



**MALMÖ HÖGSKOLA**

Fakulteten för teknik och  
samhälle Datavetenskap

**Examensarbete**  
**15 högskolepoäng, grundnivå**

# En utvärderingsstudie i lärandet av programmering i skolan

An evaluation study of learning programming in school

Gustav Svensson  
Patrik Beijar

Examen: Kandidatexamen 180hp  
Huvudämne: Datavetenskap  
Program: Informationsarkitekt  
Datum för slutseminarium: 2016-08-25

Handledare: Fredrik Rutz



# Sammanfattning

Det har länge varit en debatt om programmering ska ingå i den svenska grundskolans läroplan. I Europa är valet att lägga till programmering i grundskoleutbildningen just nu en av de tydligaste trenderna. Flera länder, så som Danmark och England, har redan på något sätt infört programmering till sin grundskoleutbildning.

I denna studie undersöks hur programmering och datalogiskt tänkande kan undervisas i skolan. Detta görs dels genom en genomgång av tidigare programmeringsspråk som har används till undervisning av programmering, men även genom en observations- och enkätstudie. Observationsstudien genomfördes i Malmö hos verksamheten CoderDojo, en global rörelse som erbjuder alla möjligheten att få testa på programmering. För att få en uppfattning av vad barn och ungdomar har för tankar och idéer om programmering genomförde vi även en enkätundersökning som skickades ut till CoderDojos samtliga verksamheter i Sverige. Resultatet från dessa enkäter gav en bild av vad barn och ungdomar har för tankar om hur programmering kan användas i den svenska skolan.

Nyckelord: Programmering, Datalogisk tänkande, CoderDojo, Skolundervisning, Systematiska observationer, Webbaserad enkät

# Abstract

For some time now, there's been a debate if programming should be included in the Swedish compulsory school curriculum. The choice to add programming in primary education is now one of the clearest education trends in Europe. Several countries, such as Denmark and England, have already in some ways introduced programming to the primary education.

This study examines how programming and computational thinking can be taught in the Swedish schools. We have done a review of past programming language used for teaching programming and computational thinking, but also through observational study and a survey study. The observational study was performed in Malmö at the community CoderDojo, a worldwide movement that offers everyone the opportunity to test programming. To get an idea of what children and teenagers think about programming, we also a performed a survey. This survey was sent out to all CoderDojos operations in Sweden. The results from these surveys give us a picture of what children and teenagers think on how programming and computational thinking might be used in the school.

Keywords: Programming, Computational thinking, CoderDojo, Teaching, Systematic observations, Web-based surveys

<b>1. Inledning</b> .....	<b>7</b>
<b>1.1 Bakgrund</b> .....	<b>7</b>
<b>1.2 Syfte &amp; Frågeställning</b> .....	<b>8</b>
1.2.1 Definition .....	8
1.2.2 Syfte .....	8
1.2.3 Frågeställning. ....	9
<b>2. Tidigare forskning</b> .....	<b>10</b>
<b>2.1 Tidigare Debatter.</b> ....	<b>10</b>
<b>2.2 Tidigare studier.</b> ....	<b>11</b>
2.2.1 Förenklad programmering.....	11
2.2.2 Grafisk programmering.....	13
2.2.3 Mekanisk programmering. ....	16
2.2.4 En svensk satsning. ....	17
<b>2.3 Slutledning.</b> ....	<b>17</b>
<b>3. Metod</b> .....	<b>19</b>
<b>3.1 Metodbeskrivning</b> .....	<b>19</b>
<b>3.2 Metoddiskussion</b> .....	<b>21</b>
<b>3.3 Genomförande</b> .....	<b>21</b>
3.3.1 Systematiska observationer.....	21
3.3.2 Webbaserad enkät.....	22
<b>4. Resultat</b> .....	<b>23</b>
<b>4.1 Sammansatt resultat av alla observationer</b> .....	<b>23</b>
<b>4.2 Sammansatt resultat från webbaserad enkät.</b> ....	<b>24</b>
<b>5. Analys</b> .....	<b>30</b>
<b>5.1 Analys av systematisk observation</b> .....	<b>30</b>
<b>5.2 Analys av webbaserad enkät</b> .....	<b>31</b>
<b>6. Diskussion och Slutsats</b> .....	<b>33</b>
<b>8. Referenslista</b> .....	<b>35</b>
<b>9. Bilagor</b> .....	<b>37</b>
<b>9.1 Svar utbildningsdepartementet från den öppna enkäten</b> .....	<b>37</b>
<b>9.2 Observationsschema</b> .....	<b>38</b>
<b>9.3 Observationsdata</b> .....	<b>39</b>
<b>9.4 Enkäten</b> .....	<b>44</b>

*”Jag tycker att skolorna ska ställa upp och bjuda oss på den kunskapen vi faktiskt borde ha om teknik”. - Elev på Sjöstadsskolan.*

# 1. Inledning

I detta avsnitt tar vi upp bakgrunden till vårt ämne, vad för syfte vi har med vår studie samt vad för frågeställningar vi har valt att använda till denna studie. Vi definierar även vissa ofta förekommande begrepp.

## 1.1 Bakgrund

Under september år 2015 gav den svenska regeringen ett nytt uppdrag till skolverket. Skolverket fick i uppdrag att ta fram en ny IT- och digitaliseringsstrategi där även ämnet programmering ska ingå i grundskolans utbildningsplan. För grundskolan ska kurs- och läroplaner ses över för att förstärka och tydliggöra programmering som ett inslag i undervisningen.[1]

Tanken bakom att implementera utbildning inom programmering handlar inte enbart om att lära unga elever att skriva kod. Problemformulering och -lösning, kreativitet och logiskt tänkande är exempel på centrala förmågor som intressenter bakom uppdraget vill ska ingå i unga elevers meriter. Dessa förmågor kan även bidra positivt till inlärning inom andra skolämnen.[1]

Idén till uppdraget från regeringen har inspirerats av en svensk skolpolitisk debatt som pågått sedan år 2012. Hacka läroplanen! Så lyder projektnamnet som genomfördes av Karin Nygård och Terese Raymond, två prisbelönta lärare och pedagoger. Teacherhack är en ideell förening som startades för att ha en gemensam utgångspunkt för att arbeta med programmering och digital kompetens i skolan. Projektet Hacka läroplanen! väcktes av samtal angående att lgr11 (läroplan för grundskolan, förskoleklassen och fritidshemmet) måste uppdateras för att inkludera digitala kompetenser i skolundervisningen.[2]

En stor bakomliggande tanke med Teacherhack och den nya läroplanen är att unga elever ska kunna gå från digitala konsumenter till producenter. Att genom lärandet av programmering och dess roll i dagens teknik, förstå tekniken vi dagligen använder. Om man lär sig mer om hur dagens teknik fungerar och arbetar behöver man inte automatiskt acceptera att vissa funktioner är som de är. Att man själv kan förändra och förbättra. Nu håller Teacherhack på att ta fram material till läroplanen om hur man kan använda programmering i teknikämnet för årskurs 7-9. De hoppas att det kommer leda till att ämnet får en högre status inom skolutbildningen.[3]

Teacherhacks intresse för datalogi och programmering inom grundskolan är ett ämne som tagits upp bland annat inom Sveriges radio, Internetstiftelsen i Sverige och Svenska dagbladet. Nu när regeringen har uppmärksammat intresseutvecklingen är det av stort värde att både se över vad andra länder i världen gör för arbete inom ämnet och vad som gjorts tidigare.

I oktober år 2015 släppte Europeiska skoldatanätet den andra upplagan av "Computing Our Future", en rapport där utbildningsministrar från 21 europeiska länder har fått svara på en enkät om hur ämnen som datalogisk tänkande, programmering och digital kompetens kan ingå i läroplanen [4, 5]. Av 21 stycken länder har 16 av dem, exempelvis Danmark och England, lagt till programmering i deras läroplan. Rapporten nämner att en av de tydligaste trenderna inom skolområdet i Europa är valet av att lägga till programmering i läroplanen.

I mars år 2015 släppte medlemsorganisationen IT&Telekomföretagen en rapport[6] där de menar att IT- och telekomsektorn har stora kompetensbrister och beräknar ett underskott på 60 000 personer redan år 2020. De har kommit fram till denna siffra då efterfrågan på kompetens är likvärdig nu som det var år 2012, då

IT&Telekomföretagen senast kartlade kompetensbehovet i IT- och Telekomsektorn [6]. Rapporten presenterar olika lösningsförslag som både på kort och lång sikt ska motverka denna kompetensbrist. De nämner exempelvis att politiker måste ta digitaliseringsfrågor på större allvar för att Sverige inte ska halka efter på området. De menar att digitaliseringsfrågan inte fått stor uppmärksamhet dels för att ämnet är nytt och därmed att kunskap saknas, men även för att det inte är en fråga som ett parti vinner fler röster på och blir då naturligt lågprioriterad.

En annan anledning är att Sveriges internationella ranking för tillfället är god och att svenska it-bolag är i framkant på den internationella marknaden. Detta kan tolkas som att digitaliseringens utveckling sköter sig själv, helt utan politikernas hjälp, vilket IT & Telekomföretagen förutspår inte kommer fungera i längden [6].

En ytterligare långsiktig lösning är att skolan ska digitaliseras fullt ut. Denna lösning har delats upp i tre kategorier: lärares tillgång till och kompetens kring digitala lärarresurser, elevers förmåga att orientera sig i ett digitaliserat samhälle samt elevers lärande inom programmering och digitalt skapande[6]. Det är den sistnämnda punkten vi har valt att fokusera vår studie på.

## 1.2 Syfte & Frågeställning

### 1.2.1 Definition

#### **Datalogisk tänkande**

Datalogiskt tänkande (*"computational thinking"*) är ett paraplybegrepp för färdigheter och förmågor relaterade till problemlösning som till stor del kommer från datavetenskapen. Till exempel att bryta ner ett problem i mindre delar, att hitta och utnyttja mönster, att automatisera lösningar genom att utveckla algoritmer, och att representera och modellera information.

#### **Programmering**

För att skapa tydlighet i uppsatsen definierar vi programmering som handlingen av att sammanställa en uppsättning symboler och/eller element som representerar en aktion eller beräkning utförd av en dator. Med hjälp av dessa symboler kan en användare uttrycka sina avsikter till en dator och med hjälp av kunskaper om symboler kan en använda att förutse handlingar av en dator. Denna definition av programmering exkluderar programmering genom demonstration, där datorn observerar användarens åtgärder och använder interna uträkningar för att generera ett program åt användaren.

### 1.2.2 Syfte

Det har blivit vanligt att hänvisa ungdomar som "digitala infödingar" på grund av deras exponering med digital teknik [8]. Det har blivit ett faktum att många unga människor är mycket bekväma med att skicka textmeddelanden, spela online-spel och konsumera information på internet via valfri webbläsare. Men även om de interagerar med digital media i sitt dagliga liv är det bara ett fåtal som kan skapa till exempel sina egna spel, animationer eller uträkningar och funktioner.

Förmågan att kunna programmera och kunskap om datavetenskap kan ge flera fördelar. Exempelvis kan det kraftigt utöka möjligheterna för vad du kan skapa (och hur du kan uttrycka dig själv) med en dator. Det expanderar även området om vad du kan lära dig om teknik. Särskilt programmering ger stöd för datalogiskt tänkande, vilket hjälper dig i viktiga problemlösning- och designstrategier (som modulering och

iterativ design) som sedan kan föras över och som hjälpmedel i icke programmeringsområden [9].

Ett sistnämnt syfte är att tidigt kunna investera i ungas kunskaper om teknik och datalogiskt tänkande för att ha en chans att fylla det underskott som redan förutspåtts till 60 000 personer i IT-branschen till år 2020. Att ge unga en chans att skapa ett intresse för området kan generera en hållbar långsiktig investering för Sverige och svensk teknik.

Vi utför en djupgående analys i för och nackdelar med att implementera regeringens uppdrag i svensk skolundervisning samt hur lärandet av programmering och datalogiskt tänkande kan utföras. Analysen utförs med stor hjälp av tidigare forskning och med egna observationer och enkätstudier på unga människor som lär sig programmering.

### 1.2.3 Frågeställning.

Vår huvudfrågeställning i denna studie är:

*Vad finns det för syfte att unga människor ska skapa sig ett datalogiskt tänkande?*

Studien innehåller även två stycken underfrågeställningar, vilka är följande:

- Är det nödvändigt att inkludera datalogiskt tänkande i skolundervisningen?
- På vilket sätt kan datalogiskt tänkande undervisas?

## 2. Tidigare forskning

I detta avsnitt tar vi tidigare debatter och studier inom ämnet programmering i skolundervisningen. I avsnittet nämner vi de programmeringsspråk som har används i syfte för att undervisa programmering. Avsnittet har även som uppgift att besvara vår frågeställning ” På vilket sätt kan datalogiskt tänkande undervisas?”

### 2.1 Tidigare debatter.

För att få en insikt i hur processen går till vid förberedning av ett nytt ämne i skolan skickade vi ut en öppen enkät till utbildningsdepartementet och till kommunpolitiker med ansvarsområde utbildning.

De frågor vi ställde i den öppna enkäten var följande:

1. Vad är statusen på att införa programmering som ämne i den svenska grundskolan? Är detta något som är planerat eller är det oklart vilken tid detta kommer införas?
2. Hur fungerar arbetet kring att utforma en kursplan vid ett ämne som för många är nytt? Hur förbereder man skolor och lärare?
3. Vad ser du för fördelar att införa programmering i grundskolan? Är det endast fördelar eller finns det även nackdelar?

Syftet men den första frågan var att få klarhet i vad som är planerat från regeringen angående införande av programmering i skolan, då vi inte kunde finna denna information på någon av regeringen eller skolverkets hemsidor.

I samtliga svar fick vi informationen att programmering inte ska införas som ett enskilt ämne, utan istället användas som ett verktyg i de redan befintliga ämnena. Vilka dessa ämnen kan tänkas vara är ännu inte bestämt.

Eftersom vi fick informationen i fråga ett om att programmering icke ska agera som ett enskilt ämne blev fråga nummer två mindre relevant. Frågan besvaras ändå från utbildningsdepartementet, de skriver i sitt svar till fråga två i enkäten följande:

*“Ändringar i läroplan och kursplaner behöver som regel föregås av någon form av implementeringsinsatser riktade till lärare, rektorer, huvudmän etc. “*

Utbildningsdepartementet svar i helhet finns som bilaga 9.1.

Vid frågan om för- och nackdelar var det övervägande fördelar som togs upp. Många tog upp vikten om att ha en läroplan som är anpassad för dagens samhälle för att på detta vis förbereda barn och ungdomar för de jobb och utmaningar som kan tänkas vara aktuella de kommande åren. En ytterligare fördel som nämndes var att programmering och datalogisk tänkande ger fler möjliga tillfällen för kreativt arbete, vilket ses som positivt. Exempel på nackdelar som togs upp var att ämnet är för många nytt och att då det kan ta tid att skapa en stadig läroplan med lärare som besitter rätt kompetens.

Vid debatter kring att införa programmering i skolan nämns både för- och nackdelar vid förslaget. Fördelar som nämns är att genom att lägga till fler datavetenskapliga ämnen så uppdateras läroplanen och därmed speglar dagens samhälle bättre. Att programmering kan användas som verktyg till elever för att exempelvis lättare förstå matematik är ännu ett argument som tas upp. Syftet med att införa programmering i skolan är inte för att göra alla elever till potentiella programmerare, utan ge eleverna en bättre förståelse för IT i allmänhet. Nackdelar som tas upp i debatten är exempelvis frågan om varför just programmering är viktigare än andra ämnen som också kan

tyckas kvalificera sig in i läroplanen. Varför ska man lära sig om programmering när eleverna likaväl kan lära sig ett tredje språk?

## 2.2 Tidigare studier.

Datalogiskt tänkande ("*computational thinking*") är ett paraplybegrepp för färdigheter och förmågor relaterade till problemlösning som till stor del kommer från datavetenskapen. Till exempel att bryta ner ett problem i mindre delar, att hitta och utnyttja mönster, att automatisera lösningar genom att utveckla algoritmer, och att representera och modellera information. En gemensam nämnare är att dessa färdigheter och förmågor naturligt tränas genom programmering (men de kan även tränas på andra sätt) [10].

