Blekinge Tekniska Högskola
Institutionen för programvaruteknik
och datavetenskap

Att hitta ingångar i formandet av programmeringskunskap

En diskussion kring möjligheter och svårigheter i grundläggande programmeringsundervisning inom högskolan.

Kandidatarbete i adb, 10 poäng
för Informationsteknologiprogrammet
2001-05-22

Författare: Maria Alsbjer

Handledare: Lena Trojer

Tack!

Till alla er som på olika sätt bidragit till detta arbete genom att delta i intervjuer och samtal såväl som diskussioner kring själva utformningen vill jag rikta ett stort och hjärtligt tack! Ingen nämnd och ingen glömd.

Maria Alsbjer, Bromölla 22/5 2001

Abstract

Learning to program at beginners' level within the frames of an academic education is a process which causes some students few difficulties while others find it both painful or even insuperable. Why is this so and how is it possible to create entrances in gaining programming knowledge that will open opportunities for everyone willing to learn? With these issues as point of departure I have studied two different educational programs at the Blekinge Institute of Technology (Blekinge Tekniska Högskola) - the program of Information technology (Informationsteknologiprogrammet) and the program of Media technology (Medieteknik-programmet). I have interviewed students and teachers from both programs and I have asked them about earlier experiences of computers and programming, views on programming, difficulties they have experienced when learning to program and possible causes of these difficulties.

The results from the interviews have been reflected through a number of texts that deal with conceptions and conducts in the formation of programming knowledge as well as more general questions about the epistemological points of departure in the formation of all kinds of knowledge, but with focus on programming knowledge. My methodology is established in gender research theories dealing with the scientific processes of knowledge, where questions about the contents of method and theory are being discussed. This connection has made a contextual analysis possible in which I have tried to emphasise general as well as more specific tendencies. That is tendencies which can be identified in the respective educational programs I have studied, as well as tendencies which may be recognised in all academic programming education.

In the final discussion I have emphasised those issues which I consider to be the most significant for the possibilities to make programming knowledge "available" to all having the interest to learn. These issues deal with the general view of knowledge in which programming education all too often is being shaped. It is a view of knowledge which expresses a more or less pronounced view of education as achievement. Questions about what consequences a uniform education might have on different individuals is also taken into account. Means of how to create multiplicity within the education is being discussed as well as the role of the teachers in this context.

With this work I want to contribute to bases of knowledge for development of a pedagogical work, that will/ should benefit students and teachers as well as the subject of programming itself.

Sammanfattning

Att lära sig programmera på nybörjarnivå inom ramen för en högskoleutbildning är en process som för vissa studenter tycks helt oproblematisk medan den för andra ter sig smärtsam eller rent av oöverstiglig. Varför är det så och hur kan man hitta ingångar i formandet av programmeringskunskap som öppnar möjligheter för alla som vill lära sig? Med dessa frågeställningar som utgångspunkt har jag tittat på två olika utbildningsprogram vid Blekinge Tekniska Högskola (BTH) - Informationsteknologiprogrammet respektive Medieteknik-programmet. Jag har intervjuat ett urval studenter och lärare från de båda programmen och ställt frågor bl a kring tidigare erfarenheter av datorer och programmering, synen på programmering, upplevda svårigheter med att lära sig programmera och tänkbara orsaker till dessa svårigheter.

Resultaten från intervjuerna har reflekterats genom ett antal texter som behandlar föreställningar och förhållningssätt i formandet av programmeringskunskap såväl som mer övergripande frågor kring epistemologiska utgångspunkter i formandet av all slags kunskap, men med fokus på programmeringskunskap. Min metodologi är förankrad i genus-forskningsteorier kring de vetenskapliga kunskapsprocesserna, inom vilka frågor rörande bl a metod- och teoriinnehåll problematiseras. Denna förankring har möjliggjort den kontextuella analys i vilken jag försökt lyfta fram allmängiltiga såväl som specifika tendenser, d v s både faktorer som kan härröras till respektive program jag studerat och faktorer som berör programmeringsundervisning i högskolemiljö generellt.

I den avslutande diskussionen har jag lagt särskild vikt vid de frågeställningar som jag uppfattar har den mest signifikanta betydelsen för hur möjligheter kan skapas för att programmeringskunskap ska uppfattas som "tillgänglig" av alla som har intresse av att lära sig programmera. Dessa frågeställningar berör just den övergripande kunskapssyn inom vilken programmeringsundervisning alltför ofta formas; en kunskapssyn som ger uttryck för ett mer eller mindre utpräglat prestationstänkande. Frågeställningar kring vilka konsekvenser en "likriktad" undervisning kan få för olika individer tas också upp och möjligheter till skapandet av mångfald i undervisningen diskuteras liksom lärarnas roll i detta sammanhang.

Med detta arbete vill jag bidra till kunskapsunderlag för ett pedagogiskt utvecklingsarbete - till gagn för studenter och lärare såväl som programmeringsämnet i sig.

Innehållsförteckning:

1. Inledning

Den text du har framför dig utgör ett examensarbete för Informationsteknologiprogrammet vid Blekinge Tekniska Högskola (BTH). Ämnet är programmeringsundervisning på nybörjarnivå inom högskolan - möjligheter och svårigheter. I detta inledande avsnitt följer en beskrivning av bakgrunden till ämnesvalet samt en förklaring till valda arbetssätt. I slutet av inledningen görs en kortfattad genomgång av rapportens uppläggning.

1.1 Problemområde - eller hur jag föll för ämnet

När jag själv lärde mig att programmera (i programspråket java) var vi ganska många i klassen som upplevde programmeringen som någonting svårt och abstrakt. De flesta hade höga förväntningar på den inledande programmeringskursen och tyckte det skulle bli oerhört roligt. De flesta la också ner väldigt mycket arbete på att försöka lära sig, men inte desto mindre brottades vi som var nybörjare med ganska stora problem. Några gav till sist upp javaprogrammeringen och så småningom också utbildningen. Vi som inte gav upp, utan trots allt genomförde laborationer och tentamina, lyckades i alltför många fall dessvärre inte frigöra oss från den gnagande känslan av att egentligen inte alls ha förstått - av att inte ha hittat dörren till ingången.

