publications/national-curriculum-in-england-computing-programmes-of-study/national-curriculum-in-england-computing-programmes-of-study

[Använd: 2017-04-04]

Grover, S, Pea, R. (2013). Computational Thinking in K–12 A Review of the State of the Field. *Educational Researcher*, *42*(1), ss38-43. Educational Researcher:Stanford:USA

Henderson, P. B. (2009). Ubiquitous computational thinking. *Computer*, *42*(10). IEEE:USA.

Infogram. (2017). *Infogram*[Online]

https://infogr.am/app/#/library

[Använd: 2017-05-02]

Kvale, S. (2014). *Den kvalitativa forskningsintervjun.* Studentlitteratur:Lund:Sverige

Lee, K. (2014). *Foundations of programming languages, undergraduate topics in computer science.* Springer:Tyskland

Leek, J. T., & Peng, R. D. (2015). Opinion: Reproducible research can still be wrong: Adopting a prevention approach. *Proceedings of the National Academy of Sciences*, *112*(6), 1645-1646. The National Academy of Science:Washington:USA

Luhandjula, M, K. Rangoaga, M, J. (2014). An approach for solving a fuzzy multiobjective programming problem. *European Journal of Operational Research*, 232(2), ss.249–255. University of South Africa:South Africa

Microsoft. (2010). *Object-oriented Programming* [Online]
https://msdn.microsoft.com/en-us/library/dd460654(v=vs.100).aspx
[Använd: 2017-04-10]

Microsoft. (2017). *Database Applications*[Online]
https://products.office.com/sv-se/access
[Använd: 2017-04-21]

Microsoft. (2017). *Common Language Runtime(CLR)*[Online]
https://msdn.microsoft.com/en-us/library/8bs2ecf4(v=vs.110).aspx
*[*Använd*: 2017-04-12]*

Naceweb. (2015). *Attributes employers want to see on new college graduates resumes*[Online]
http://www.naceweb.org/s11182015/employers-look-for-in-new-hires.aspx
[Använd: 2017-05-07]

Nygren, H. (2015). Förändra eller dö. *Entreprenör*[Online]
https://www.entreprenor.se/nyheter/forandra-eller-do-digitaliseringen-gar-i-rasande-fart_617780.html
[Använd: 2017-05-07]

Nygård, K. (2015). *Kod, Programmering, Datalogiskt tänkande eller Digital kompetens?*[Online]
http://www.teacherhack.com/2015/09/17/kod-programmering-datalogiskt-tankande-eller-digital-kompetens/
[Använd: 2017-04-10]

Nyman, M. (2010). *Agila metoder – radikal revolution eller enkel revolution?* Wenell Management:Sverige

Recker, J. (2013). *Scientific research in information systems: a beginner's guide,* Springer, New York; Berlin.

Riley, D. Hunt, K. (2014). *Computational thinking for the modern problem solver*. CRC Press:USA.

Ross, J.B. Goodenough, and C.A. Irvine, (1975), Software Engineering: Process, Principles, and Goals, *IEEE Computer*,8(5),ss. 17 - 27. IEEE:USA.

Shamoo, A. E., & Resnik, D. B. (2009). *Responsible conduct of research*. Oxford University Press:Oxford:Storbrittanien.

Skolverket. (2016). Pålsson, S. *Forskning och erfarenheter kring programmering i skolan*[Online]
http://omvarld.blogg.skolverket.se/2016/06/10/forskning-och-erfarenheter-kring-programmering-i-skolan/
[Använd: 2017-03-31]

Sneider, C. Stephenson, C. Schafer, B. Flick, L. (2013). *Computational thinking in High School Science Classrooms: Exploring the Science "Framework" and "NGSS.* Science Teacher, ss.53-59. Teachers Toolkit:USA.

Snyder, A. (1986). Encapsulation and inheritance in object-oriented programming languages. In *ACM Sigplan Notices* (Vol. 21, No. 11, s. 38-45). ACM:USA.