Sedan tidigt 60-tal har forskare skapat ett antal programmeringsspråk och miljöer vars ändamål har varit att göra programmering mer inbjudande och åtkomligt till en större skala människor. Det fanns tidigt en entusiasm för att undervisa unga elever i datavetenskap och programmering. Medan Sverige införde projektet med COMPIS i svenska skolor, lärde tusentals amerikanska skolor sina studenter att skriva enkla program i Logo eller Basic. Seymour Papert, pionjär inom artificiell intelligens och matematiker på MIT, presenterade i sin bok från år 1980 Logo som en hörnsten för nytänkande metoder till utbildning och lärande.[11]

Vi har studerat alla de programmeringsspråk som tagits fram för undervisning av programmering i skolor. Syftet med att studera dessa initiativ har varit att få svar på vår frågeställning: *På vilket sätt kan datalogiskt tänkande undervisas?*

I studien samlade vi in alla de framtagna programmeringsspråken och delade upp dessa i tre kategorier. Gemensamt för alla programmeringsspråk i dessa tre kategorier är att de utformades med målet att hjälpa människor att lära sig programmering och datalogiskt tänkande. De flesta språken är enkla programmeringsverktyg som skulle ge nybörjare en introduktion till grundläggande aspekter av programmering. Efter att studenterna fått erfarenhet av inlärningsspråket förväntades de att övergå till ett mer generellt och kommersiellt språk.[12]

Ur vardera kategori presenterar vi några av de projekt och programmeringsspråk som exempel, sammanfattar dessa och sedan analyserar deras strategier. Designstrategier som de har valt för att göra programmering mer tillgängligt för nybörjare. Dessa tre övergripande kategorier är följande: *förenklad programmering*, *grafisk programmering* och *mekanisk programmering*.

### 2.2.1 Förenklad programmering.

Inom denna kategori har forskarna använt generella programmeringsspråk som grund och inspiration, för att sedan förenkla dessa.

Systemen inom denna kategori undersökte tre metoder för denna förenkling.

1. Förenkling av språket, att skapa en förenklad version av ett tidigare etablerat språk.
2. Skräddarsy språket för ett specifikt område inom programmering.
3. Förebygga och minimera risken för att användarna ska kunna göra syntaktiska fel.

Systemen i denna kategori är uppbyggda kring hypotesen om att den primära barriären i att lära sig programmet ligger i mekaniken i att skriva program. För att

framgångsrikt kunna skriva ett program, måste användarna förstå ett antal ämnen. De måste förstå hur man uttrycker instruktioner till datorn (syntax), hur man organiserar dessa instruktioner (t.ex. programmeringsstil) och hur datorn utför dessa instruktioner.[12]

Systemen i denna kategori försöker göra det lättare för nybörjare att lära sig dessa ovanstående färdigheter. Som tidigare nämnt förväntas elever övergå från introduktionsspråken till ett mer generellt programmeringsspråk, så här är det flesta språken, med avsikt, likt ett tidigare etablerat programmeringsspråk.

### **BASIC.**

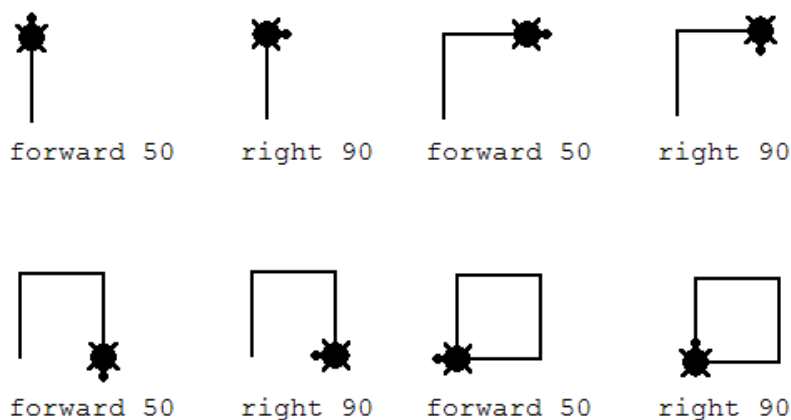
Det första initiativet till ett sådant programmeringsspråk var *BASIC* (Beginners All-purpose Symbolic Instruction Code) från Dartmouth College och utvecklades år 1963 [13]. Basic var designat för att lära icke vetenskapliga studenter om datoranvändning genom programmering. De vanligaste språken under den tiden var både omfattande och komplexa. Skaparna av BASIC, Kemeny och Kurtz, förutsåg att studenter skulle bli överväldigade av alla detaljer och möjligheter i språken och syntaxen. Som en lösning på detta designades BASIC för att stödja bara en liten uppsättning instruktioner och med en så enkel syntax som möjligt.

En instruktion i BASIC består av tre delar. Ett radnummer, en operatör (exempelvis *LET*) och en operand (exempelvis  $B = A + 1$ ). Alla kommandon började med ett engelskt ord för att göra språket lättare för nybörjare. Designerna var övertygade att  $LET B = A + 1$  skulle vara lättare att förstå för elever än  $B = A + 1$ .

### **LOGO.**

Ett annat tidigt programmeringsspråk som utformades för att lära ut programmering var *LOGO*. Logo designades år 1967 av Daniel G. Bobrow, Wally Feurzeig, Seymour Papert och Cynthia Solomon [14]. Målet med Logo var att skapa ett språk där barn kunde leka med ord och uttryck. Att låta barn utforska olika ämnesområden som matematik, naturvetenskap, språk och musik var även detta ett mål från Logo [15]. Programmeringsspråket Logo är en dialekt av Lisp med en förenkling av syntaxen för att göra språket mer tillgänglig för nybörjare.

Den mest kända delen av Logo är sköldpaddan (Logo turtle) som börjades som en robotsköldpadda som kunde rita på marken. Roboten ersattes senare av en digital aktör som kunde flytta, vända och rita linjer i en tvådimensionell grafisk värld. Genom att ge sköldpaddan instruktioner som "forward 30" kan studenten få sköldpaddan att röra sig i sin egna framåt riktning.



© 2000 Logo Foundation

Figur 2. Exempelbilder på den programmeringsbara Logosköldpaddan som ritade linjer.

Många barn blev introducerade till programmering genom att få sköldpaddan att rita enkla bilder. Men språket Logo har stöd för ett bredare utbud av möjligheter. Det fanns skolklasser av barn som skrev program som översatte ord från två språk och program som kunde spela upp melodier [16]. Logo blev senare en föregångare och inspiration till en rad av initiativ till programmeringsspråk för barn och nybörjare [17].

### Junior Java.

Ett senare initiativ till att förenkla den då redan etablerade formen av programmering (textbaserat) och ett initiativ till förenkling av ett etablerat programmeringsspråk var JJ (Junior Java) från California State University och California Institute of Technology. I designen av JJ togs mycket komplexiteten av Javas syntax bort för att hjälpa studenterna att kunna fokusera på själva konceptet av programmering utan att känna sig överväldigade av syntaxen [18]. I JJ tog man bort deklARATIONER som semikolon (som instruerar att en instruktion är slut) och måsvingar (curly brackets), samt att man bara kan utföra en sak på ett sätt. Det sistnämnda innebar till exempel att det bara fanns en typ av integer och ett tillvägagångssätt att skriva kommentarer i koden.

Computing weekly pay in JJ:	The same code in Java:
<pre>If (hours &lt;= 40) then   Set pay = 10 * hours Else   Set pay =     400 + 15*(hours - 40) EndIf  Output "The pay is " Outputln pay</pre>	<pre>if (hours &lt;= 40) {   pay = 10 * hours; } else {   pay =     400 + 15 * (hours - 40); } // EndIf  System.out.print ("The pay is " ); System.out.println( pay );</pre>

Figur 3. Exempelkod på en if-sats i Junior Java och samma exempel på samma if-sats i Java.

JJ designades även med ett stöd för att migrera sin kod till Java. Tanken var att eleverna skulle göra det med sin kod efter en halv termin. Antingen skulle de översätta sin kod från JJ till Java för hand eller så kunde utvecklingsmiljön konvertera deras kod automatiskt.

Men på grund av dålig adoptering av JJ gick utvecklarna vidare till att förbättra studenters användarupplevelse av Java istället, genom tydligare error-meddelanden från kompilering och möjlighet för att studenterna kunde programmera Java på webben.

### Processing.

Processing skapades år 2001 av Casey Reas och Benjamin Fry, båda studenter från MIT Media Lab. Från början var det tänkt att Processing skulle vara ett nybörjarprogrammeringsspråk, med inspiration från både BASIC och Logo. Sedan utvecklades Processing till att fungera som en "digital skissbok" och för lära sig grunderna inom programmering genom en visuell miljö med bilder, animationer och olika typer av interaktioner. Processing är uppbyggt genom programmeringsspråket Java, men använder sig av en grafisk miljö och förenklade syntaxer.[19]

## 2.2.2 Grafisk programmering.

Grafisk programmering har experimenterats som ett alternativ till den skriftliga programmeringen. Trots tidigare försök att göra programmeringsspråk enklare och

lättförståeliga har många nybörjare fortfarande svårt med syntax. Till exempel att komma ihåg namnen på kommandon, ordningen av parametrar, om de ska använda måsvinge ( { } ) eller klammer ( [ ] ) och så vidare.

Inom grafisk programmering skapar man instruktioner till en dator genom sammansättning av grafiska symboler och element istället för text. Grafiska symboler som representerar funktioner, uträkningar och värden. Designvalet i att göra programmering grafiskt är uppbyggt kring idén om att göra det möjligt för nybörjare att förstå vad programmering egentligen är, genom att kringgå syntaxproblemet.

Denna kategori representerar tre huvudriktningar för att kringgå syntax:

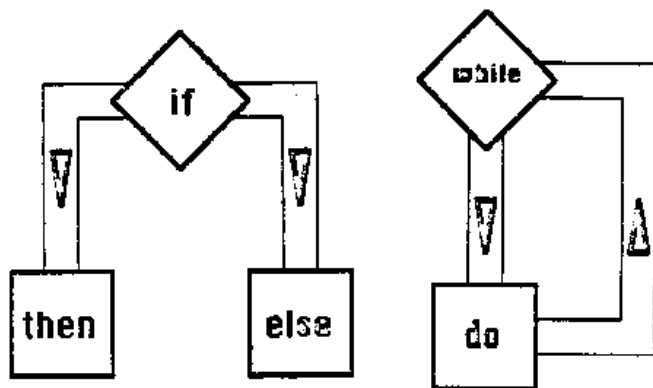
- Att skapa objekt som representerar kod.
- Objekten ska kunna flyttas runt och kombineras på olika sätt av användaren inom gränssnittet.
- Skapa program genom användarens val inom gränssnittet.

Nybörjarprogrammerare behöver endast känna igen namnen på kommandon och syntaxen är inkodat i formerna och färgerna hos objekten, vilket skulle ha förhindrat dem från att skapa syntaktiskt felaktiga program.

### PICT

Ett av det första grafiska programmeringsspråk var *PICT*. *PICT* skapade Ephraim Glinert och Steven Tanimoto på *University of Washington*. *PICT* tillåter nybörjare att skapa enkla program genom att koppla samman grafiska ikoner som representerar kommandon.

*PICT* tillåter användare att skapa program som gör enkla numeriska beräkningar med addition och subtraktion av tal, variabeltilldelning och Booleska tester. Genom att användaren väljer ut relevanta ikoner (kommandon) från en meny och ansluter ikoner tillsammans skapar användaren ett program. Dessa ikoner väljs ut och positioneras på ett gränssnitt med hjälp av en joystick.[20]



Figur 4. Exempelbild på gränssnittet i *PICT*.

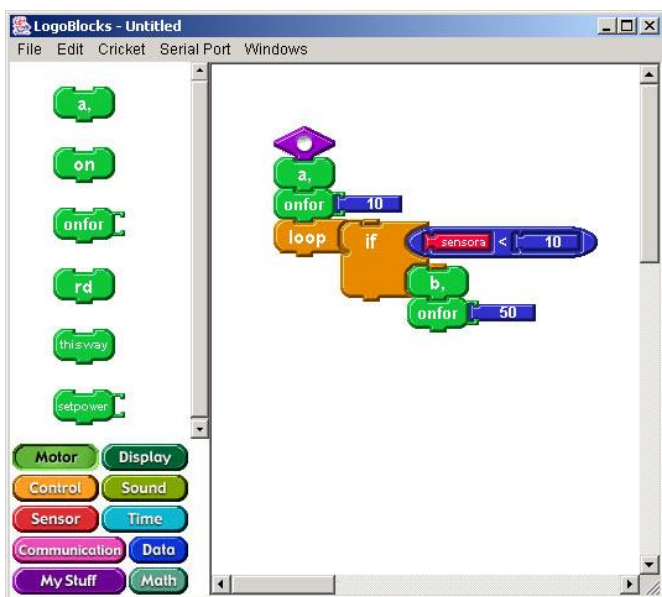
När en användare kör ett program animerar *PICT* genomförandet av programmet genom att flytta en vit ruta längst banan av ikoner som använder kopplat ihop. Användare kan köra en *PICT* program vid valfri punkt under sin utveckling. Om användarens program når en punkt där dess beteende inte har specificerats, kommer den att stanna och anmäla till användaren att ytterligare programmering är nödvändig.[20]

### LogoBlocks

LogoBlocks utvecklades av MIT Media Lab år 1996. LogoBlocks designades för den programmeringsbara mikrodatorn Brick, en föregångare till det kommersiella Lego Mindstorms-systemet som lanserades år 1998.[11]

I LogoBlocks, som även är en förlängning av Logo, representerades kommandon som märkta, grafiska former och dessa former kunde användarna sammansätta och kombinera för att skapa program till Brick.

Dessa grafiska block kunde dras ut från ett menyfält vid ena sidan av gränssnittet och placeras på en arbetsyta, där det kunde placeras tillsammans med andra block för att skapa ett program.



Figur 5. Ett LogoBlock program som starar motor A under 10 sekunder om en ljuskälla aktiveras samt aktiverar en loop så länge en if-sats uppnås.

Kommandon och villkor har flera former, blocken i menyfältet representerar olika typer av objekt snarare än alla tillgängliga objekt. Kommandon och villkor som kräver argument har former med utskärningar för att placera argumentet. Genom att se formen på blocket och utskärningen kan användaren se vilka typer av argument och kommandon som passar ihop.

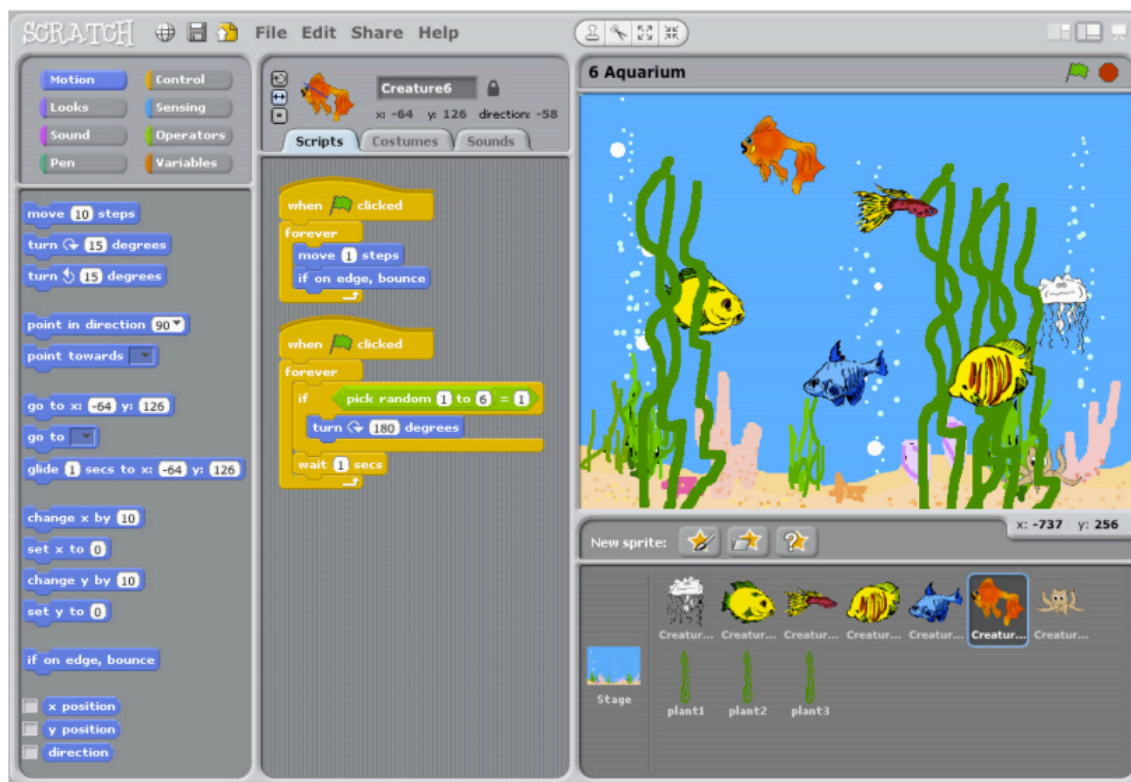
För att göra förändringar i existerande program kunde användarna flytta befintliga uttalanden i programmet för att göra plats för nya. Eller genom att markera existerande block, till exempel "a"-blocket, kan användaren genom vänstermenyn ändra de till ett annat block.