Länge gick jag omkring med föreställningen att problemet var specifikt för vår klass. Den allenarådande förklaringen skulle vara att vårt program var nytt och därför brottades med barnsjukdomar i form av bristfällig organisation och planering och inte minst bristande erfarenhet av att undervisa på distans. Så småningom insåg jag dock att även om vår situation som pionjärer på ett nytt program medförde en hel del svårigheter så fanns det även studenter, på betydligt mer etablerade program, som konfronterats med liknande problem som vi vad gällde programmeringen.

1.2 Problemavgränsning - att omsätta känslan i ett konkret projekt

Hur omsätter man en känsla i ett konkret projekt? Det gäller att finna en realistisk begränsning i de frågor man ställer och av de verkligheter man väljer att undersöka. För mig var det naturligt att välja att titta på mitt eget program, men eftersom jag ville spegla det mot något gällde det att hitta minst ytterligare ett program med likartade förutsättningar. Även om jag aldrig har haft för avsikt att göra någon direkt jämförelse såg jag det som betydelsefullt att programmen gick att relatera till varandra. Anledningen var att jag därigenom ville visa att problemen - och möjligheterna - inte i första hand ligger på en nivå som enbart berör den enskilda utbildningen. Snarare finns de på en övergripande, strukturell nivå. Samtidigt är jag medveten om att i den övergripande strukturen ryms enskildheter som har betydelse för utfallet. Det generella och det specifika går hand i hand. Det går inte att bortse från det ena utan att förlora något väsentligt i beskrivningen. Därför har jag försökt behålla båda perspektiven.

De två program jag till sist valde att arbeta med som utgångspunkt för mina frågeställningar var Informationsteknologiprogrammet respektive Medieteknikprogrammet. Båda är relativt

nystartade (höstterminen 1997 respektive 2000). De har samma antagningskrav (standard-behörighet D.4.1 vilket innebär matematik C respektive samhällskunskap A). De grundläggande programmeringskurserna ges i samma språk (java). När jag inledde mitt arbete kunde utbildningarna dessutom leda till samma slutexamen (filosofie kandidatexamen i datavetenskap)[1]. Från början hade jag tänkt ta med ytterligare ett program, men jag insåg snart att med den begränsade tid jag hade till mitt förfogande var detta inte realistiskt. Flera av de andra programmen vid BTH finns trots allt representerade indirekt genom alla de informella samtal jag fört med olika studenter och lärare under arbetets gång.

1.3 Frågeställningar - eller vad var det egentligen jag ville ha svar på?

Bland mina inledande funderingar inför arbetet, fanns frågor som berörde varför vissa nybörjarstudenter i programmering upplever ämnet som så svårt och varför andra inte gör det. Jag funderade vidare över om det finns något i läroprocessen som tilltalar vissa, men förvirrar andra och om undervisningen möjligen i praktiken är upplagd utifrån att studenterna förväntas ha en ganska bred datorvana (även om detta aldrig uttalas). Följdfrågan blev då om de studenter som saknar denna datorvana kunde antas ha sämre förutsättningar vad gäller att lära sig programmering. Slutligen funderade jag över om det därutöver går att skönja någon könsmässig skillnad och vilka uttryck den i så fall tar sig.

I de ursprungliga funderingarna utgick jag huvudsakligen från studenternas perspektiv, men eftersom kunskap enligt min uppfattning är något som i stor utsträckning formas i mötet mellan lärare och student(er) såg jag det som väsentligt att ha med båda dessa perspektiv. Dessa funderingar resulterade i ett antal frågeställningar som jag hoppas detta arbete lyckas belysa:

1) Vilken betydelse har bakgrundsfaktorer som olika utbildningsbakgrund, tidigare datorvana (inklusive eventuella erfarenheter av programmering), kön e t c för programmeringsinlärningen?
2) Vilka föreställningar och förhållningssätt finns kring programmering?
3) Vad, mer exakt, är det som de studenter som upplever sig ha svårigheter med inlärningen upplever som problematiskt?
4) Hur upplever de lärare som undervisar i programmering situationen?
5) Hur ser den kontext, det sammanhang ut som undervisningen och individerna formas i och hur påverkar den läroprocessen?

1.4 Metod - de redskap som jag valde att arbeta med

För att belysa mina frågeställningar bestämde jag mig på ett tidigt stadium för att göra inspelade intervjuer med studenter och lärare. Jag hade tänkt begränsa mig till mellan fyra och sex studenter respektive två lärare per program. Under arbetets gång valde jag dock att utöka antalet intervjuer.

Anledningen till att jag valt att arbeta med intervjuer som metod, är att jag såg det som en möjlighet att få fram mer genomtänkta och nyanserade svar än vad som blir fallet vid en enkätundersökning. En intervju ger en helt annan möjlighet till uppföljning av svar genom att

[1] Studenterna vid MT-programmet kan enligt beslut som fattats under våren 2001 erhålla antingen en teknologie kandidatexamen eller en filosofie kandidatexamen i huvudämnet medieteknik.

jag som intervjuare kan ställa följdfrågor som bestäms av intervjusituationen. Vad gäller de intervjuade studenterna såg jag det som väsentligt att göra en uppföljning efter avslutad kurs. Det var då en fördel att i viss mån ha kunnat följa informanten i dennes inlärningsprocess.

De informella samtalen är ett annat av de redskap jag använt mig av i arbetet med föreliggande rapport. Inledningsvis betraktade jag inte dessa informella samtal som något som ingick i min metod. Så småningom insåg jag att de i allra högsta grad utgjorde en del. Likaså ingår de tillfällen då jag suttit med och lyssnat - (gäller framförallt Medieteknikprogrammet) på lektioner, utvärdering och redovisningstillfällen. Det fjärde redskapet utgörs av den litteratur jag har läst - i tyst dialog med de tre övriga instrumenten.

1.4 Metodologi - utgångspunkter i skapandet av kunskap

När jag först började fundera över hur jag skulle närma mig mitt ämne upplevde jag en stor osäkerhet. I den genomgång av hur ett examensarbete bör skrivas som jag deltagit i vid BTH koncentrerade sig mycket av diskussionen kring vikten av att formulera en hypotes, d v s ett antagande som antingen kan bevisas eller motbevisas. Problemet var att jag inte visste vad jag skulle bevisa (eller motbevisa): - "att programmeringsundervisningen var dålig?", eller "att om alla fick börja med att lära sig programmera grafiska användargränssnitt skulle alla svårigheter försvinna?", eller "att med mindre teori och mer hårdkodning skulle världen bli lyckligare?" Säkert är att samtliga påståenden ovan hade gått att bevisa, såväl som motbevisa, genom att med omsorg välja referenser och intervjupersoner som passade det resultat jag önskade uppnå. I intervjusituationen såväl som i litteraturen har jag stött på förespråkare såväl som motspråkare för alla de tre påståendena. Förespråkare respektive motspråkare som säkert skulle ställa upp och skriva under på att "så här är det". Men vad skulle jag i så fall uppnå med det? Skulle jag ha bevisat "sanningen"? Vilken sanning i så fall och vems?