STEM, (2017), *About us[Online]*
https://www.stem.org.uk/about-us
[Använd: 2017-05-10]

Tuttle, B. Butcher, D. (2016). *The most in-demand programming languages on Wall-street.* Efinancialcareers[Online]
http://news.efinancialcareers.com/us-en/197544/demand-programming-languages-wall-street
[Använd: 2017-04-20]

Vetenskapsrådet. (2017). *Etikriktlinjer*[Online]
https://www.vr.se/forskningsfinansiering/sokabidrag/forutsattningarforansokningarochbidrag/etikriktlinjer.4.29b9c5ae1268d01cd5c8000955.html
[Använd: 2017-05-08]

Wang, P. S. (2016). *From computing to computational thinking*. CRC Press:USA.

Webbenkäter, (2017), *Webbenkäter*[Online]
https://www.webbenkater.com/
[Använd: 2017-04-05]

Webbjobb. (2016). *De populäraste programmeringsspråken*[Online]
https://webbjobb.io/programmeringssprak
[Använd: 2017-04-10]

Wing, J. (2008). Computational thinking and thinking about computing. *Philosophical transactions of the royal society of London A: mathematical, physical and engineering sciences*, *366*(1881), 3717-3725.

Wing, J. (2014). Computational thinking benefits society. *40th Anniversary Blog of Social Issues in Computing*, *2014*.

Bilagor

Bilaga 1: Frågor – intervjuer

Här presenteras de förberedda intervjufrågorna som planerats innan intervjuerna. Dock var intervjuerna semi-strukturerade vilket leder till en del följdfrågor som inte presenteras i bilagan. Följdordningen av de ställda frågorna varierade och därför presenteras frågorna utan specifik ordning.

Vad anser du om datalogiskt tänkande, har du hört något om det och vad tror du att datalogiskt tänkande är?
Vad tror du god datalogiskt tänkande har för fördelar?
När ni rekryterar en programmerare, hur är den processen? (Från urval av kandidater till rekrytering) Vad tittar ni på?
Vilken egenskap värderar ni högst hos en individ som är kandidat för en utvecklar-tjänst?(Personliga, Kunskap, erfarenhet osv)
Vilka egenskaper undviker ni/önskar ni hos en individ och varför?
Hur utvärderar ni en kandidat och vad grundar ni urvalet på?
Har ni några tester vid rekrytering? Hur ser testerna ut? (Personlighetstest, kunskapstest, programmeringstest) (*ifall inte besvarad i första frågan)*
Är programmeringskraven i era ansökningar absolut nödvändiga eller ser ni någon framgång i att lära upp individer? Om inte, varför?
Tycker du det är viktigt att programmerare har helhetslösningen i fokus? Varför?
Hur viktigt är det att en programmerare är strukturerad vid sitt arbetssätt som t.ex. att skriva lättläst och lättförståelig källkod eller är det viktigare att bara få uppgiften gjord? *vid svar:* Varför bör det vara så?
Bör en programmerare ha god problemlösningsförmåga för att kunna lösa problem själv eller är det lättast för dem att bara koda och låta andra lösa problem? Varför?(Övergripande problem som t.ex. ändrade krav eller budget)
Abstraktion är ett begrepp som har liknande betydelse inom programmering och datalogiskt tänkande Inom programmering är abstraktion att dölja komplexiteten från användaren genom lager. I datalogiskt tänkande handlar det om att avgränsa problemet och fokusera på det som faktiskt efterfrågas. Är det viktigt att en individ har förmågan att avgränsa och kunna fokusera på det viktiga med uppgiften och vad ger det för fördelar?
Har personliga egenskaper en stor betydelse? (Social, samarbetsvillig)
Analytiskt tänkande är ett begrepp inom datalogiskt tänkande som är hur en individ analyserar och studerar möjliga utfall med problem för att hitta den bästa lösningen. Är analytiskt tänkande viktigt hos en kandidat eller vilka andra egenskaper prioriteras? Hur kan ni upptäcka dessa egenskaper eftersom dem kan verka abstrakta?
Vid utförande av programmering, vilka är de främsta egenskaperna hos programmeraren? (Syntax, algorithm, objektorientering)
Är det viktigt att en programmerare är med i besluten om omfattande problem eller "ska dem bara göra som de blir tillsagda?"
Eftersom datalogiskt tänkande är abstrakt blir det svårt att märka av direkt, märker du efter en lång tid om en person har den problemlösningsförmåga och datalogiskt tänkande som skrevs och hur åtgärdas det?
Vad anser du är de viktigaste delarna inom programmeringsspråket?