### Scratch

Scratch är ett grafiskt programmeringsspråk som har influerats av LogoBlocks. Mitchel Resnick och hans forskarteam på MIT började utveckla Scratch år 2003, fyra år senare, släpptes den första version av Scratch. I Scratch skapas projekt som innehåller olika typer av media och programmeringsskript. Programmering sker likt LogoBlock, genom att koppla ihop färgglada kommandoblock. Från början var det tänkt att Scratch skulle finnas vid så kallade "efter-skolan datacenter" för barn och ungdomar i åldrarna 8 till 16. [21]

Genom att blanda programmering med olika typer av medier blev Scratch snabbt populärt och används idag i exempelvis skolor och bibliotek. Enligt Mitchel Resnick är

huvudmålet med Scratch att introducera programmering för människor utan tidigare programmeringserfarenhet. Scratch är gratis och finns tillgängligt online och som nedladdningsbart program. I oktober 2016 var det cirka 14 miljoner registrerade användare och 17 miljoner projekt som har delats.[21]



Figur 6. Scratchs programmeringsmiljö.

### 2.2.3 Mekanisk programmering.

Ett mekaniskt programmeringsspråk tillåter användaren att förflytta och organisera fysiska objekt för att uttrycka ett program, istället för att skriva in kommandon eller förflytta grafiska objekt i en digitalmiljö. I de vissa fall måste de slutgiltiga resultatet av programmet ses på en datorskärm. I andra fall kan resultatet av programmeringen finnas tillgänglig utanför datorn, som till exempel system med leksakståg, en kommersiell produkt från Legos Intellistudio.[22]

#### **TORTIS.**

TORTIS systemet, skapades av Radia Perlman år 1976. Radia Perlman blev en pionjär i sitt arbete med Tortis och dess mål med att lära unga barn (mellan 3 till 5 år) programmering [22]. Tortis systemet bestod av fysiska knapplådor som tillät användaren att styra en robotsköldpadda, inspirerad av Logosköldpaddan. Det fanns fyra uppsättningar knapplådor som gavs stegvis till användaren.

Den första lådan innehöll knappar som kunde ge sköldpaddan instruktioner att förflytta sig, vända, plocka upp och sätta ner en penna, tända och släcka en lampa samt tuta. Den andra lådan gav användaren möjligheten att upprepa instruktioner från första lådan. Med den tredje lådan öppnade upp möjligheter för användaren att spara sina program till sköldpaddan och sedan spela upp de sparade programmen [22]. Systemet med knapplådor tillät dock inte användaren att kunna ändra ett program efter att man sparat dem, vilket gjorde det svårt att gradvis utveckla ett program.

## 2.2.4 En svensk satsning.

Som en intressant notis hittade vi en tidig svensk satsning till att lära ut programmering. Satsningen var projektet ”COMPIS – Computer i Skolan” av Tudis, år 1981 (Teknikupphandlingsprojekt Datorn i Skolan). Målet var att skapa en dator som skulle vara anpassad för den svenska skolan. Tudis ville främst att COMPIS skulle ha ett lättlärt programmeringsspråk och samtidigt kunna fungera som ett bra läromedel.

Programmeringsspråket som rekommenderades till undervisning i COMPIS var *Comal*. Comal beskrivs som en blandning av den tidens populära utbildningsprogrammeringsspråk, BASIC och Pascal. Comal utvecklades i Danmark av Benedict Løfstedt och Børge R. Christensen [23]. Språket valdes under 80-talet till ‘Svensk utbildningsstandard’. Beslutet var omtvistat då Comal, trots att det passar utmärkt för utbildning, var ovanligt utanför skolväsendet.

Det var många faktorer som spelade in för att klassa COMPIS som ett misslyckande. Det största var att under tiden som COMPIS utvecklades och producerades hade omvärlden sprungit ifatt. År 1981 släppte IBM operativsystemet MS-DOS (Microsoft Disk Operating System), ett operativsystem som COMPIS inte var kompatibel med. Det var detta operativsystem som eleverna skulle mötas av på arbetsmarknaden, vilket resulterade i att det eleverna lärde sig med COMPIS blev oanvändbart utanför skolan. COMPIS-projektet klassas som ett stort misslyckande och kostade svenska skolor 120 miljoner under tre års tid.[24]

## 2.3 Slutledning.

Många av de tidigare initiativen, som till exempel Logo och PICT, levde aldrig upp till sina förväntningar och löften. Tidigare studier har lyft fram ett flertal faktorer som anledningen till de misslyckade satsningarna på att lära unga människor programmering

Tidigare forskning utsåg att de tidiga programmeringsspråk var alltför svåra att använda och att många elever inte klarade av, eller orkade, att bemästra programmeringsspråkets syntax, även om de hade förenklad syntax eller syntax som skulle likna det engelska språket.[25]

Programmeringsspråket var även ofta introducerat tillsammans med specifika aktiviteter (som att göra listor med primtal eller skapa enkla bilder med linjer) som inte kunde kopplas till unga studenters intressen, erfarenhet eller behov. Bristande kunskap hos lärarna har ansett vara en misslyckande faktor då programmering infördes i utbildningen när ingen kunde ge vägledning när saker gick fel eller uppmuntra till djupare lärande när det gick rätt.

Som vidare forskning har våra källor hävdad att ett programmeringsspråk som är riktat till att lära ut programmering bör ha ett “lågt golv”, det vill säga att det ska vara lätt att komma igång med och en “högt i tak” möjlighet. Högt i tak innebär att det ska finnas stöd till att skapa allt mer komplexa projekt över tid. Dessutom bör språket ha “breda väggar”, att det ska stödja många olika typer av projekt så att studenter med olika intressen och inlärningsstilar kan engagera sig i lärandet.

Att kunna uppnå denna trippelt av krav för ett programmeringsspråk är dock allt annat än enkelt.

En problematisering med de tidigare forskningsinsatserna är deras fullständiga fokus på programmeringen som ägde rum, utan att titta in i de pedagogiska processer som ägde rum samtidigt eller de utbildningsmiljöer som programmeringen var tänkt att appliceras på. Dokumentationen rapporterar oftast bara barns "framgångsrika" användning av systemen eller hur barn lyckades utföra en specifik aktivitet, fristående från en skolkontext och barnens egna intresse.

## 3. Metod

I detta avsnitt beskrivs och motiveras studiens tillvägagångssätt för insamling av data. Här beskrivs hur datainsamlingen genomfördes samt vilka beslut som var avgörande för studien. De forskningsmetoder som vi valde för insamling av data som kan hjälpa oss att besvara våra frågeställningar är systematiska observationer och webbaserade enkäter.

För att undersöka hur unga människor interagerar med programmering valde vi att vända oss till organisationen CoderDojo. CoderDojo beskriver sig själva som en rörelse för att lära barn och unga att programmera. CoderDojo började i Malmö år 2012 och det är gratis att delta och är öppet för alla barn mellan 7 och 17 år. Hos CoderDojo får barnen lära sig att programmera, göra appar, spel och webbsidor. De behöver inga förkunskaper innan de deltar på CoderDojo [7].

CoderDojo har volontärer som agerar som mentorer för barnen, de flesta jobbar som programmerare. Vid den tidpunkten som vi utförde vår studie på CoderDojo hade de över 20 elever uppdelade i två grupper baserade på ålder, tonåringar och yngre. För att handledarna ska kunna hjälpa samtliga deltagare hade de för tillfället tre typer av kategorier som deltagarna kunde välja att lära sig [7].

Dessa tre kategorier var:

- Programmera spel och animationer med Scratch
- Bygga webbsidor med HTML, JavaScript och CSS
- Programmera spel och appar med Processing

Många av CoderDojos verksamheter använder Scratch då detta är ett beprövat visuellt programmeringsspråk och en bra introducering till programmering.

### 3.1 Metodbeskrivning

#### 3.1.1 Systematiska observationer

Vid användning av observation som metod för datainsamling används huvudsakligen två stycken typer av observationsmetoder, *systematisk* och *deltagande observation*. [26]

Den systematiska observationsmetoden har sin grund i socialpsykologin och har använts för att exempelvis observera interaktioner i klassrum och liknande miljöer. Kvantitativ data och statistisk analys är två element som ofta förknippas med den systematiska observationsmetoden [26]. Den deltagande observationsmetoden kopplas oftast ihop med sociologi och antropologi. Metoden bygger på att forskaren deltar i de situationer som ska observeras och på så sätt försöker förstå de miljöer och processer som studeras [26].

Även om metoderna skiljer sig åt finns vissa gemensamma nämnare: direkt observation, fältarbete, naturliga miljöer samt frågan om perception. [26]

Data som insamlas vid observationer skiljer sig från data som insamlas via exempelvis intervjuer och frågeformulär. Data från sådana insamlingsmetoder baseras på vad informanterna berättar och beskriver för forskaren, till skillnad från observationer. Vid observationer samlar forskaren direkt informationen genom händelser. Genom att befinna sig i naturliga miljöer och inte i en konstruerad miljö, som exempelvis ett laboratorium eller intervjusituation, kan forskaren med säkerhet veta att det som observeras sker i dess normala tillstånd och är inte påverkad av forskarna själva.

Viktigt att notera är att perceptionen hos varje forskare kan störa resultatet vid datainsamlingen. Då varje forskare har olika personliga erfarenheter och influenser kan en observation av samma händelse registreras olika. För att försöka undvika problemen som kan uppstå med perception använder sig den systematiska metoden av *observationsscheman*. [26]

Ett observationsschema är ett verktyg som används för att minska risken att värdefull information glöms bort vid genomförandet av en observation. Genom att skapa ett schema med olika fokuseringskategorier kan forskare enklare registrera de händelser som uppstår under observationen och därmed undvika att glömma viktiga detaljer. För att skapa ett observationsschema behöver forskaren i förväg bestämma vilka punkter som är intressanta för just den aktuella studien. Dessa punkter väljes ut genom att följa vissa kriterier som försäkrar att det man registrerar är relevant för studien. [26]

Det som registreras i observationsschemat ska främst gälla öppna beteenden som går att mäta på ett direkt och enkelt sätt. Allt är inte möjligt att observera med det blotta ögat, så som tankar och synsätt. Händelserna som observeras ska vara konkreta och självklara, de ska inte behöva analyseras av forskaren för att veta vilken kategori handlingen tillhör. Händelserna ska även vara relevanta för studien, det finns ingen mening med att registrera händelser som inte kan kopplas till studien. Gemensam regel för alla händelser är att de ska förekomma regelbundet och tillräckligt ofta. Om händelsen man har valt att observera sker för sällan är det svårt samla in tillräcklig data för just den händelsen. För att observationsschemat ska anses pålitligt är det viktigt att det täcker alla möjligheter. För att alla händelser ska registreras på bästa sätt bör varje kategori vara väl definierad och tydliga. Om det förekommer otydligheter i kategorierna finns risken att exempelvis kategorierna överlappar och då kan fel data registreras.

En viktig komponent vid systematisk observation är att inte störa den miljön observationen sker på. När människor är medvetna om att de blir observerade finns det risk att de agerar annorlunda. Genom att exempelvis placera sig diskret och undvika interaktion med de människorna man ska observera förhindrar man att den naturliga miljön inte påverkas av forskarens närvaro. [26]

### **3.1.2 Webbaserad enkät**

Utöver observationer har vi även använt oss av en webbaserad enkät. Enkäten utformades från resultatet av ovannämnda observationsstudien för att kunna ge oss en djupare kunskap om vår urvalsgrupp, eleverna på CoderDojo.

En webbaserad enkät är uppbyggd som en webbsida där respondenter kan besöka för att besvara enkätens frågor. Fördelarna med detta är dels att det är enkelt att få till en stilren design på enkäten vilket gör det enklare för respondenterna att svara på frågorna. De svar som genereras från enkäten behöver inte sammanställas manuellt utan sker automatiskt via den webbaserade tjänsten. Detta sparar tid och forskaren får en snabbare överblick över resultatet. En nackdel som kan förekomma med att använda webbaserade enkäter är att forskaren ställer ett krav på respondenterna att de är tillräckligt tekniskt kunniga för att genomföra denna typ av enkät. Då vår målgrupp innehar denna kompetens påverkar denna nackdel inte vår studie. [26]

För att åstadkomma så tydligt resultat som möjligt ska alla frågor i formuläret följa vissa riktlinjer. Riktlinjer som följdes som vi följde beskrivs nedan:

Innan frågorna presenteras i enkäten fanns information till respondenterna om temat i frågorna och allmän relevant bakgrundsfakta till studien. Genom att göra detta blev respondenterna introducerade till ämnet och då vetande om vad för typer av frågor som fanns. Det var viktigt att frågorna i enkäten var anpassade för den målgrupp som

studien undersöker [26]. Vid skapandet av enkät till, exempelvis, barn måste formuleringar och ordval vara anpassade för att barnen enkelt ska förstå frågorna korrekt. För att frågorna ska generera så användbara svar som möjligt för studien var det viktigt att alla frågor är konkreta, icke-ledande, korta och enkla. Varje formulering på samtliga frågor skulle vara omöjlig att missförstå [26]. Att ha frågor som skapar förvirring eller gör respondenten osäker leder endast till dåliga svar och därmed ett dåligt slutresultat. Samma resultat blir det om frågorna blev för långa, då risken finns för att respondenten blir uttråkad och svarar på frågorna snabbt endast för att bli klar med enkäten så snabbt som möjligt [26]. Det var även viktigt att varje fråga är upplagd för att respondenten ska kunna svara från egna erfarenheter, och inte någon annans. Om en fråga exempelvis hade som syfte att ta reda på vad elever har för favoritämne finns det inget syfte att fråga elevernas föräldrar, eftersom det svaret de skulle ha gett endast är deras personliga uppfattning, och inte barnens [26].

För att få många svar var det viktigt att frågorna formulerades på ett sätt så att respondenterna var villiga att svara på dessa frågor. Frågorna fick inte göra respondenterna obekväma eller ovilliga att svara. För att undvika detta hade vi exempelvis inga frågor som gjorde antagande på respondenten, eftersom möjligheten finns att respondenten känner att denne inte lever upp till detta antagande. Frågorna hade även ha ett språk som inte skapade negativa reaktioner hos respondenterna [26].

## 3.2 Metoddiskussion

I vår studie har vi valt att använda oss av den systematiska versionen av observationer. Detta val gjorde vi då den passade oss bäst i förhållandet till vad vi ska skulle undersöka: barn och ungdomars inlärning av programmering i en naturlig miljö för lärande, utan några påverkningar från oss som forskare. Valet av webbenkäter gjordes eftersom vi då kunde nå ut till fler än bara CoderDojos verksamhet i Malmö, och genom detta få in mer data som kunde hjälpa oss att få ett mer omfattande resultat.

En kritik som kan riktas till vårt val av metod är att vi endast observerar och frågar en målgrupp som vistas på en plats. Dessa barn kan tänkas vara mer intresserad av teknik och programmering än genomsnittet, då de aktivt har ansökt om att få delta vid CoderDojos verksamheter. Vi är medvetna om att våran målgrupp inte är representabel för alla unga skolelever i Sverige men vi är ändå försiktigt optimistiska om att vi kan få ut data och resultat som kan bidra till mer kunskap inom detta ämnesområde.