Mitt i dessa bryderier fick jag ett erbjudande om att gå en kurs med rubriken: Metod, metodologi och epistemologi i feministiska forskningsprocesser[2] (benämns hädanefter metod & metodologikursen). Under arbetet med denna kurs började jag förstå vari mitt dilemma låg. De funderingar jag bar på bottnade i en epistemologisk/ kunskapsteoretisk konflikt. Det blev tydligt för mig att jag i min roll som student vid en teknisk högskola befann mig i ett sammanhang som präglas av en positivistisk kunskapssyn. Det är inte så att jag vill påstå att denna prägling kännetecknas av en total blindhet inför den kritik mot "objektiv" kunskap som är nästan lika gammal som positivismen själv. I den kurslitteratur för examensarbete som flitigt används inom högskolan beskrivs två vetenskapliga förhållningssätt varav det ena är positivismen och det andra är hermeneutiken[3]. Det senare förhållningssättet beskrivs som det förras motsats i och med att det kännetecknas av tolkning av verkligheten snarare än bestämning av densamma. Problemet är att i praktiken ställs aldrig den grundläggande frågan: Vilken verklighet och vems?

För att formulera mina egna utgångspunkter har jag haft stor hjälp av den läsning och de diskussioner som förts i metod & metodologi kursen. Två andra kurser har också haft stor

[2] Kursen var en doktorandkurs som gavs i samarbete mellan BTH - Institutionen för Arbetsvetenskap och Medieteknik, forskningsenheten för IT och genusforskning (numera Teknovetenskapliga studier) - och Luleå tekniska universitet - Institutionen för arbetsvetenskap, avdelningen genus och teknik. Kursen var uppdelad i tre avsnitt som var förlagda till olika tidpunkter under perioden september 2000 - februari 2001. Det avsnitt jag diskuterar kring här utgjorde det första avsnittet.

[3] Jag tänker här framförallt på Runa Patels och Bo Davidsons: Forskningsmetodikens grunder.

betydelse i mina epistemologiska funderingar: Den ena är IT:s idéhistoria i genusforskningsperspektiv, den andra är Yrkeskunnande och Teknologi. Båda kurserna gavs vid BTH, Institutionen för Arbetsvetenskap och Medieteknik, forskningsenheten för Teknovetenskapliga studier. I dessa båda kurser har jag fått viktiga redskap i form av texter etablerade i den vetenskapliga världen, d v s texter som har granskats av andra forskare och därmed betraktas som vederhäftiga[4]. Genom denna läsning har jag fått möjlighet att reflektera över kunskapen som begrepp och vad som utgör min egen kunskapssyn. Jag har också haft möjlighet att pröva en del av mina tankegångar i skriftliga analyser, såväl som muntliga diskussioner, vilka varit oerhört värdefulla i min egen tankeprocess.

Synen på kunskap, min egen och andras, spelar således en grundläggande roll i det arbete du har framför dig. Det är dock viktigt att komma ihåg att min ämnesmässiga utgångspunkt är datavetenskap/ adb. Det är tack vare den utgångspunkten jag kan välja de perspektiv jag gör i form av frågeställningar och tolkningar. Hade jag saknat denna ämnesmässiga förankring hade perspektivet blivit ett annat.

1.5 Syfte & målgrupp

Min målsättning har varit att visa på några av de svårigheter, men också möjligheter, som kan uppfattas i formandet av programmeringskunskap på nybörjarnivå. Genom att samtala med såväl lärare som studenter har jag försökt tydliggöra hur skapandet av kunskap är något som sker i mötet dem emellan. De frågeställningar jag beskriver i avsnitt 1.3 har utgjort basen för dessa samtal. Framförallt har jag försökt synliggöra den kontext inom vilken undervisningen tillämpas. Min förhoppning är att arbetet i förlängningen kan ge impulser till nya vägar i förmedlandet av programmeringskunskap. Som potentiella läsare ser jag utbildningsansvariga, studenter och lärare, liksom andra som på olika sätt är involverade i programmeringsundervisningen. Rapporten kan även vara intressant för dem som har ett allmänt intresse av att studera hur läroprocessen formar och formas av de deltagande parterna.

1.6 Rapportens uppläggning

Rapporten är indelad i fem kapitel. De olika kapitlen behandlar i korthet följande:

1) I inledningen beskriver jag mitt ämnesval och bakgrunden till det. Jag förklarar också vilka metoder jag använt och de metodologiska utgångspunkterna för mitt arbete. Slutligen beskriver jag syftet med rapporten liksom vem den riktar sig till.
2) Bakgrundskapitlet utgör en kort beskrivning av de båda utbildningar som ingått i den empiriska delen av undersökningen. Framförallt innehåller det en ingående redogörelse för utformningen av programmeringskurserna vid de respektive programmen.
3) Kapitel tre tecknar en bild av resultaten från den empiriska undersökningen. Underrubrikerna återspeglar de teman som frågeställningarna kretsat kring. Exempel på dessa teman är: innebörder av programmeringskunskap, olika förhållningssätt i förvärvandet av programmeringskunskap, tidigare erfarenheter av datorer och programmering, m m.

[4]Elisabeth Gulbrandsen för i artikeln Objectivity? Reflexivity! Authority?!, som ingår i hennes licentiatavhandling, ett intressant resonemang om hur föreställningar om auktoritet eller brist på auktoritet i en text formas inom den vetenskapliga världen. I detta sammanhang väljer jag dock att bara nämna diskussionen som en fingervisning om att vetenskap och vetenskaplighet som begrepp inte är helt okomplicerade ☺

4) Analyskapitlet är ett försök att återspegla den fortgående dialog mellan fältstudierna och den litteratur jag läst som varit utmärkande för arbetsprocessen. I analysen har jag formulerat tre kunskapsteman under vilka olika aspekter av kunskap diskuteras. Det första temat behandlar synen på kunskap, det andra formandet av kunskap och det tredje diskuterar vägledare i formandet av kunskap.
5) Det sista kapitlet utgör en diskussion kring de, enligt min uppfattning, mest centrala frågeställningarna i arbetet. Jag redogör här för de kontextuella problem jag identifierar i analysen samt för ett resonemang kring möjliga orsaker till dem.