Bilaga 2: Frågor – enkät

Har du innan den här enkäten, hört talas om begreppet datalogiskt tänkande?
Hur högt rankar du en individs problemlösningsförmåga vid en rekryteringsprocess?
Hur högt rankar du en individs problemlösningsförmåga under det dagliga arbetet som en utvecklare?
Påverkar individens problemlösningsförmåga möjligheten att få arbete?
Analytiskt tänkande är ett annat begrepp inom datalogiskt tänkande som syftar på förmågan att analysera flera olika utfall och sedan välja det som är rätt för den situationen. Helt enkelt förmågan att läsa av situationer och finna olika lösningar. Hur högt värderar du det analytiska tänkandet?
Abstraktion är ett begrepp inom både datalogiskt tänkande och programspråk. Inom programspråk innebär det att man skall dölja komplexiteten så att användaren inte behöver förstå hur det fungerar för att använda det. Inom datalogiskt tänkande innebär det snarare att man har förmågan att fokusera på det väsentliga och ignorera detaljer. Hur viktig är den egenskapen att avgränsa sig och hålla sig till riktlinjerna?
Tror du att tester vid anställning har någon effekt? Om ja, vilken?
Vilka egenskaper värderar ni högst hos en individ som skall anställas till utvecklare?
Har ni, vid anställning av programmerare, krav på kunskap om ett specifikt programspråk?
Ser ni någon nytta i att lära upp kandidater i ett nytt språk som de inte hanterat innan?
Hur viktigt är det att koden är välstrukturerad och innehåller tydliga kommentarer?
Bör en person ha en god problemlösningsförmåga och kunna lösa problem själv eller föredrar ni en bra programmerare som du själv kan instruera och berätta för vad han ska göra?
Hur viktigt är det att individen har en social förmåga att t.ex. samarbeta och kommunicera på ett effektivt sätt?

Högskolan i Borås är en modern högskola mitt i city. Vi bedriver utbildningar inom ekonomi och informatik, biblioteks- och informationsvetenskap, mode och textil, beteendevetenskap och lärarutbildning, teknik samt vårdvetenskap.

På **sektionen för informationsteknologi** har vi tagit fasta på studenternas framtida behov. Därför har vi skapat utbildningar där anställningsbarhet är ett nyckelord. Ämnesintegration, helhet och sammanhang är andra viktiga begrepp. På sektionen råder en närhet, såväl mellan studenter och lärare som mellan företag och utbildning.

Våra **utbildningar** med huvudområdet informatik är centrerade kring grundläggande begrepp som systemutveckling och verksamhetsutveckling. Inom vårt breda spektrum av inriktningar finns allt ifrån att programmera avancerade system, analysera behov och krav på verksamheter, till att bedriva integrerad IT- och affärsutveckling, dock med gemensamt syfte att verka för god IT-användning i företag och organisationer.

Vid sektionen bedrivs IT-relaterad **forskning** inom högskolans forskningsområde Handel & IT. Forskningsverksamheten är huvudsakligen ämnesmässigt inom **datavetenskap** respektive **systemvetenskap**. Speciella fokusområden är **data science** respektive **information systems science**. Forskningen är både vetenskapligt och professions-orienterad, vilket bland annat tar sig uttryck i att forskningen i många fall bedrivs med grund i domänspecifika verksamhetsbehov, med företag och offentliga organisationer på lokal, nationell och internationell arena. Forskningens professionsinriktning manifesteras också ofta genom vår delaktighet i Swedish Institute for Innovative Retailing (SIIR), som är en centrumbildning vid Högskolan med syfte att bidra till handelsföretag och det omgivande samhället med utveckling av innovativ och hållbar handel.

HÖGSKOLAN
I BORÅS

BESÖKSADRESS: JÄRNVÄGSGATAN 5 · POSTADRESS: ALLÉGATAN 1, 501 90 BORÅS
TFN: 033-435 40 00 · E-POST: INST.HIT@HB.SE · WEBB: WWW.HB.SE/HIT