## 3.3 Genomförande

### 3.3.1 Systematiska observationer

Våra systematiska observationer skedde hos CoderDojo. För att inte dra till oss uppmärksamhet introducerade vi oss endast till arrangörerna, inte barnen som vi skulle utföra observationen på. Detta gjorde vi för att undvika den så kallade observatörseffekten. Effekten innebär att människor som är medvetna om att de blir observerade agerar annorlunda än vad de gör vid ett normalt tillfälle. De kan exempelvis bli generade eller inte vara fullt fokuserade på deras uppgift.[26]

För att bevara den naturliga miljön valde vi även att vara på plats i god tid innan deltagarna hade kommit till platsen. Genom detta kan vi dels hitta en placering som både är diskret men ändå ger oss en god blick över observationsområdet, samt undvika interaktion med deltagarna genom att komma tidigt till platsen.

När observationssessionen startade valde vi att fokusera oss på en deltagare var. Om man väljer för många observationsobjekt finns det stora risker till att viktig information förbises. Vi observerade sedan deltagaren genom vårt observationsschema och registrerade varje gång något av förseelserna i schemat inträffade. Sammanlagt genomfördes åtta och en halv timme av observationer.

Observationsschemat vi använde visas nedan i figur 1. Här förklaras varför vi valde att fokusera på just dessa händelser.

Händelser	Förklaring/motivering
<b>Användaren ber någon av handledare om hjälp</b>	Detta registreras då det kan ge en bild av dels hur mycket assistans en användare behöver för programmering, men även hur handledningen av detta ämne kan gå till.
<b>Användaren ber någon av de andra användarna om hjälp</b>	Detta registreras för att undersöka om det finns en skillnad att få hjälp från en annan användare än från en handledare.
<b>Användaren hjälper en annan användare</b>	Kan användare hjälpa varandra och hur sker detta? Hur ofta? Är programmering något som man lär sig helt enskilt eller byter man kunskaper och tips med andra?
<b>Användaren håller på med aktiviteter som inte är kopplade till huvudaktiviteten (att programmera).</b>	Genom att registrera detta kan vi undersöka hur mycket tid som går till annat än huvudaktiviteten. Då programmeringen sker på en dator är det enkelt för användaren att snabbt byta aktivitet och därmed förlora koncentrationen på huvudaktiviteten.
<b>Användaren pratar med en annan användare om sitt projekt</b>	Att dela tankar och funderingar om sitt projekt är intressant att registrera eftersom detta kan ge en bild hur kunskap delas och hur frekvent detta görs.

Figur 1. Observationsschema

### 3.3.2 Webbaserad enkät

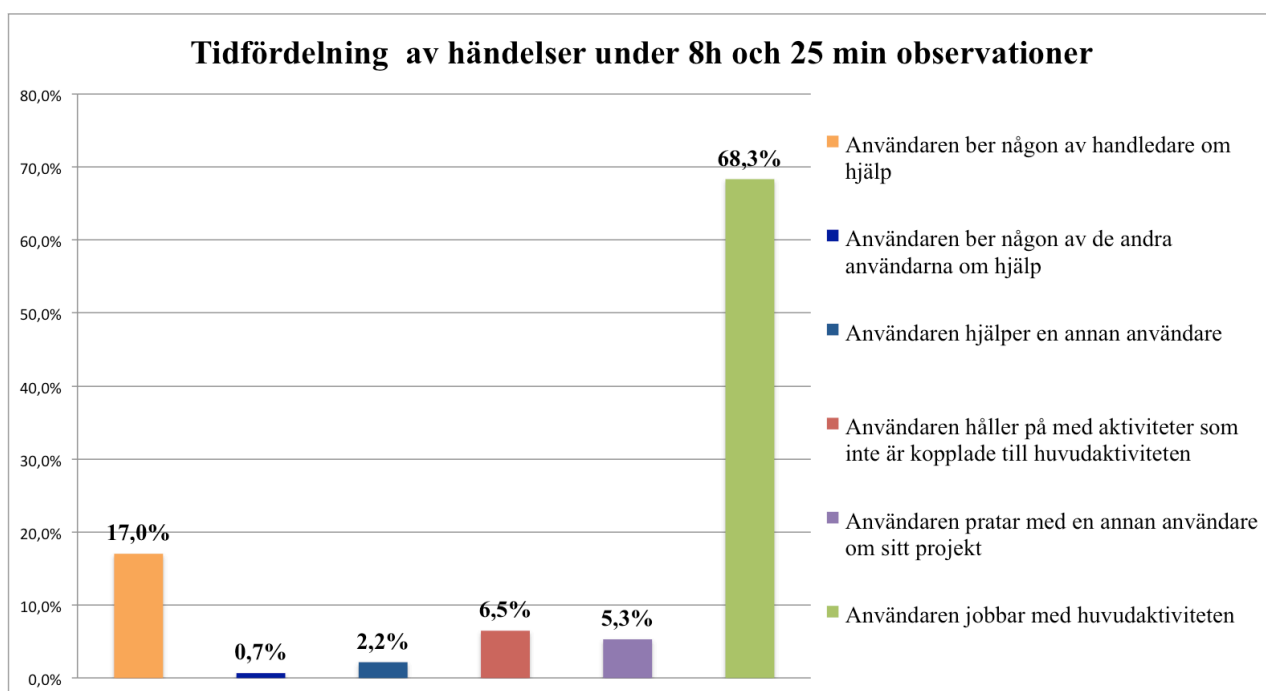
Enkäten skapades genom verktyget Google Forms. Detta val gjordes då man enkelt får samtliga svar på samma ställe och det var ett verktyg vi använt oss av innan. Frågorna i enkäten är dels kopplade till våra frågeställningar och de resultat vi fått fram genom vår observationsstudie, men även för att vi ska kunna få en generell bild av hur barn och ungdomar ser på programmering och programmering i skolan. Vi valde att skicka ut enkäten via mail till CoderDojos samtliga verksamheter i Sverige. Eftersom målgruppen är densamma hos alla CoderDojos verksamheter hade vi möjligheten att göra detta. Då vi hade genomfört vår observationsstudie hos CoderDojos verksamhet i Malmö valde vi att genomföra enkäten på plats hos dem.

## 4. Resultat

I detta avsnitt redovisar vi de resultat vi har samlat in från våra två olika forskningsmetoder.

### 4.1 Sammansatt resultat av alla observationer

Vi har valt att visualisera resultatet genom en sammanfattad tabell. Sammanlagt var det nio stycken observationspersoner som observerades genom vårt observationsschema under sammanlagt åtta timmar och 25 minuter observation.

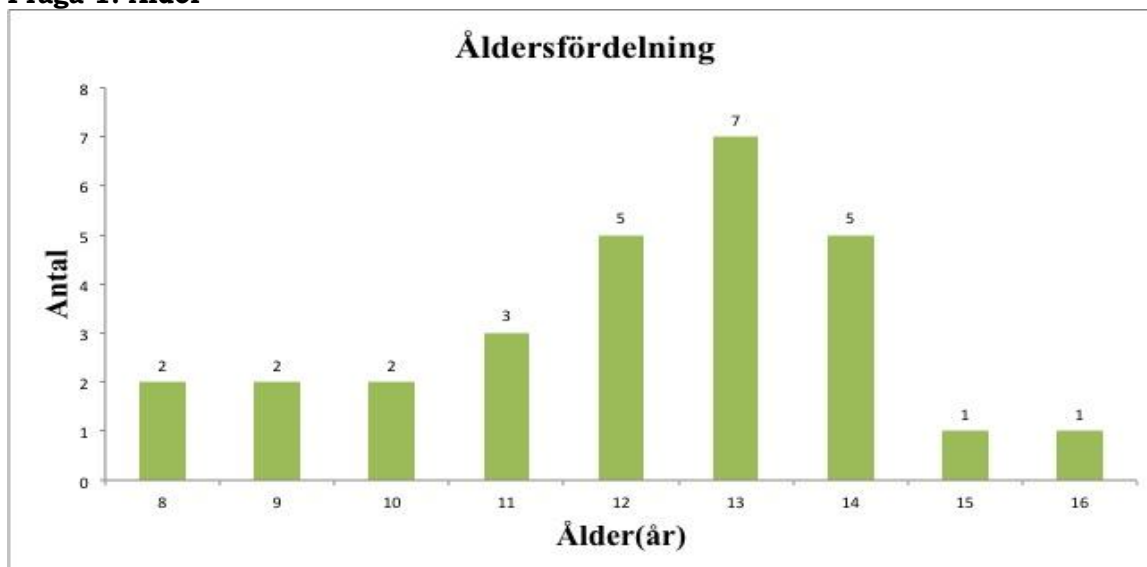


All individuell data, det vill säga data för varje person som observerades, finns att se på bilaga 3 i kapitel 9.

## 4.2 Sammansatt resultat från webbaserad enkät.

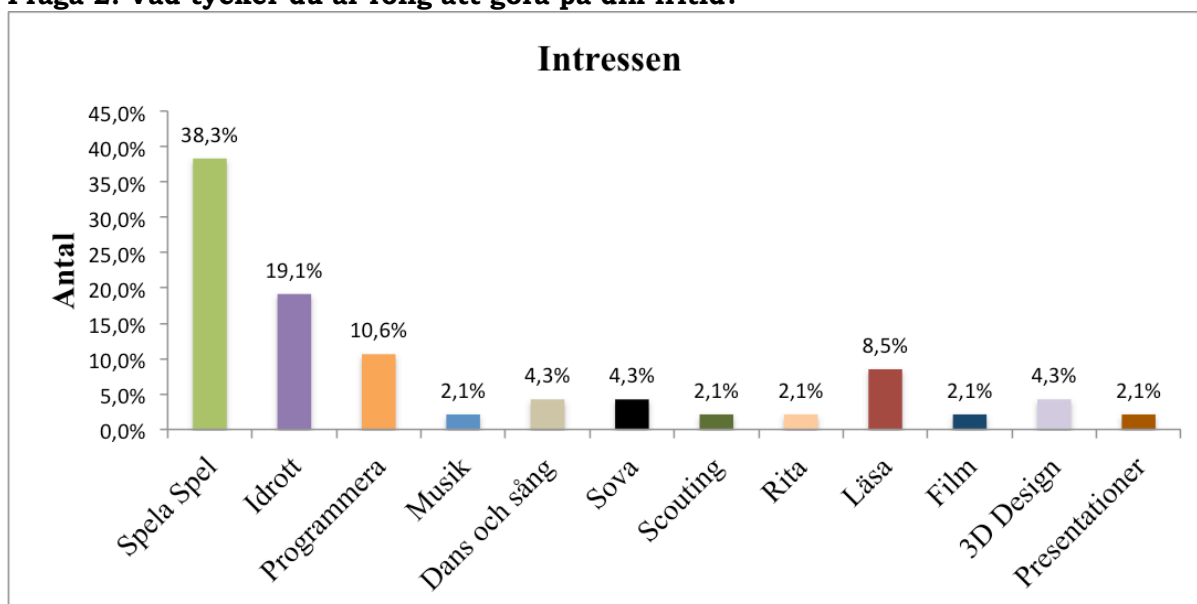
Dessa resultat är från de webbaserade enkäter vi har genomfört hos verksamheten CoderDojo. Vi har valt att visualisera resultatet genom tabeller och grafer. Nedan visas resultat från sammanlagt 28 stycken respondenter. Enkäten finns som bilaga 9.3.

### Fråga 1: Ålder



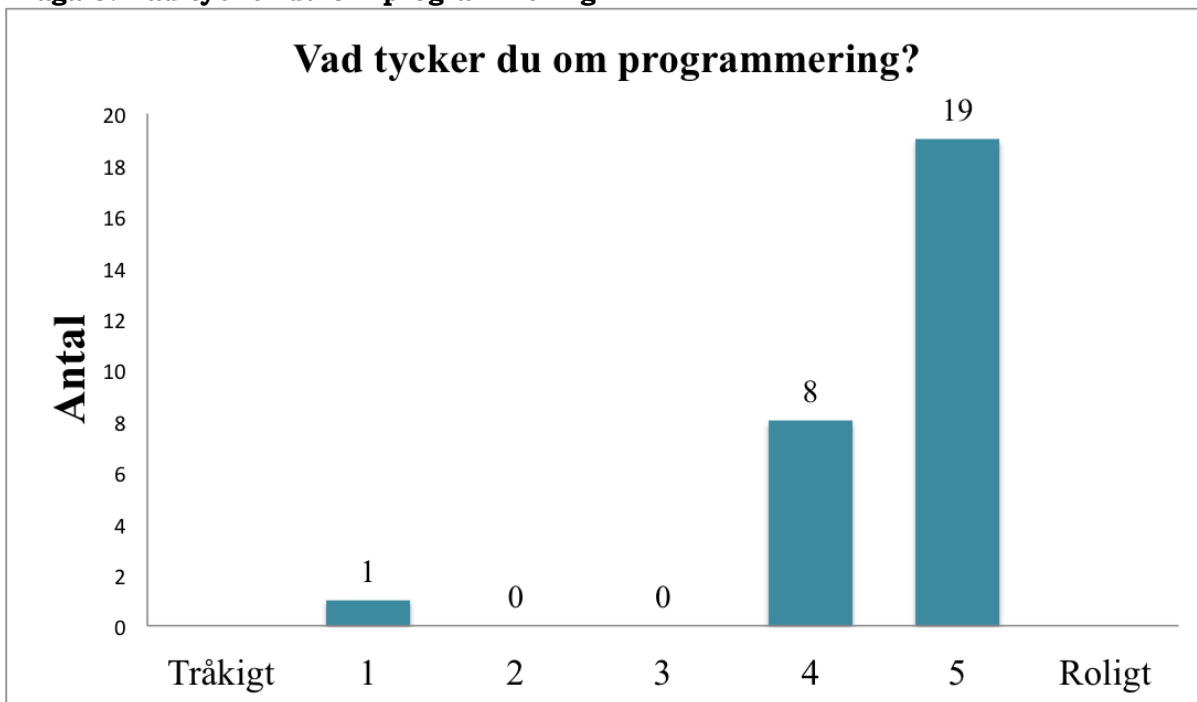
Figur 4.3.1, Åldersfördelning av elever på CoderDojo i år.

### Fråga 2: Vad tycker du är rolig att göra på din fritid?



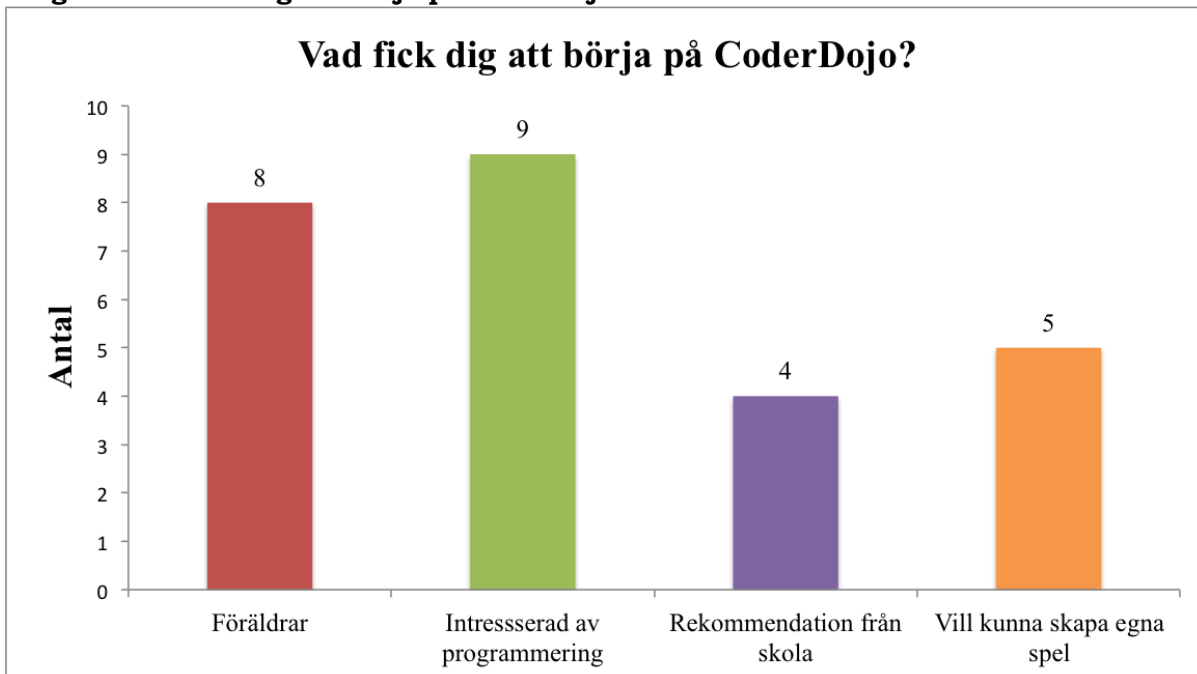
Figur 4.3.2, Elevers fritidsintresse från öppen fråga.