Bilagedelen innehåller en ordlista med programmeringsrelaterade termer som förekommer i arbetet. Den är tänkt att fungera som ett stöd vid läsningen för dem som inte umgås med dessa termer dagligen. Bland bilagorna finns också de frågemallar jag utgått ifrån i samband med intervjuarbetet.

2. Bakgrund

I detta kapitel ger jag en kortfattad beskrivning av de två utbildningar som ingår i undersökningen. Jag redogör också för uppläggningen av och innehållet i de respektive utbildningarnas inledande programmeringskurs. Avslutningsvis gör jag en jämförande kommentar kring kursernas utformning.

2.1 Bakgrund Informationsteknologiprogrammet

Informationsteknologiutbildningen eller IT-programmet som den kallas internt, ges vid Institutionen för programvaruteknik och datavetenskap (ipd). Den är en distansutbildning där studenterna har två alternativ. Det ena innebär att två år läses helt på distans, varefter en högskoleexamen i adb erhålles. Det andra alternativet innebär att det första året läses på distans, tillsammans med dem som läser till en högskoleexamen, och att de två sista åren läses på BTH, i Ronneby. Avslutad utbildning leder till en filosofie kandidatexamen i adb/datavetenskap[5]. Oavsett om studenten läser två eller tre år har utbildningen som syfte att ge en grundläggande datorkompetens och vana vad gäller att arbeta med olika typer av tillämpningar såväl som *"vid att utforma rutiner och verktyg till stöd för det egna arbetet"* .[6] Utbildningen sägs förbereda studenten för *"kvalificerade uppgifter inom systemutveckling, programmering och upphandling av datoriserade informationss ystem"*.[7]

Under distansdelen av utbildningen anordnas föreläsningar och laborationer på högskolan i Ronneby var fjortonde dag. Dessa träffar är inte obligatoriska, men rekommenderas från programledningens sida. Studenterna uppmanas även att bilda lokala studiegrupper. Adressuppgifter och telefonnummer distribueras därför till alla registrerade studenter i början av terminen.

När IT-programmets fjärde årskull av studenter började höstterminen 2000, registrerades 73 studenter, varav 23 kvinnor. Vid vårterminens början, d v s i mitten av januari 2001 stod 38 av dessa - varav 14 kvinnor - fortfarande som registrerade vid programmet.

2.2 Bakgrund Medieteknikprogrammet

Medieteknikprogrammet (MT-programmet) ges vid institutionen för Arbetsvetenskap och Medieteknik (iam), forskningsenheten för Teknovetenskapliga studier. Programmet är nytt för höstterminen 2000 och programmets fysiska placering är i Karlshamn. 67 studenter, varav 13 kvinnor stod som registrerade vid höstterminens början. Då vårterminen inleddes hade en (kvinnlig) student hoppat av. Utbildningen är tre-årig och leder, beroende på inriktning, till antingen en teknologie kandidatexamen eller en filosofie kandidatexamen i medieteknik.

Syftet med utbildningen är att med utgångspunkt från mediebranschens *"behov av yrkesskickliga människor som kan hantera digitala teknik- arbets- och distributionsflöden*

[5] Vilket av ämnena som utgör huvudämne i examen bestäms av vilken inriktning den enskilde studenten väljer. De som väljer datavetenskaplig inriktning har möjlighet att bygga på utbildningen med ett fjärde magisterår.
[6] Enligt den utbildningsbeskrivning som formulerades inför antagningen till höstterminen 2000.
[7] Ibid.

[...] utbilda digitala hantverkare som kan tillämpa sina kunskaper och erfarenheter i många olika digitala medier".[8]

De studerande vid programmet utrustas var och en med en bärbar dator som de får behålla under hela utbildningen. En av tankarna bakom detta är att studenterna inte ska vara bundna av att befinna sig på en bestämd plats i studiearbetet. Studenterna blir mer mobila och kan därmed lättare tillvarata den potentiella kreativitet det innebär att många individer och sammanhang möts. Studenterna uppmuntras också att ta ett stort eget ansvar för den egna läroprocessen. I detta ansvar ingår att formulera den egna målsättningen med utbildningen och möjliga vägar för att nå denna målsättning. Studenterna får t ex redan inför andra året välja de områden inom vilka de är intresserade av att fördjupa sina kunskaper. Lektionsnärvaron är frivillig. Den enda gången man från högskolans sida förutsätter att alla är närvarande är vid den återkommande veckosamlingen som hålls varje fredagsförmiddag.

Vid utbildningen förs en pågående diskussion kring betydelsen av "avlärning", d v s att vi som deltagare i ett utbildningssammanhang måste arbeta med att "lära bort" inarbetade föreställningar som varit dominerande inom traditionellt svenskt utbildningsväsende; föreställningar och värderingar som bygger *på ”konkurrenstänkande, lärarauktoritet och ’kända svar på kända frågor’"*. Istället vill de ansvariga för utbildningen tillvarata både läroprocesser och resultat. Ambitionen är att skapa förutsättningar *"för att den enskildes sociala och personliga kompetens [ska kunna] växa vad gäller t ex ansvar, samarbete, omdöme och ett kritiskt förhållningssätt."* För att göra det möjligt att skapa dessa förutsättningar är det betydelsefullt att betrakta deltagarnas läroprocesser i ett tre-års perspektiv[9].

I programbeskrivningen förklaras de förhållningssätt de ansvariga vill utgå från i utbildningssituationen. I korthet betonas här vikten av att både tillvarata traditionella undervisningsformer, t ex i form av föreläsningar och seminarier som att försöka finna nya, bl a genom integrerade kurser och andra tvärvetenskapliga öppningar. Ett sätt att åstadkomma detta är genom att lämna utrymme för förutsättningslösa idésamtal mellan studenter, lärare och externa intressenter. De tre delar programmet vill integrera är:

1) **Datavetenskap, medieteknik och gestaltning** - innefattar grundtekniker inom de olika huvudämnena.
2) **Reflektion** - omfattar bl a projektledning, människa- och datorinteraktion, cyborgteorier, IT:s idéhistoria mm.
3) **Projekt** - innebär att kurserna i grundteknik och reflektion ska tillämpas i olika digitala medier genom projekt som bygger på idéskapande, planering och finansiering, genomförande samt reflektion.