**Fråga 3: Vad tycker du om programmering?**



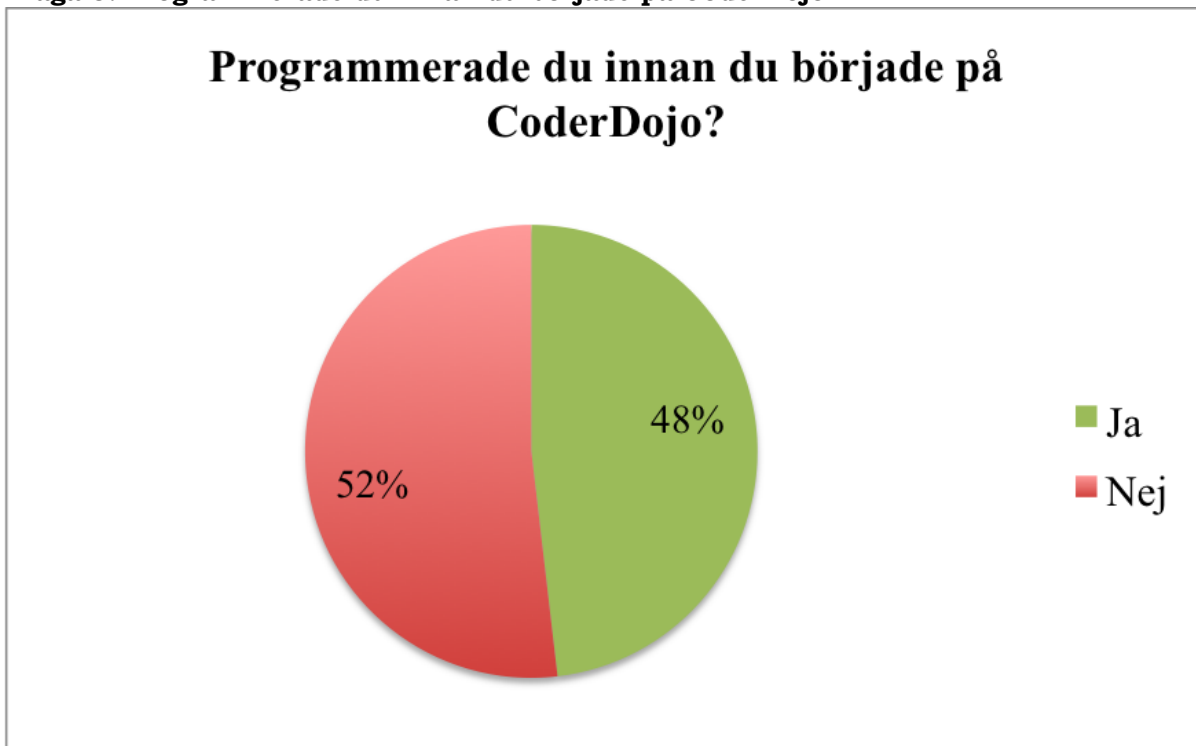
Figur 4.3.3, Elevers upplevelse av programmering på en tråkig till rolig skala.

**Fråga 4: Vad fick dig att börja på CoderDojo?**



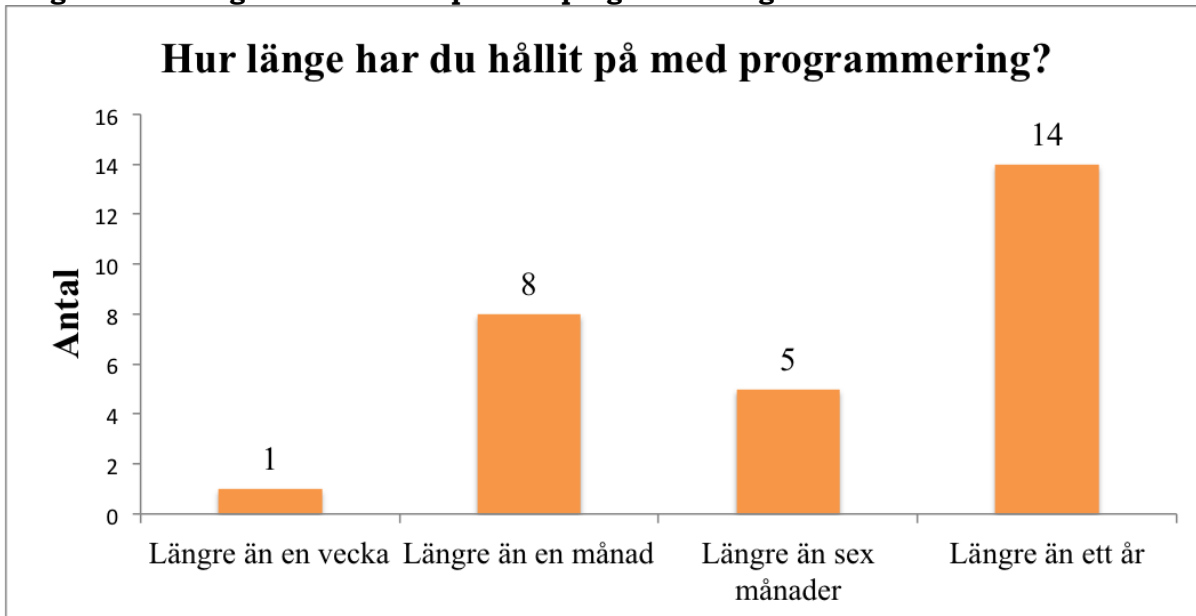
Figur. 4.3.4, Elevers motivering till att lära sig programmering på CoderDojo.

**Fråga 5: Programmerade du innan du började på CoderDojo?**

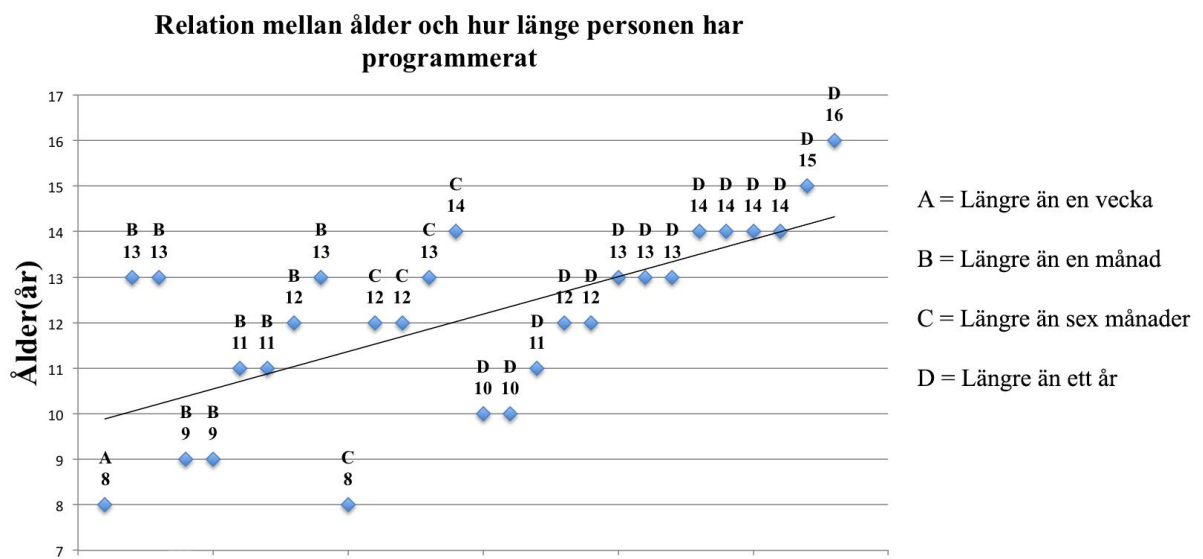


Figur 4.3.5, Andelen elever med erfarenhet av programmering innan CoderDojos utbildning.

**Fråga 6: Hur länge har du hållit på med programmering?**

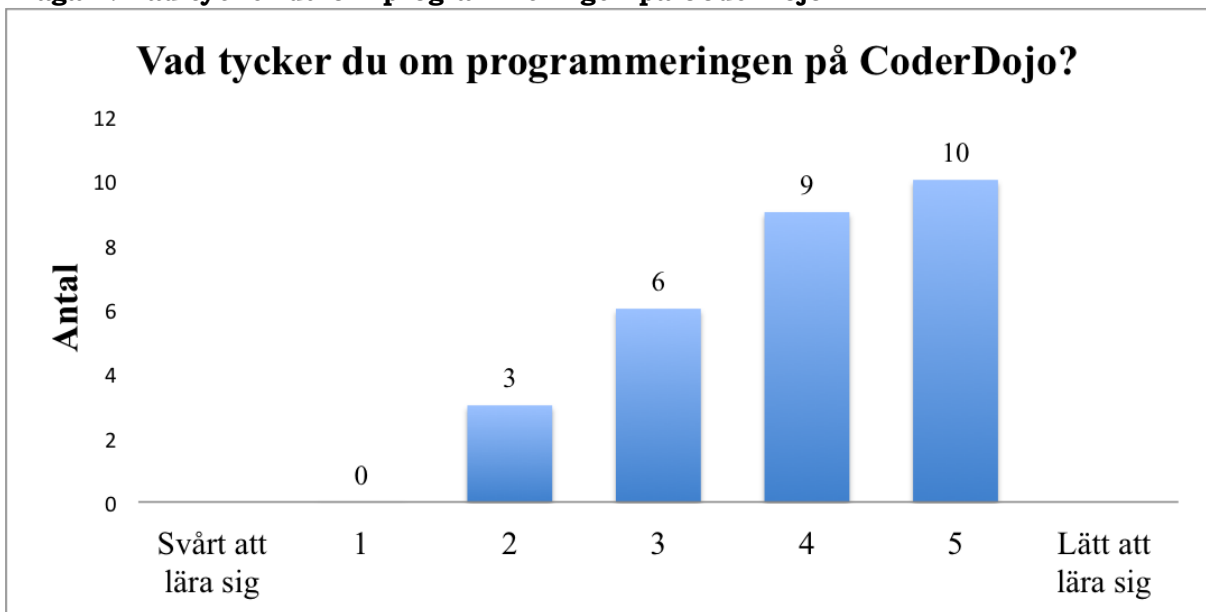


Figur 4.3.6, Elevers erfarenhet i programmering i förhållande till tid.



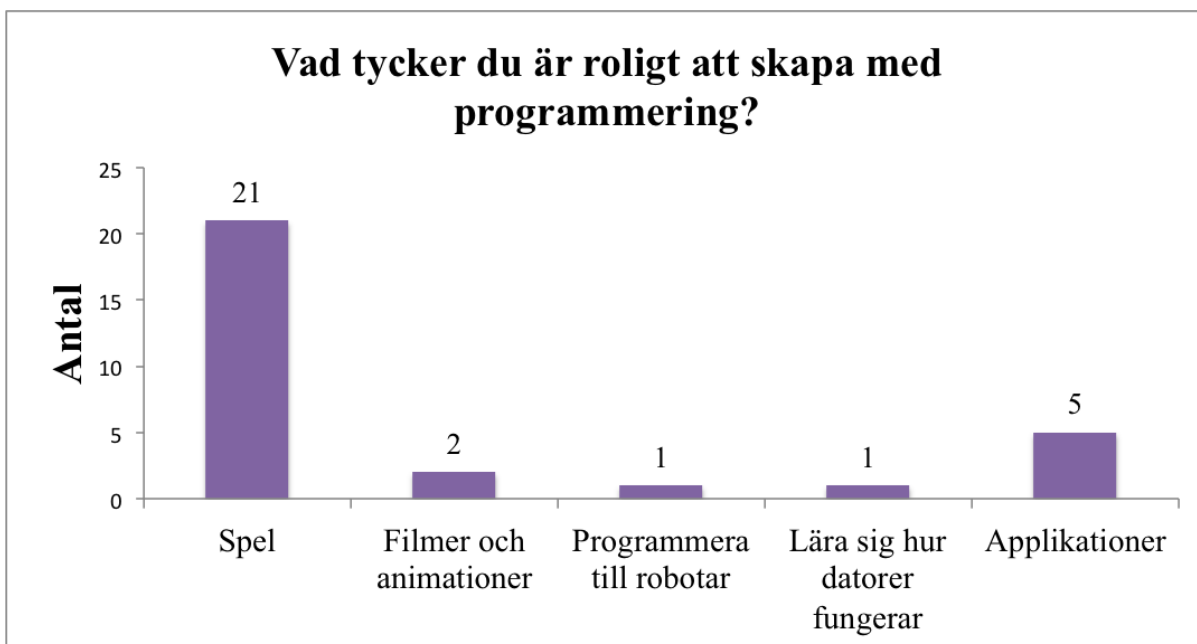
Figur 4.3.6.1, Programmerings erfarenhet i korrelation med ålder.

**Fråga 7: Vad tycker du om programmeringen på CoderDojo?**



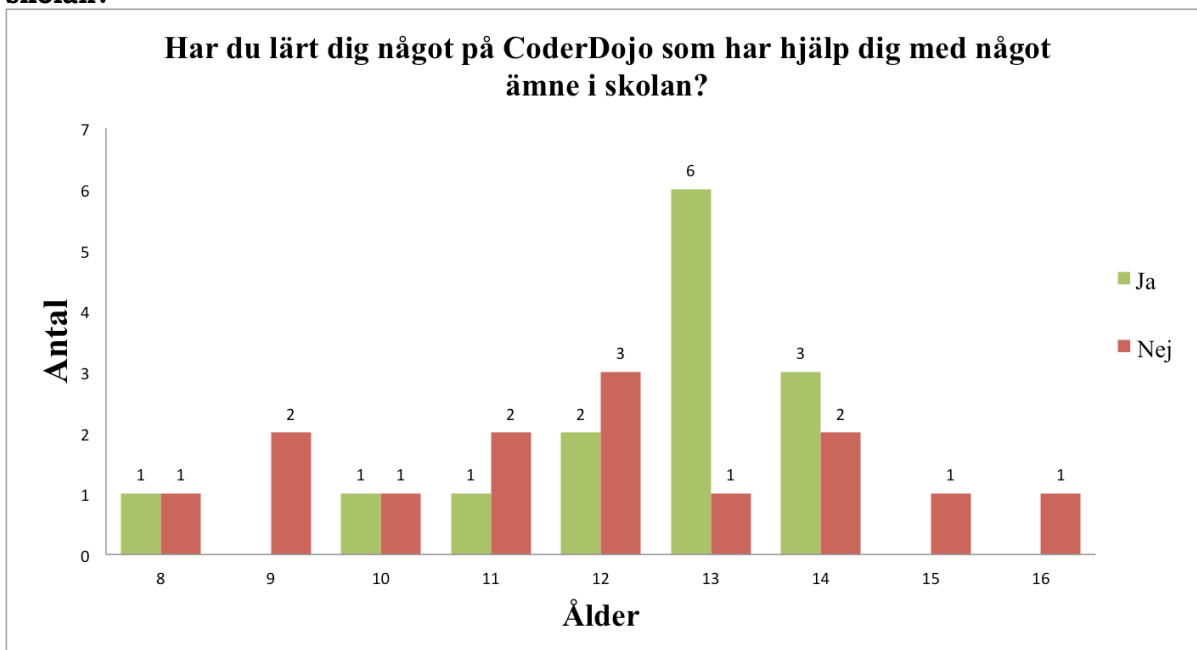
Figur 4.3.7, upplevelse av svårighet i inläring av programmering enligt CoderDojos lärometod

**Fråga 8: Vad tycker du är roligt att skapa med programmering?**



Figur 4.3.8, Elevers primära intressen (sammanfattat) i att skapa genom programmering.

**Fråga 9: Har du lärt dig något på CoderDojo som har hjälpt dig med något ämne i skolan?**

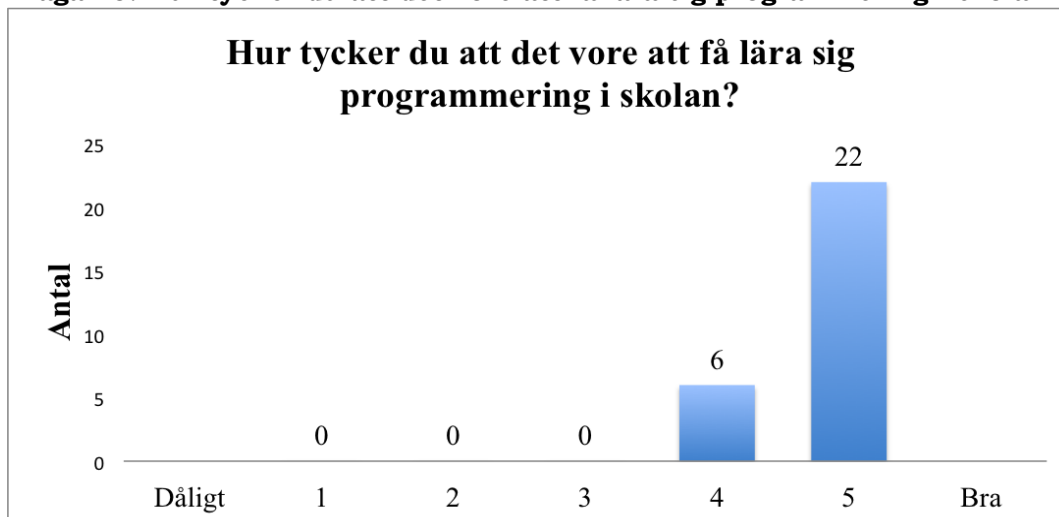


Figur. 4.3.9, Om elevers kunskapsinläring på CoderDojo har kunnat appliceras i skolan i relation till ålder.

Vissa respondenter angav exempel på vad de har lärt sig från CoderDojo som de har kunnat använda i skolan. Exempelen som dominerades bland svaren var matematik och logiskt tänkande. Några av eleverna beskrev även att programmeringsstudierna på CoderDojo har hjälpt dem så pass mycket med matematiken att de har höjt sina skolbetyg i ämnet. Nedan visas citat från en respondent:

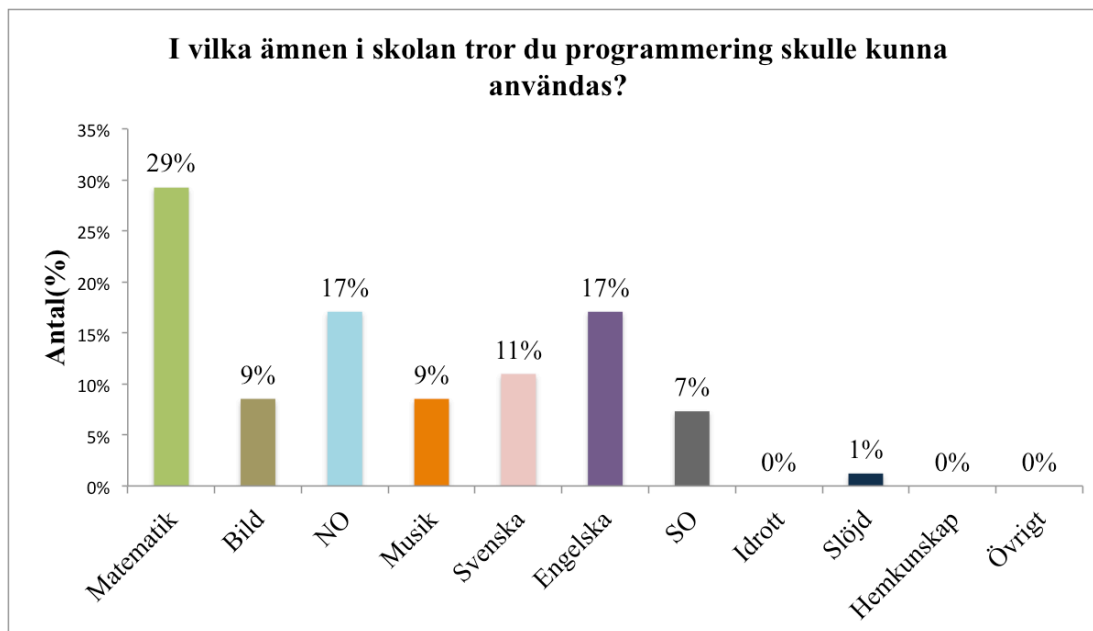
“Jag har haft väldigt mycket enklare med matte i skolan, jag har höjt mig från E eller D i matte till C eller B”

**Fråga 10: Hur tycker du att det vore att få lära sig programmering i skolan?**



Figur 4.3.10, Elevers intresse av att implementera programmering i deras skolutbildning.

**Fråga 11: I vilka ämnen i skolan tror du programmering skulle kunna användas**



Figur 4.3.11, Ämnesförslag från elever i vilket ämne som skulle kunna innehålla programmering.

I den sista fråga i enkäten frågades eleverna ifall de hade idéer och förslag till hur man kan använda programmering i de skolämnen som dem själva hade gett förslag på i ovanstående fråga. Då matematik var överlägset det populäraste ämnesvalet fick vi förstås många idéer på implementation inom detta ämne. Att använda programmering

till att utföra beräkningar med variabler och att beräkna grader var populära alternativ. De ansåg att detta kunde ge dem en djupare förståelse i beräkningar, formler och ekvationer.

Några elever tog sina förslag ett steg längre och ville applicera sina nyfunna matematiska kunskaper i skapande av spel och spelutveckling. Andra förslag till övriga ämnen var exempelvis att öva engelska glosor genom att programmera egna glossspel med endast engelska instruktioner. Ett förslag var att programmera olika typer av toner och skapa musik med tonerna inom ämnet musik. Även att skapa visuella bilder med hjälp av Scratch eller Processing till ämnet bild.

Sistnämnt var förslag på hur logiskt tänkande och problemlösningar som man utför i programmering, samt tekniska termer, kunde ingå i ämnet NO.

## 5. Analys

Analysavsnittet försöker vi främst besvara frågeställningen “*Är det nödvändigt att inkludera datalogiskt tänkande i skolundervisningen?*” men även “*På vilket sätt kan datalogiskt tänkande undervisas?*”

### 5.1 Analys av systematisk observation

Syftet med att utföra de systematiska observationerna var att undersöka hur datalogiskt tänkande kan läras ut med hjälp av programmering. Genom observationerna försökte vi dessutom studera hur användarna tog till sig den information som förmedlades och hur de använde verktygen (programmeringsspråken och miljöerna) under lektionerna. Den data vi samlat in under sammanlagt åtta timmar och 25 minuters observationer analyseras under detta avsnitt.

En och en halv timme, vilket motsvarar 17 % av den sammanlagda observationstiden, gick åt till att användarna sökte hjälp hos en handledare. Slutsatser som går att dra utifrån detta är dels användarnas förmåga att fråga om hjälp vid problem, men även hur viktigt det är med undervisare med kompetens inom området. Speciellt om man jämför denna data med den som innehar tid spenderat med att eleverna får hjälp av eller hjälper en annan elev. Det är 17 % (hjälp från handledare) och totalt 3 % (elever hjälper varandra). Om lärare eller mentorer saknar kompetens finns det risk att användarna inte får den hjälp som behövs.

Kategorierna “*användaren ber någon av de andra användarna om hjälp*” och “*användaren hjälper en annan användare*” ska logiskt sett vara lika stora. Anledningen varför första kategorin har 0,7 % och den andra har 2,2 % är ett resultat av att observationerna genomfördes på en person i taget. Vårt resultat kan då tolkas som att vi har observerat de användare som hjälper andra användare mer än vad de själva frågar om hjälp.

6,5 % av tiden gick åt till den fjärde kategorin, “*Användaren håller på med aktiviteter som inte är kopplade till huvudaktiviteten(programmera)*”. Detta resultat jämförde vi med en studie som har gjort av skolor i sydvästra Skåne vars syfte var att ta reda på hur mycket av undervisningstiden som går åt till annat än just undervisning. Studien genomfördes av 15 olika lärare på grund- och gymnasieskolor samt på Komvux under en vecka. Av sammanlagt 220 undervisningstimmar gick 36 av dessa till annat än undervisning, vilket motsvarar 16 %. [27]

Skillnaden i resultatet mellan de två studierna är att vår studie har 9,5 % mindre tid spenderat på annat än huvudaktiviteten. Denna skillnad kan vara ett resultat av att användarna i vårt fall är på en plats de själva har valt att befinna sig på, där de även utför en aktivitet de tycker om. Ytterligare en anledning till varför andelen tid lagd på annat än huvudaktiviteten i vår studies var lägre kan tänkas bero på den miljö som observationerna skedde. Under observationerna var det sju till tretton användare i lokalen och tre till fyra handledare. Detta resulterade i att användarna fick hjälp snabbare vid behov och då inte tillbringade tid med att göra aktiviteter utanför huvudområdet.

Vår analys av denna data är att den stora majoriteten av tid spenderat med aktiviteter som är eller direkt kopplade till huvudaktiviteten, programmering, pekar på att det finns ett riktigt intresse bland elever att programmera, skapa digitala ting och utföra uppgifter som kräver datalogiskt tänkande.

Vi kan även utsluta att det är en specifik utvecklingsmiljö eller programmeringsspråk som är orsak till intresseutvecklingen då eleverna arbetade i och med olika miljöer. Utifrån denna upptäckt antar vi att valet av teknik, som är anpassad till det eleverna vill skapa, är av betydelse för att eleverna lägger ner så pass mycket tid på sina uppgifter. CoderDojo har helt enkelt använt rätt verktyg på rätt plats, vilket bidrar till arbetsglädjen hos barnen.

Det första vi som observatörer slogs av under studien var nivån av koncentration som eleverna hade under programmeringslektionerna. Innan vi påbörjade observationerna hade vi en förväntning om att barn mellan 7 och 17 år som är samlade i mindre rum skulle vara högljudda, springa runt och inte fokusera sig helt åt huvudaktiviteten. Denna förväntning grundades dels av att en av observatörerna har erfarenhet som lärarvikarie för liknande åldrar. Detta visar sig i resultatet då i genomsnitt tillbringade eleverna 93,5 % av tiden med huvudaktiviteten (programmering), endast 6,5 % av tiden gick åt till aktiviteter som inte var kopplade med huvudaktiviteten. Av de 93,5 % var 68,3 % ägnat åt självständigt arbete, utan att få hjälp eller hjälpa någon annan med programmering

## 5.2 Analys av webbaserad enkät

Syftet med de webbaserade enkäterna var att försöka förstå vad användarna själva tycker om programmering och tanken på att få lära sig det i skolan.

Ett genomgående tema i resultatet till dessa frågor är fokuset på spel:

- Fråga 2: Vad tycker du är roligt att göra på din fritid?
- Fråga 4: Vad fick dig att börja på CoderDojo?
- Fråga 8: Vad tycker du är roligt att skapa med programmering?

Av 28 respondenter angav 38,3 % av dem spel som fritidsintresse och detta speglar även av sig på resultaten i frågorna fyra och åtta. En av de främsta anledningarna för att börja på CoderDojo var för vissa användare att få möjligheten att lära sig att skapa egna spel. Detta kan man även se i resultat till fråga åtta där 70 % av respondenterna har angett att spel är något de tycker roligt att skapa med programmering. Dessa resultat kan tolkas att det är viktigt att tillämpa programmering till en kontext och ämne som intresserar eleverna själva och det motiverar deras inlärning och arbete.

Frågor som var kopplade till programmering i skolan var:

- Fråga 9: Har du lärt dig något på CoderDojo som har hjälpt dig med något ämne i skolan?
- Fråga 10: Hur tycker du att det vore att få lära sig programmering i skolan?
- Fråga 11: I vilka ämnen i skolan tror du programmering skulle kunna användas?

I fråga 9("Har du lärt dig något på CoderDojo som har hjälpt dig med något ämne i skolan?") berättar respondenter hur aktiviteterna på CoderDojo har hjälpt dem med matematik och logiskt tänkande. Detta syns även i fråga 11("vilka ämnen i skolan tror du programmering skulle kunna användas") där 29 % angav matematik som det ämne de tror programmering skulle fungera som ett bra verktyg. Detta är data som pekar på att programmering som verktyg och datalogiska koncept har potential att införas och hjälpa inom existerande skolämne, vilket i vår analys är främst matematik. Vilket har varit en idé och förhoppning i debatten om införande av programmering i skolan.

Från resultatet i fråga 9 kan man även utläsa att aktiviteterna på CoderDojo har haft störst inverkan på respondenter i åldrarna 13 och 14 år. Detta resultat kan kopplas ihop med resultaten i fråga 6, där vi frågar hur länge respondenten har hållit på med programmering. Större delen av de respondenter i åldrarna 13 och 14 år har hållit på med programmering längre än ett år. Relationen mellan svaren i fråga 6 ("Hur länge har du hållit på med programmering?") och fråga 9("Har du lärt dig något på CoderDojo som har hjälpt dig med något ämne i skolan?") kan tolkas som att de personer som har programmerat under en längre tid har större chans till att använda denna kunskap i skolundervisningen och att datalogisk tänkande kan tänkas ha störst inverkan på elever i sjätte och sjunde årsklass.

Denna tolkning är dock inte självklar i alla åldrar. Av de respondenter som var äldre än 14 år hade samtliga programmerat i längre än ett år, men ingen hade tagit nytta av de aktiviteter som genomfördes på CoderDojo i skolundervisningen. Detta kan kopplas ihop med resultaten från fråga 7, där det framgår att svårighetsnivån på de programmeringsspråk som lärs ut på CoderDojo inte är hög. Detta kan då leda till att äldre deltagare inte kan utnyttja de kunskaper de lär sig på CoderDojo i samma utsträckning som yngre deltagare.

Sammanfattningsvis hade respondenterna en övervägande positiv inställning till både programmering och tanken på att få lära sig mer om det i skolan.

## 6. Diskussion och Slutsats

Vår slutsats är att om elever arbetade med personligt meningsfulla projekt, upptäckte vi att de var redo och ivriga att lära sig viktiga matematiska och datalogiska koncept som kunde appliceras på deras projekt.

Vi uppmärksammade att det finns få hinder för barn så långt ner som i sju års ålder att förstå programmering och datalogiskt tänkande. Visuella programmeringsspråk, som var populärt bland de yngre eleverna, verkar hjälpa dem att enkelt ta till sig koncept och tekniker inom programmering. Vi upplever snarare att barnen tycker att det är både roligt och spännande att få en dator och digitalt visuella artefakter att göra som de ville.

Om man vill att barn ska få en fast datalogisk kunskap genom programmering uppmuntra dem i så fall att bygga något som upplevs som häftigt eller hjälper med existerande uppgifter och moment i skolan. Det ska inte genom att skriva in exakta korrekta kommandon i en programmeringsmiljö. Genom att lära bara tillräckligt om hur den digitala och lilla kantiga värld inuti datorn arbetar för att upptäcka vad de och deras vänner kan göra tillsammans.

Vi menar att grundtanken med att ha programmering i skolan inte borde vara att lära ut specifika programmeringsspråk, något som snabbt riskerar att bli omodernt eller poänglöst för elever. Till exempel att hålla lektioner i Java-programmering och bara lära ut hur man programmerar. Så man borde inte lära barnen ren programmering utan istället ge dem olika strukturerade möjligheter att leka med hårdvara, mjukvara och datalogiska tekniker för att utforska och fördjupa sig i redan existerande skolämnen. Eleverna ska bemöta datorn, programmering och datalogi som ett verktyg.

Sammanfattningsvis har vår studie kommit fram till att syftet med att implementera inläring av datalogi och programmering som verktyg skulle ge elever egenskaper som att bryta ner problem i mindre delar, hitta och utnyttja mönster, skapa lösningar med hjälp av matematik och logik samt att kunna modellera information. Vår studie visar att dessa ovanstående egenskaper kan hjälpa elever främst inom ämnen som matematik och NO. Samt som resultatet visade kan även programmering tänkas ge störst inverkan på dessa studieämnen mellan årskurserna sex till åtta.

Som ytterligare en fördel ger det elever ytterligare digital kompetens och kan ge dem en insikt i hur modern teknik fungerar och hur man skapar något på en dator. Vilket kan vara en framtidsinvestering för elever som upptäcker ett intresse för datavetenskap.

Men det är inte bara att implementera programmering i skolundervisningen för att få ut alla dessa vinster i kompetens. Ett tydligt mönster vi kunde tyda från vårt resultat är vikten av kompetens från de personer som lär ut datalogisk tänkande. Att som elev ha möjligheten att kunna fråga, diskutera och samtala om ett så pass nytt ämne är viktigt för elevernas utveckling. Om lärarna inte innehar den kompetens som eleverna behöver är det en risk för att viljan att lära sig mer försvinner eller minskar.