[8] Detta citat liksom övrig information i min beskrivning av MT-programmet i detta avsnitt (2.2) är, då inget annat anges, hämtad från *Programbeskrivning version 7*, författad av Peter Ekdahl 00-08-25. Att jag valt använda detta som källa istället för den officiella utbildningsbeskrivning som publicerades i samma serie med informationsblad som IT-programmets utbildningsbeskrivning härrör från, beror på att jag såg det som viktigt att använda den källa som de studenter som sökt till programmet senast kan ha kommit i kontakt med. Denna beskrivning skiljer sig nämligen på vissa väsentliga punkter från den ursprungliga officiella utbildnings-beskrivningen. Däremot stämmer den väl överens med den presentation av programmet som görs i BTH:s utbildningskatalog för 2001/02.

[9] Betydelsen av att betrakta deltagarnas läroprocesser i ett tre-års perspektiv, istället för ”termin för termin”, har särskilt betonats i de återkommande samtal jag har haft med de ansvariga för utbildningen.

Utbildningen är uppbyggd kring olika projekt (3) - ett för varje termin. För att realisera de olika projekten planeras relevanta kurser in, som kan ge kunskaper om de - för sammanhanget - nödvändiga grundteknikerna (1). Inom de olika projekten vill de ansvariga att ett kontinuerligt utrymme för reflektion skapas genom att studenternas erfarenheter och kunskaper tas tillvara. Ett av medlen för att skapa detta utrymme är de inplanerade kurser som till sitt innehåll problematiserar relationen(erna) människa – dator, diskuterar idéhistoriska perspektiv av IT, konkretiserar frågeställningar kring projektledning, o s v (2).

2.3 Programmeringskursernas uppläggning

2.3.1 IT-PROGRAMMET, ALLMÄNT

IT-programmets programmeringskurs, ddv 115[10], är en 10-poängskurs som består av vad som tidigare varit två separata 5-poängskurser. Grundläggande IT, dab 100 respektive Programmering i java, ddv 104 lästes tidigare under varsin läsperiod. 10 poäng företagsekonomi läses som parallell kurs, utdragen över den första terminens båda läsperioder. Genom att slå samman de båda inledande kurserna i datavetenskap och låta den hopslagna kursen löpa på halvfart under 20 veckor, hoppas programledningen uppnå positiva effekter för javainlärningen i och med att bearbetningsperioden blir längre. Tidigare års studenter har i kursutvärderingar framfört att inlärningen blivit alltför komprimerad då man läst 5 poäng java på halvfart under 10 veckor.

IT-programmets inledande år ska enligt programförklaringen kunna genomföras helt på distans. Cirka var fjortonde dag anordnas dock förläsningar och/ eller laborationer för dem som vill, och har möjlighet att, resa till Ronneby. I år fanns dessutom en studiegrupp i Höganäs med egen handledare. Denna studiegrupp hade tillgång till en datasal som tillsammans med handledarens lön finansierades av Höganäs kommun. Det var också de som tagit initiativet till denna lösning. Momenten i java har presenterats genom fem föreläsningar och sju handledda laborationstillfällen. Under terminen blev en föreläsning och ett laborationstillfälle i Ronneby inställda. De moment som man skulle ha gått igenom vid dessa tillfällen fick studenterna läsa in på egen hand. All övrig handledning utfördes via e-post - antingen genom kommunikation i den nyhetsgrupp som var kopplad till kursen eller genom att studenter och handledare kommunicerade direkt med varandra. Från högskolans sida uppmuntrades användningen av nyhetsgruppen - det virtuella klassrummet - för att alla skulle kunna få ta del av frågor och svar. Handledarna la också ut frågor av allmänt intresse.

De personer som har varit involverade i kursen var en extern förläsare - tidigare student vid skolans Dataingenjörsprogram och numera delägare i ett utbildningsföretag, samt fyra IT-studenter (två tredje-års och två fjärde-års), anställda som amanuenser. I amanuensrollen ingick ursprungligen enbart att de skulle fungera som labbhandledare. I handledaruppdraget ingår bl a vägledning vid laborationernas genomförande samt rättning av desamma och assistans vid tentamensrättning. Vid terminens början utökades uppdraget till att även omfatta utformningen av laborationerna samt vecko- och kontrolluppgifter och all rättning av tentamina. Som formellt kursansvarig har IT-programmets programansvarige fungerat.

[10] DDV115- Grundläggande IT och programmering i java - är den officiella kurskoden för IT-programmets inledande programmeringskurs.

2.3.2 PRAKTISKA MOMENT FÖR IT-PROGRAMMET

De studerande vid IT-programmet skulle under höstterminen genomföra fyra obligatoriska laborationer. Stoppdatum för inlämning varierade på mellan två och fem veckor, beroende på laborationens omfattning[11]. Dessutom gjorde de ansvariga handledarna kontrollfrågor till de två första laborationerna som avsåg täcka in de begrepp studenterna borde ha lärt sig efter genomgången laboration. Kontrollfrågorna fyllde funktionen av självtest, d v s studenterna kunde testa sina kunskaper, diskutera dem i det virtuella klassrummet och fick efter en vecka svaren till frågorna. Labbhandledarna la även ut s k veckouppgifter vilka också hade funktionen av självtest och där diskussion i det virtuella klassrummet uppmuntrades. I slutet av november slutade labbhandledarna att lägga ut veckouppgifter med hänvisning till att kursansvarig beslutat att studenterna skulle ges möjlighet att skrapa ihop bonuspoäng till den kommande tentamen genom att utföra tester i ett webbaserat testsystem[12]. Studenterna kunde erhålla bonuspoäng genom att genomföra de elva första testerna i systemet med minst 80% rätt på varje test. Varje genomfört test med minst 80% rätt genererade 1% på tentan. Systemet var uppbyggt så att nya frågor slumpades fram varje gång en person genomfört ett test, vilket medförde möjligheten att testa sig flera gånger.