Avslutningsvis vill vi problematisera resultatet från vår studie till alla läsare som är engagerad i svensk skolutveckling inom digital kompetens och datalogi. Då vår urvalsgrupp för studien var elever som kan ha varit influerade hemifrån eller haft ett eget tidigare intresse för teknik, programmering och digitalt skapande så finns en verklig möjlighet att det resultat vi funnit från vår urvalsgrupp inte är representativ för hela populationen, vilken alltså är samtliga grundskoleelever i Sverige.

Genom vår forskning har vi kommit fram till att programmering i skolan behöver införas genom noggranna strategier. Att val av skolämne, årskurs och programmeringsverktyg/miljö är kritisk för att programmering ska bli ett hjälpmedel och inte en belastning för utbildningen. Att ge elever de bästa möjligheterna att upptäcka och lära sig datalogiskt tänkande samt att lära dem applicera detta tänkande i sina utbildningar.

För att få fram konkret kunskap om hur dessa strategier bäst skapas och utförs kommer det behövas vidare forskning med fler testpersoner, tester i riktiga skolmiljöer och forskning som utförs över en längre tid. Detta för att få ett säkrare resultat än det som vi fick.

Vi hoppas att efter ni har läst denna uppsats funnit kunskap i de faktiska vinster, svårigheter och möjligheter som finns med att applicera programmering som verktyg inom utbildning för barn och ungdomar. Genom att ge unga elever och barn chansen att utveckla ett datalogiskt tänkande samt ge dem nya verktyg till att utveckla de centrala ämnesområden som idag utmärker svensk grundskoleutbildning möjliggör man större utveckling och digital kompetens av det svenska skolsystemet för eleverna.

## 8. Referenslista

- [1]G. Fridolin, A. Hadzialic and M. Kaplan, "Programmering in på schemat i ny skolstrategi", *Regeringskansliet*, 2015. [Online]. Tillgänglig: <http://www.regeringen.se/debattartiklar/2015/09/programmering-in-pa-schemat-i-ny-skolstrategi/>. [Hämtad: 11- Apr- 2016].
- [2]K. Nygård and T. Raymond, "Hackad läroplan | Teacherhack", *Teacherhack.com*, 2016. [Online]. Tillgänglig: <http://www.teacherhack.com/hackad-laroplan/>. [Hämtad: 11- Apr- 2016].
- [3]"Teacherhack vill hacka läroplanen och få in programmering i skolan", *Webbstjarnan.se*, 2013. [Online]. Tillgänglig: <https://www.webbstjarnan.se/blogg/teacherhack-vill-hacka-laroplanen-och-fa-in-programmering-i-skolan/>. [Hämtad: 11- Apr- 2016].
- [4]A. Balanskat and K. Engelhardt, "Computing our future - Computer programming and coding Priorities, school curricula and initiatives across Europe", European Schoolnet, Brussels, 2015.
- [5]S. Pålsson, "Programmering i europeiska läroplaner", *Omvärldsbloggen Skolans digitalisering*, 2015.
- [6]F. von Essen, "AKUT OCH STRUKTURELL I IT- OCH TELEKOMSEKTORN KOMPETENSBRIST", IT&Telekomföretagen, Stockholm, 2015.
- [7]"CoderDojo Malmö", *CoderDojo Malmö*, 2016. [Online]. Tillgänglig: <http://malmo.coderdojo.se/>. [Hämtad: 11- Apr- 2016].
- [8]M. Prensky, "Digital Natives, Digital Immigrants Part 1", *On the Horizon*, vol. 9, no. 5, pp. 1-6, 2001.
- [9]J. Wing, "Computational thinking", *Communications of the ACM*, vol. 49, no. 3, p. 33, 2006.
- [10]"Programmering vs. datalogiskt tänkande", *EdTech Sweden*, 2015. [Online]. Tillgänglig: <https://edtechswe.com/2015/04/10/programmering-vs-datalogiskt-tankande/>. [Hämtad: 04- Maj- 2016].
- [11]S. Papert, *Mindstorms*. New York: Basic Books, 1980.
- [12]C. Kelleher and R. Pausch, "Lowering the barriers to programming", *CSUR*, vol. 37, no. 2, pp. 83-137, 2005.
- [13]"BASIC at Dartmouth", *Dartmouth.edu*, 2014. [Online]. Tillgänglig: <http://www.dartmouth.edu/basicfifty/basic.html>. [Hämtad: 16- Apr- 2016].
- [14]S. Wang, *Technology integration and foundations for effective leadership*. Hershey, Pa.: IGI Global (701 E. Chocolate Avenue, Hershey, Pennsylvania, 17033, USA), 2013, p. 24.
- [15]"logothings - home", *Logothings.wikispaces.com*, 2016. [Online]. Tillgänglig: <https://logothings.wikispaces.com/>. [Hämtad: 13- Apr- 2016].

- [16]M. Guzdial, "Teaching Programming with Music: An Approach to Teaching Young Students About Logo", *Logo Foundation Online Papers*, 2006.
- [17]T. McNerney, *Tangible programming bricks*. 1999.
- [18]J. Motil and D. Epstein, "JJ: A Language Designed For Beginners", 1998.
- [19]"Overview \ Processing.org", Processing.org, 2016. [Online]. Available: <https://www.processing.org/overview/>. [Hämtad: 03- Oct- 2016].
- [20]Glinert and Tanimoto, "Pict: An Interactive Graphical Programming Environment", *Computer*, vol. 17, no. 11, pp. 7-25, 1984.
- [21]J. Maloney, M. Resnick, N. Rusk, B. Silverman and E. Eastmond, "The Scratch Programming Language and Environment", *ACM Transactions on Computing Education*, vol. 10, no. 4, pp. 1-15, 2010.
- [22]K. Kahn, M. Cruz and L. Morgado, "Radia Perlman A pioneer of young children computer programming", *Academia.edu*, 2016. [Online]. Tillgänglig: [http://www.academia.edu/2795453/Radia\\_Perlman\\_A\\_pioneer\\_of\\_young\\_children\\_computer\\_programming](http://www.academia.edu/2795453/Radia_Perlman_A_pioneer_of_young_children_computer_programming). [Hämtad: 21- Apr- 2016].
- [23]"COMAL", Wikipedia, 2016. [Online]. Tillgänglig: <https://en.wikipedia.org/wiki/COMAL>. [Hämtad: 21- Apr- 2016].
- [24]A. Engström, "Compis - datorn som blev ovän med alla", *IDG.se*, 2011. [Online]. Tillgänglig: <http://www.idg.se/2.10186/1.142677/compis---datorn-som-blev-ovan-med-alla>. [Hämtad: 11- Apr- 2016].
- [25]M. Resnick, B. Silverman, Y. Kafai, J. Maloney, A. Monroy-Hernández, N. Rusk, E. Eastmond, K. Brennan, A. Millner, E. Rosenbaum and J. Silver, "Scratch", *Communications of the ACM*, vol. 52, no. 11, p. 60, 2009.
- [26]M. Denscombe, *Forskningshandboken*. Lund: Studentlitteratur, 2009.
- [27]"Datorkrängel och tjafs stjal lärarnas tid", *Sydsvenskan*, 2014. [Online]. Tillgänglig: <http://www.sydsvenskan.se/2014-11-12/datorkrangel-och-tjafs-stjal-lararnas-tid>. [Hämtad: 02- Jun- 2016].

## 9. Bilagor

### 9.1 Svar utbildningsdepartementet från den öppna enkäten



REGERINGSKANSLIET

2016-03-31

U2016/01356/S

#### Utbildningsdepartementet

Gustav Svensson  
gustavsvenssonn@gmail.com

*Skolenheten  
Departementssekreterare  
Filip Nilsson  
Telefon 08-405 41 78  
Mobil 072-517 13 83  
E-post filip.nilsson@regeringskansliet.se*

Tack för ditt brev om programmering i grundskolan som du ställt till utbildningsminister Gustav Fridolin och gymnasie- och kunskapslyftsminister Aida Hadzialic. Jag har blivit ombedd att besvara brevet.

För att stärka förutsättningarna för en genomtänkt användning av it i skolan gav regeringen den 24 september 2015 Statens skolverk i uppdrag att föreslå nationella it-strategier för skolväsendet (U2015/04666/S). Strategierna ska bidra till ökad måluppfyllelse och likvärdighet genom att den strategiska potential som it har tillvaratas i hela skolväsendet. Förslagen ska för grundskolan och motsvarande skolformer även innefatta förändringar i läroplaner och kursplaner för att förstärka och tydliggöra programmering som ett inslag i undervisningen.

Skolverket kommer att redovisa de delar som rör förskolan, förskoleklassen, fritidshemmet, grundskolan och motsvarande skolformer nu i dagarna men uppdraget har ändrats på så sätt att redovisning av den del av uppdraget som avser att lämna förslag på förändrade lydelse i läroplaner, kursplaner eller ämnesplaner flyttas fram till den 30 juni 2016. De förslag på förändrade lydelse Skolverket då ska redovisa kommer sedan att beredas inom Regeringskansliet. Uppdraget i sin helhet kan du ta del av här: [www.regeringen.se/regeringsuppdrag/2015/09/uppdrag-att-foresla-nationella-it-strategier-for-skolvaseudet/](http://www.regeringen.se/regeringsuppdrag/2015/09/uppdrag-att-foresla-nationella-it-strategier-for-skolvaseudet/)

Ändringar i läroplan och kursplaner behöver som regel föregås av någon form av implementeringsinsatser riktade till lärare, rektorer, huvudmän etc. För att få en bild av hur Skolverket som förvaltningsmyndighet för skolan arbetar kan du ta del av den plan som myndigheten redovisade för implementeringen av nuvarande läro- och kursplaner. Du hittar den här: [www.skolverket.se/publikationer?id=2321](http://www.skolverket.se/publikationer?id=2321)

Med vänlig hälsning

Filip Nilsson

Postadress  
103 33 Stockholm  
Besöksadress  
Drottninggatan 16

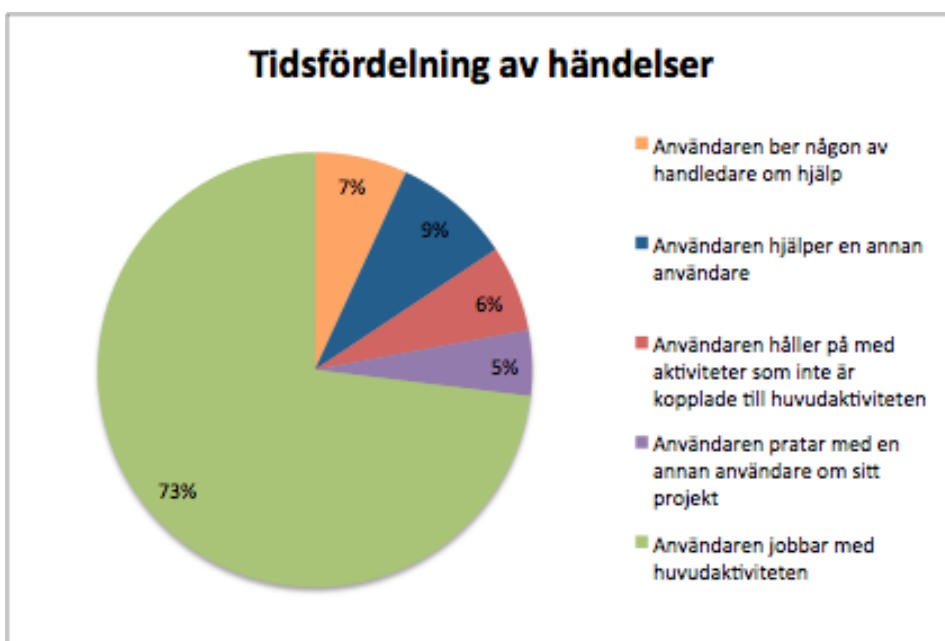
Telefonväxel  
08-405 10 00  
Telefax  
08-21 68 13

E-post: u.registrator@regeringskansliet.se

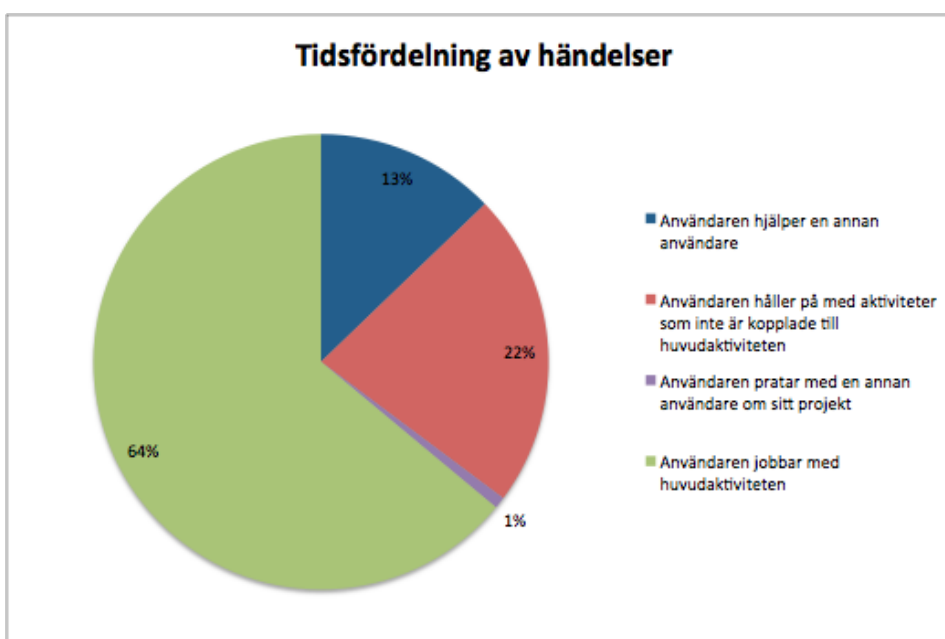
## 9.2 Observationsschema

	Tidpunkt	Varighet	Kommentar
Användaren ber någon av <b>handledare</b> om hjälp			
Användaren ber någon av de andra <b>användarna</b> om hjälp			
Användaren hjälper en annan användare			
Användaren håller på med aktiviteter som inte är kopplade till huvudaktiviteten			
Användaren pratar med en annan användare om sina projekt			

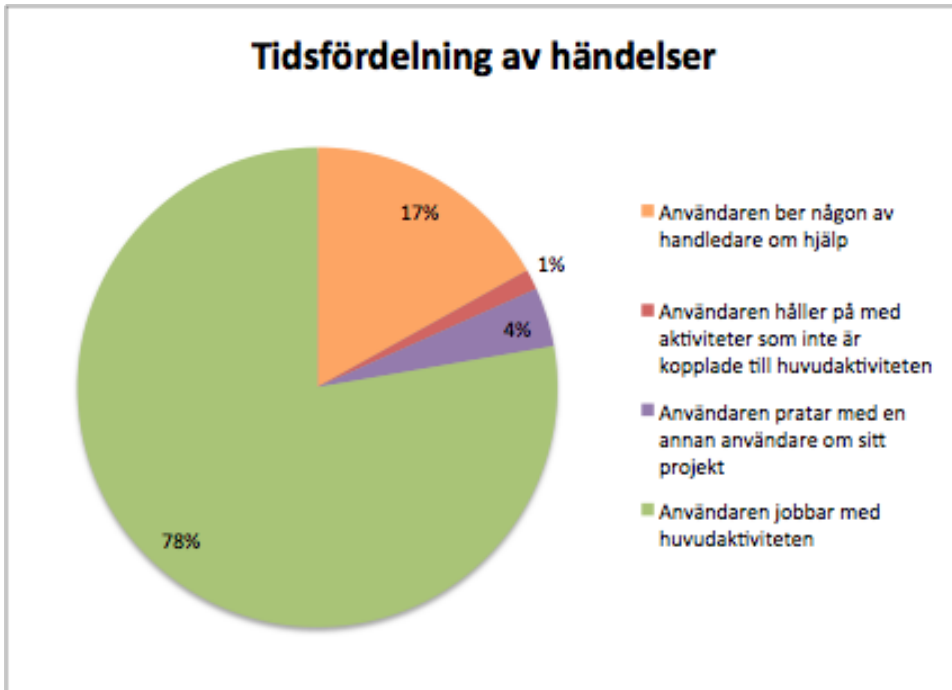
## 9.3 Observationsdata



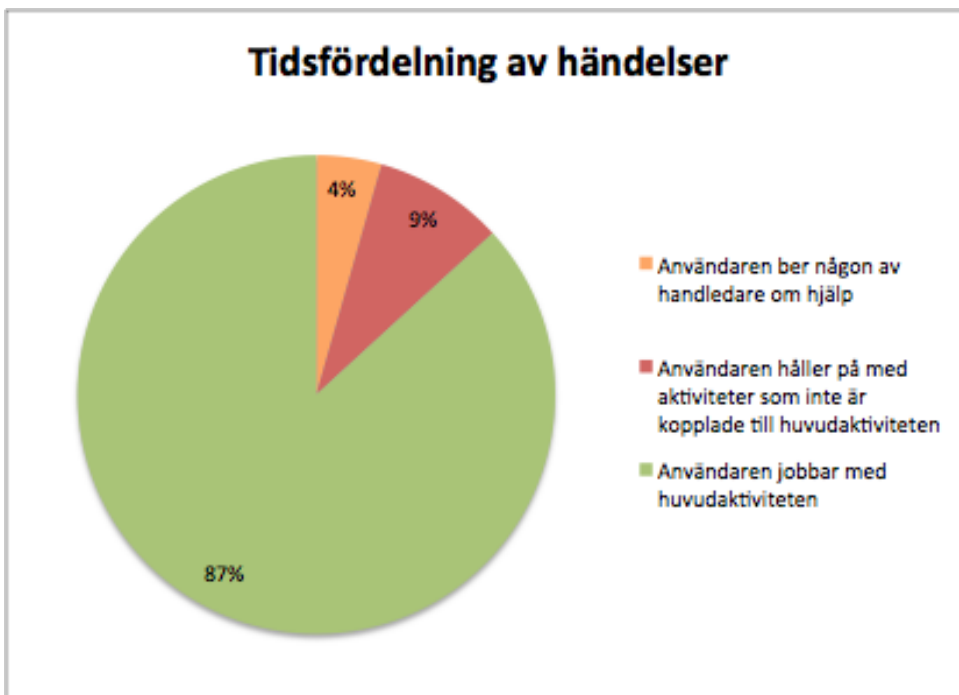
Figur 9.3.1 Observationsperson A, 60 minuter



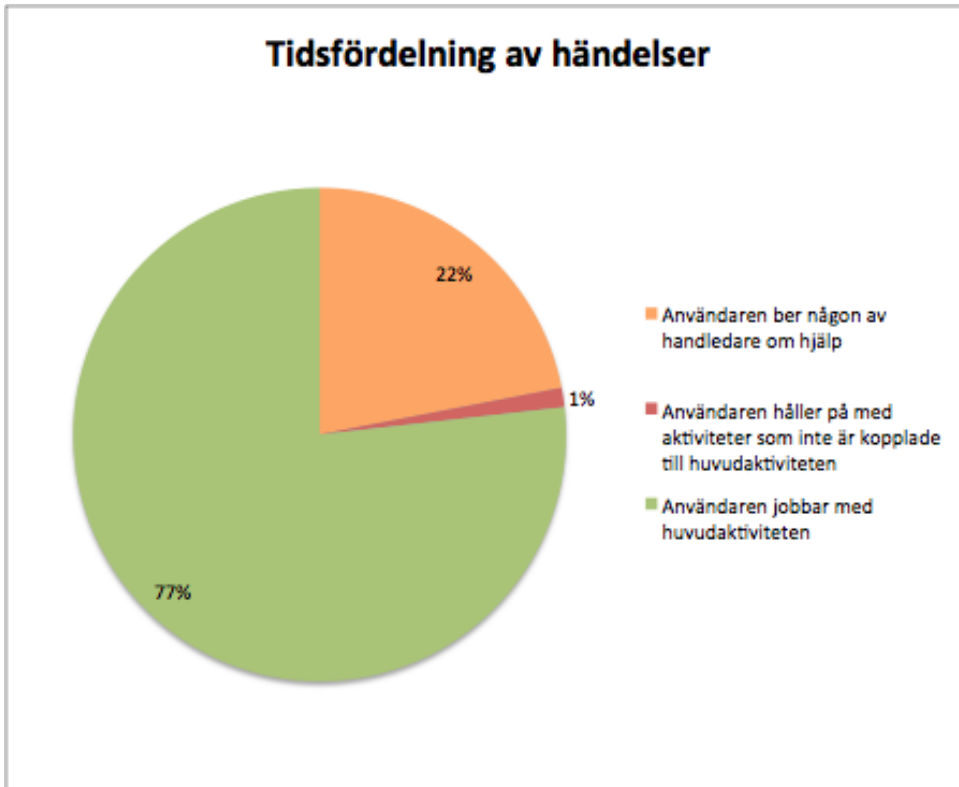
Figur 9.3.2, Observationsperson B, 45 minuter



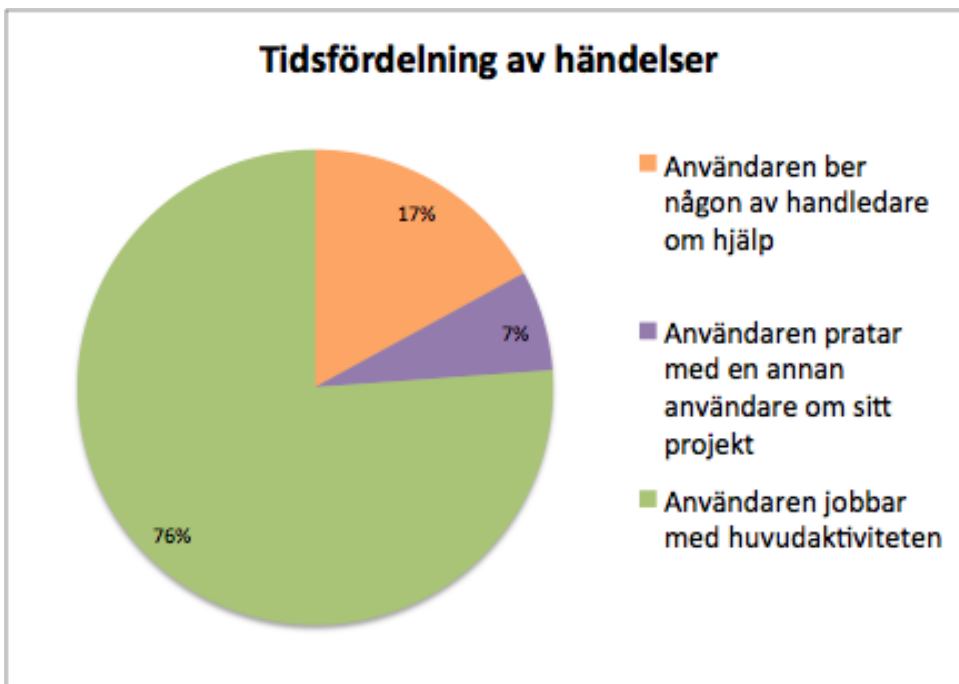
Figur 9.3.3 Observationsperson C, 60 minuter



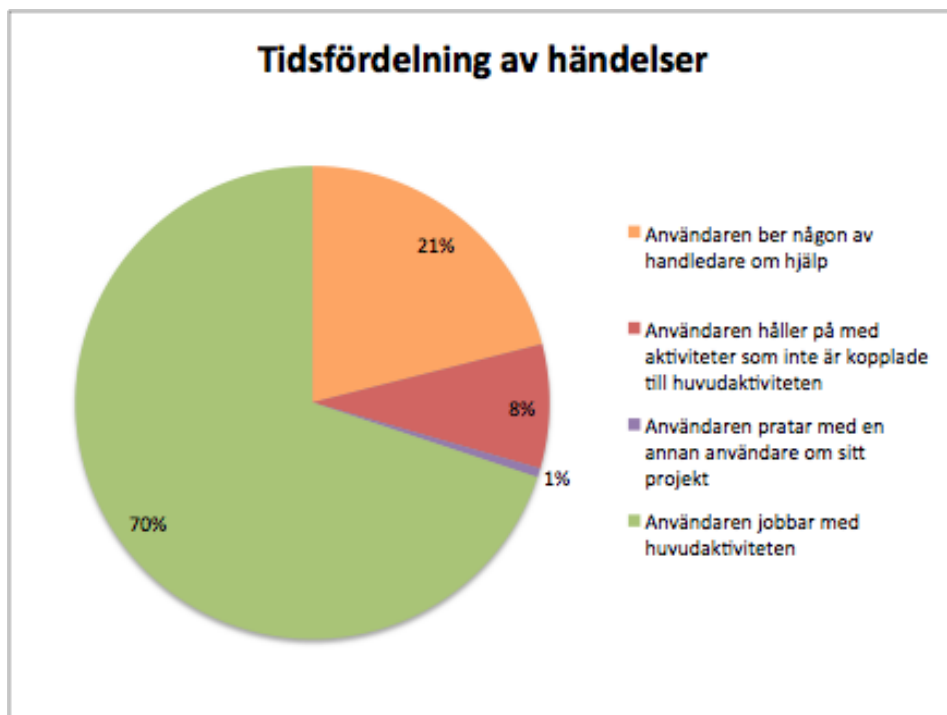
Figur 9.3.4, Observationsperson D, 60 minuter



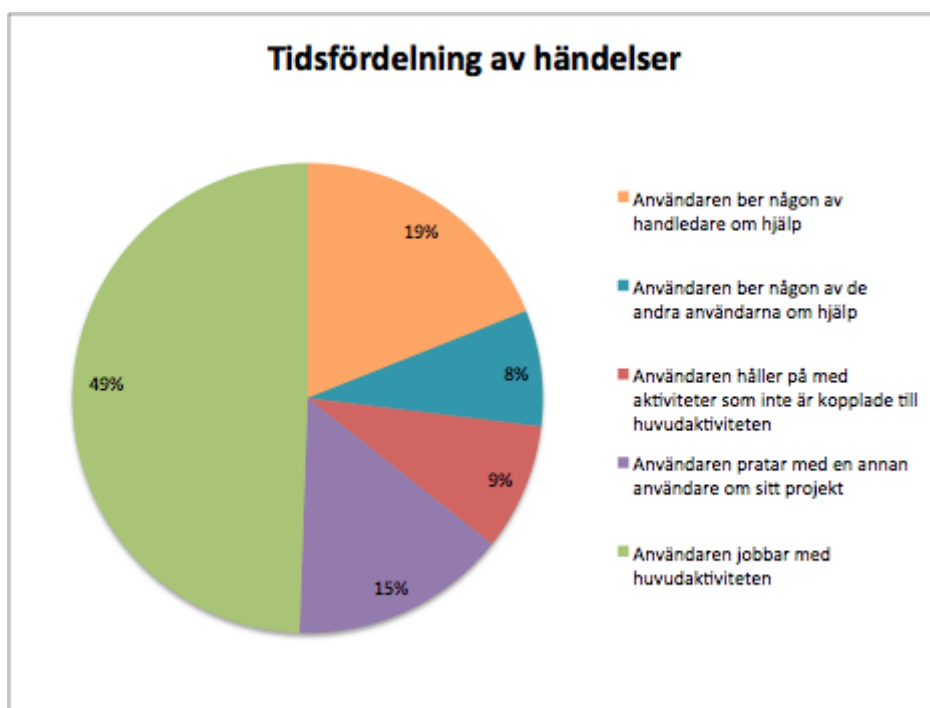
Figur 9.3.5, Observationsperson E, 45 minuter



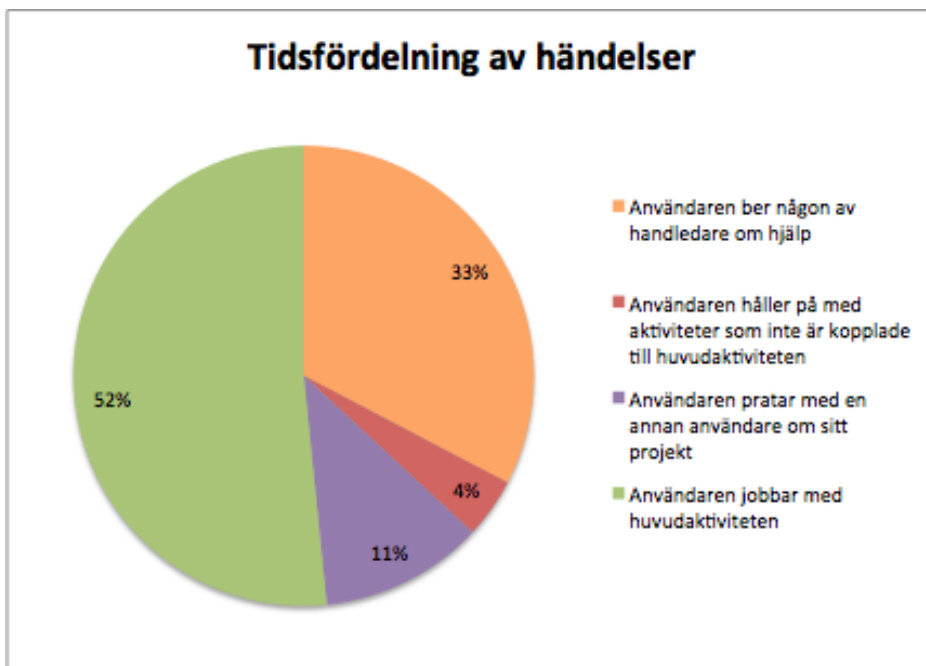
Figur 9.3.6, Observationsperson F, 45 minuter



Figur 9.3.7, Observationsperson G, 60 minuter



Figur 9.3.8, Observationsperson H, 45 minuter



Figur 9.3.9, Observationsperson I, 1h 45 minuter

## 9.4 Enkäten

2016-05-05

Frågor om kodning och programmering

### Frågor om kodning och programmering

Hej!

Vi heter Gustav och Patrik och studerar på Malmö Högskola. Just nu skriver vi en uppsats om programmering för unga människor.

Nu skulle vi vilja ställa lite frågor om programmering till dig! Med programmering menar vi exempelvis Scratch, C#, Python, Processing, HTML eller något annat programmeringsspråk.

Det finns inga dåliga svar, vi blir glada för alla svar!

Tack för din hjälp och glöm inte klicka på knappen "SKICKA" i slutet! :)

**\*Required**

**1. 1. Alder \***

Ange din ålder i siffror

.....

**2. 2. Vad tycker du är roligt att göra på din fritid?**

Till exempel din hobby, idrott eller något annat

.....

.....

.....

.....

**3. 3. Vad tycker du om programmering? \***

För att svara på denna fråga klickar du på en av siffrorna på skalan ett till fem.  
*Mark only one oval.*

1      2      3      4      5

Tråkigt                  Roligt

**4. 4. Vad fick dig att börja på CoderDojo? \***

.....

.....

.....

.....

**5. 5. Programmerade du innan du började på CoderDojo? \****Mark only one oval.*

- Ja
- Nej

**6. 6. Hur länge har du hållit på med programmering? \****Tick all that apply.*

- Längre än ett år
- Längre än sex månader
- Längre än en månad
- Längre än en vecka

**7. 7. Vad tycker du om programmeringen på CoderDojo? \***

För att svara på denna fråga klickar du på en av siffrorna på skalan ett till fem.  
*Mark only one oval.*

	1	2	3	4	5	
Svårt att lära sig	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Enkelt att lära sig

**8. 8. Vad tycker du är roligt att skapa med programmering?**

.....

.....

.....

.....

.....

**9. 9. Har du lärt dig något på CoderDojo som har hjälpt dig med något ämne i skolan? \***

.....

.....

.....

.....

.....

**10. 10. Hur tycker du att det vore att få lära sig programmering i skolan? \****Mark only one oval.*

	1	2	3	4	5	
Dåligt	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Bra

**11. 11. I vilka ämnen i skolan tror du programmering skulle kunna användas?**

Kryssa i ett eller flera ämnen som du tror skulle kunna använda programmering i skolan.  
*Tick all that apply.*

- Bild
- Hemkunskap
- Idrott
- Musik
- Slöjd
- Svenska
- Engelska
- Matematik
- SO (Geografi, historia, religionskunskap, samhällskunskap)
- NO (Biologi, fysik, kemi, teknik)
- Other: .....

**12. 12. Om du kryssat i något/några av ämne ovan, berätta gärna hur du tycker man skulle kunna använda programmering i dessa ämnen.**

.....

.....

.....

.....

.....