För att studenterna skulle kunna genomföra laborationer i java krävdes det att de installerat JDK (Java Development Kit) på sina respektive datorer. JDK var den utvecklingsmiljö som användes vid utvecklandet av javaprogram. Som inledande övning fanns därför en laboration 0 som noggrant beskrev var man kan få tag på JDK och hur det installeras. Vidare ingick en beskrivning av hur man kunde skriva ett enkelt program[13] samt hur koden kompileras respektive exekveras. Slutligen fanns också en beskrivning av en normal arbetsgång vid skapandet av program samt förslag på olika editorer som kunde användas, varav vissa innehöll stödfunktioner som t ex radnumrering och färglagd kod.

De fyra laborationer som studenterna hade i uppgift att redovisa var uppbyggda efter ett gemensamt tema. Avsikten var att laborationerna skulle bygga på varandra. I den första laborationen var det huvudsakliga syftet att studenterna stiftade bekantskap med, de för objektorienterad programmering grundläggande begreppen, objekt och klass. Det var dock fler begrepp/ koncept som tränades genom laborationen som bestod i att var och en av studenterna förväntades skapa en klass som representerar en CD-skiva med dess typiska data som titel, artist, utgivningsår, längd på skivan mm. Metoder skulle skapas för att visa (returnera) såväl som ändra dessa data. För att testa metoderna i CD-klassen måste studenterna även skapa en testklass. Efter genomförd laboration förväntades studenten förutom att ha fått förståelse för begreppen klass och objekt, också förstå innebörden av programmeringstekniska begrepp som variabel (attribut), metod, konstruktor, tillgänglighet (public respektive private), strängar, heltal, flyttal, tilldelning, main-metoden, instansiering, parameter.

I den andra laborationen var syftet att studenterna lärde sig att förstå och använda begreppet arv. Uppgiften bestod av att studenten skulle skapa ett mediaarkiv (i form av en generell basklass, eller mer korrekt en abstrakt superklass) med tre s k subklasser för att representera olika typer av media; video, cd och bok som skulle ärva gemensamma data och funktionalitet

[11] Senare inlämningar godtogs också, men rättningen av dessa laborationer prioriterades inte och alla studenter uppmanades att försöka hålla inlämningstiderna för sin egen skull, så att de skulle slippa halka efter.
[12] Testsystemet återfinns på adress: http://cw.prenhall.com/savitch/ (nerladdad 2001-02).
[13] Programexemplet var det i programmeringssammanhang klassiska Hello World, som skriver ut just Hello World på skärmen.

från superklassen media. De koncept/ begrepp som studenten förhoppningsvis hade fått förståelse för efter genomförd laboration var: arv, subklass, superklass, abstrakt klass, abstrakt metod, polymorfism, returvärde, extends, tillgänglighet (protected).

Den tredje laborationen hade som huvudsyfte att ge förståelse för begrepp relaterade till datastrukturer (vektorer, mängder) och arv (superklass, subklasser, inkapsling genom användande av public/ private/protected och super). Uppgiften bestod i att utveckla det påbörjade mediaarkivet till ett effektivt registerprogram. Registret (mediarkivet) skulle förändras så att det innehöll vissa attribut (variabler) och metoder, varav en del kunde återanvändas från föregående laboration. De funktionella kraven bestod av konstruerandet av metoder för: skapande av ny post (t ex inläggning av en ny skivtitel), ändring av befintlig post, borttagande av post, sökning, listning samt felhantering. Studenterna skulle även skapa ett menysystem så att användaren lätt skulle kunna komma åt dessa funktioner.

I den fjärde och sista laborationen var huvudsyftet att skapa förståelse för hur java kan användas till att programmera grafiska gränssnitt samt hur man sparar till fil. Uppgiften var att skapa ett grafiskt användargränssnitt till registerprogrammet som gjordes i den föregående laborationen. Dessutom skulle de nya uppgifter som en användare vill lägga in i systemet kunna sparas till fil. De funktionella kraven i övrigt var i stort sett desamma som i föregående laboration.

2.3.3 MEDIETEKNIKPROGRAMMET, ALLMÄNT

Medieteknikprogrammets inledande programmeringskurs har den officiella kurskoden dpt 100 - Objektorienterad programmering i java. Detta är en fempoängskurs som normalt läses på halvfart under tio veckor. Ansvariga vid Medieteknikprogrammet har precis som ansvariga vid IT-programmet haft ambitionen att ge studenterna en längre bearbetningstid för programmeringskunskaperna än vad som normalt är fallet vid BTH:s programmeringskurser. För att åstadkomma detta har man vid MT-programmet valt att låta den inledande javakursen löpa över hela terminen med en lektion i veckan. De kurser studenterna läst parallellt är webbproduktion 5 poäng (också över hela terminen), databaser 5 poäng samt datorteknik och operativsystem 5 poäng.

Undervisningen har varit baserad på lektioner som har getts en gång i veckan i grupper om ca 15 personer. Lektionerna har ibland varit av föreläsningskaraktär och ibland mer av seminariekaraktär där man i gruppen har diskuterat olika programmeringsbegrepp och/ eller lösningar. Eftersom alla studenter vid MT-programmet har utrustats med en bärbar dator har lärarna haft möjlighet att gå igenom exempel som studenterna direkt har kunnat testa - en möjlighet som de kursansvariga också aktivt använt sig av.

De kursansvariga vid MT-programmet rekryterades till att börja med för att utforma nya laborationer. Skälet var att programledningen ville få fram nya idéer kring laborationernas utformning i ett försök att skapa en ny kultur kring programmeringen. De två studenter som rekryterades för uppdraget blev rekommenderade av MDA-programmets programansvarige. De engagerade studenterna läser sista året vid MDA-programmet och är båda mycket intresserade av och duktiga på att programmera. De är framförallt intresserade av spelprogrammering och ägnar en stor del av sin fritid åt detta. Uppdraget för deras del utökades till att vara labbhandledare för att till sist också omfatta rollen som kursansvariga.

2.3.4 PRAKTISKA MOMENT MEDIETEKNIK-PROGRAMMET

I kursen ingick tre laborationer samt ett antal övningar. Som avslutning fick alla studenter arbeta med ett valfritt programmeringsprojekt i grupper om 2-3 personer[14]. Som examination genomfördes en s k codereview (kodgenomgång) där de olika individerna i projektgrupperna delades in i olika redovisningsgrupper. Var och en skulle sedan presentera koden för sin projektgrupps program under 15 minuter och sedan aktivt delta i de andra studenternas redovisningar genom att ställa frågor kring det programmeringstekniska utförandet. Varje individ måste således formulera sin egen förståelse av såväl det projekt hon/ han deltagit i som av de andra studenternas projekt. Laborationerna var till en början obligatoriska, men de kursansvariga bestämde sig ganska snart för att de inte skulle vara det. Vid tveksamma fall vid bedömningen av den slutgiltiga examinationen kunde dock genomförandet av laborationerna vara utslagsgivande.

Vid utformningen av laborationerna hade de kursansvariga inte någon uttalad ambition att dessa skulle följa någon röd tråd. Man ville hellre fokusera på att träna olika begrepp/ koncept, samt på att skapa laborationer som skulle kännas grafiskt tilltalande. Detta gjorde man i förhoppningen om att studenterna skulle känna sig mer motiverade och tycka att det kändes roligare om det "hände något på skärmen".[15] De kursansvariga hade också i åtanke de studenter som redan kunde en del programmering vilka man hoppades skulle bli stimulerade av dessa ganska okonventionella laborationer. För att kombinera känslan av att "något händer på skärmen" med den verklighet det innebär att undervisa nybörjare i programmering valde lärarna att ge studenterna mycket färdig kod, i vilken enbart mindre förändringar behövde utföras. En ytterligare målsättning med detta var att studenterna skulle lära sig genom att läsa kod.

Som inledning till de laborationer studenterna skulle utföra på egen hand (samarbete var tillåtet) utfördes en övning benämnd laboration 0 under lektionstid. Denna övning gick ut på att studenterna skulle bekanta sig med programmeringsspråket java genom att ändra på några variablers värde i ett färdigt program. Programmet var en applet som visade en text som rullade över en färgad bakgrund. Genom övningen tränade studenterna sig i att använda programmet UltraEdit, som var den texteditor som rekommenderades för skrivandet av kod, samt på att kompilera. Nödvändig programvara (JDK respektive UltraEdit) hade dessförinnan laddats ner och installerats av studenterna på den egna datorn.

Inför varje laboration höll de kursansvariga en lektion där målsättningen var att gå igenom de viktigaste begreppen som berörde den aktuella laborationen. Man gjorde även skriftliga beskrivningar som delvis överlappade de muntliga. Samtliga laborationer hade stoppdatum, men dessa var inte absoluta. De kursansvariga försökte dock uppmuntra till att laborationerna skickades in före introduktionen av en ny laboration. Samtliga laborationer gicks igenom före mitten av oktober. Detta innebar att i princip en ny laboration per vecka presenterades. Tankarna med att gå igenom laborationerna i detta snabba tempo var flera. Det handlade dels om att få igång studenterna genom att "chocka" dem lite, men också om att de som redan ägnat sig en del åt programmering inte skulle tappa sugen redan från början p g a det långsamma tempot. Delvis och kanske framförallt handlade det om att lämna resten av terminen öppen för ett mer individuellt tempo. De som kände behov av att repetera skulle få tid till detta innan det var dags att fördjupa sig i det avslutande projektarbetet. De som tyckte sig behärska de grundläggande bitarna i programmering skulle istället kunna ägna mer tid åt

[14] Några har valt att arbeta på egen hand och i något fall har gruppen bestått av fler än tre personer.
[15] Traditionellt lär sig nybörjarstudenter att skapa program som är helt textbaserade som i IT-programmets fall.

projektarbetet genom att gå direkt in i detta. Varje grupp (som skapades av studenterna själva) fick välja ämne och nivå för sitt projekt. Tanken var att alla därigenom skulle ges möjlighet att arbeta utifrån sin kunskapsnivå och sina förutsättningar. Det enda kravet var att varje grupp skapade ett eget program.

I den första laborationen fick studenterna koden till ett halvfärdigt bilspel, gjort som applet. Det som saknades var själva bilen och uppgiften bestod i att skapa denna bil samt skapa metoder för bilens rörelser framåt, bakåt och åt sidorna. Syftet med laborationen var att ge en första insikt i vad klasser, objekt, metoder och variabler är samt hur objekt kommunicerar med varandra. Förståelse skulle även ges för programmeringstekniska begrepp som konstruktor respektive åtkomst.

Även i den andra laborationen fick studenterna ett halvfärdigt program som de skulle färdigställa. Uppgiften bestod av att komplettera ett delvis uppbyggt register för olika media. Den färdiga kod som studenterna hade tillgång till bestod av:

- **GUI-klass** för att hantera det grafiska gränssnittet (knappar, textfält, listor mm).
- **Knapplyssnar-klass (ButtonListener)** som kontrollerar om någon knapp i GUI:t har blivit nedtryckt (eller klickad på).
- **MediaStorage-klass** som fungerar som en lagringsplats för alla medier.
- **Media-klass** som utgör basklass (superklass) till övriga medier och som innehåller generella egenskaper giltiga för alla former av media.
- **Startklass** i vilken finns metoder (e g main-metoden) för att starta programmet.

Det som återstod för studenterna att göra var att skapa klasser som representerar de olika mediatyperna bok, cd och videofilm samt att se till att arvsstrukturen fungerade (vilket bl a krävde förståelse av abstrakta klasser respektive metoder och hur dessa fungerar - samt principen för polymorfism, även om termen aldrig användes). Studenterna måste också kunna skapa nya objekt av de olika mediatyperna och säkerställa att dessa lagrades i MediaStorage. Studenterna skulle även kommentera klassen MediaStorage, d v s beskriva vad den gör.

I den tredje och sista laborationen skulle studenterna färdigställa koden till ett s k "hänga gubbe -spel", som är en typ av ordspel. Målsättningen med laborationen var att skapa en djupare förståelse för kommunikationen mellan objekt, skillnaden mellan klass och objekt, variabel och attribut. Studenterna skulle även få insikt i användningen av lagringsstrukturer i form av fält samt styrfunktioner (if-satser) och loopar (for och while).

Den färdiga koden bestod av en Start-klass, en GUI-klass (innehållande spelets grafik), en KeyboardListener-klass (lyssnar efter tangenter som trycks ned, d v s i det här fallet vilka bokstäver som väljs), en ButtonListener-klass och en klass för GameRules, d v s den klass som innehåller spelets regler. Den primära uppgiften för studenterna bestod i att färdigställa den sistnämnda klassen. Förutom att hitta på ord och lägga in dessa i ett förkonstruerat fält, måste studenterna kunna hantera de inmatningar som görs via tangentbordet så att GUI-klassen får information om huruvida den ska skriva ut bokstäver eller rita (delar av) en hängd gubbe. Varje bokstav som spelaren använt sig av skulle också skrivas ut på skärmen, så att spelaren slipper hålla dessa i huvudet. Kursansvariga hade utformat s k pseudokod för en metod som kan kontrollera om en bokstav finns i ett ord. Den slutgiltiga koden fick dock studenterna skriva själva. De skulle också beskriva hur programmet hoppar i koden efter att en spelare tryckt ned en tangent för att gissa på en bokstav. Studenterna skulle m a o beskriva de metoder programmet går in i, i vilken ordning och varför.

2.4 Jämförande kommentarer

Av genomgången ovan framgår tydligt att pedagogiken i programmeringskurserna på de båda programmen skiljer sig en del åt. Nedan försöker jag belysa dessa skillnader. En förståelse av de olika sätten att förmedla programmeringskunskap är viktig för förståelsen av den fortsatta diskussionen. Jag vill dock betona att det inte handlar om att dikotomisera, d v s skapa ett antingen-eller-perspektiv där det ena sättet framhålls som bättre eller sämre än det andra. De respektive programmen har sina olika förutsättningar; det jag ser som ett gemensamt drag är ambitionen att skapa programmeringskurser anpassade till de egna villkoren och till målen med programmet i sin helhet.

I IT-programmets fall har man valt att arbeta utifrån en modell där stor vikt läggs vid att skapa förståelse för olika programmeringstekniska begrepp. I de laborationsbeskrivningar studenterna får betonas vilka begrepp/ koncept de bör ha lärt sig efter respektive laboration. Detta är i sig en fingervisning om vad de bör läsa in sig på. I en kurs som drivs på distans har det ansetts naturligt att betona vikten av att studenterna lär sig att själva inhämta nödvändig kunskap (via böcker, tidskrifter, Internet). Även deltagande i interna och externa nyhets-/ diskussionsgrupper uppmuntras. För att kunna använda dessa redskap krävs att studenterna behärskar vissa begrepp/ koncept. Det är t ex svårt att slå upp något i en bok om man inte vet vad man ska leta efter. Samma sak gäller vid deltagande i ämnesbaserade diskussionsgrupper. Det är lättare att fråga och få svar om alla använder en gemensam terminologi.

IT-programmet inledande programmeringskurs har av många av studenterna ansetts som svår, trots försök till förändringar allt sedan den första kursen gavs 1997. Detta har märkts i tentamensresultat, utelämnade laborationsinlämningar samt av de utvärderingar som gjorts vid kursslut. IT-programmet har också haft ett stort antal studenter som slutat i förtid, varav flertalet redan under första året[16]. Kursansvarig för javakursen, tillika programansvarig, gick i mitten av höstterminen ut med ett riktat brev till de studenter som inte visat någon aktivitet i javakursen (d v s som inte skickat in några laborationer). Syftet var att försöka motverka antalet avhopp, som programledningen misstänker beror bl a på svårigheter med att läsa programmering på distans. Innan detta brev skickades ut hade allmän information gått ut till samtliga studenter via den interna diskussionsgruppen om vikten av att arbeta med javakursen och om brevet som skulle skickas till berörda studenter. Det som sades i brevet var i korthet att man från högskolans sida konstaterat att mottagaren av brevet inte varit aktiv i javakursen. Man ville därför att vederbörande skulle ta kontakt antingen med kurs/ programansvarig eller med institutionssekreteraren och eventuellt ta en diskussion med någon av studievägledarna. Mottagaren av brevet fick också argument till varför man från högskolans sida ansåg att hon/ han skulle satsa på javakursen. Bl a framhölls att de verktyg som javakursen tillhandahåller behövs för att klara efterföljande kurser och att därför javakursen är speciellt olämplig att ha släpande efter sig. Dessutom påpekades det faktum att tillgången till handledare som kan besvara frågor är begränsad till den tid som kursen ges.

Bland de studenter som tog kontakt med högskolan efter att ha mottagit brevet från programledningen, visade det sig att en stor andel hade arbete vid sidan av studierna och att detta var en viktig orsak till att de ansåg sig ha svårt att hinna med. Vid den kursutvärdering som gjordes ingick därför en fråga i vilken omfattning den studerande eventuellt arbetade vid

[16] Distansutbildningar har generellt en lägre genomströmning än andra utbildningar, enligt civilingenjör Christina Björkman som är verksam inom datavetenskap vid bl a Uppsala Universitet och som även tjänstgjort som studievägledare inom ämnesområdet. En diskussion kring orsakerna till detta ligger dock utanför ramen för detta arbete.

sidan av. Resultatet från utvärderingen visade att av de 27 studenter som deltog vid det första tentamenstillfället, arbetade 11 av dem hel- eller deltid (majoriteten arbetade deltid). Eftersom man från högskolans sida hade anledning att tro att det fanns ännu fler som arbetade bland dem som inte försökt göra tentamen skickades ett informationsbrev ut via nyhetsgruppen. I detta tog programledningen upp att man såg det som ett bekymmer att alltfler studenter tycktes hamna på efterkälken med sina studier p g a att de arbetade vid sidan av. Konsekvenserna man lyfte fram var dels att den individuella studiegång som då skulle vara alternativet - kan bli komplicerad eftersom många kurser är beroende av varandra, dels att högskolan får problem att hålla platser till efterkommande kurstillfällen p g a att så få av IT-studenterna läser på heltid.

Vid Medieteknikprogrammet har kursansvariga medvetet valt att inte fokusera så mycket på olika programmeringstekniska begrepp/ koncept utan enbart tagit upp de allra mest grundläggande så som beskrivs ovan. Avsikten med detta har varit att studenterna ska lära sig programmera genom att praktisera programmering. Terminologin, har man menat, är i regel lättare att ta till sig när studenten kommit in lite mer i det praktiska tillvägagångssättet. Samtidigt har lärarna varit väl medvetna om att det hela är en balansgång. Medie-teknikprogrammets pedagogik bygger till stor del på att man lär sig bäst genom att hitta sin egen inlärningsprocess, vilket förutsätter ett aktivt arbete från den enskilde studenten vad gäller att söka information. De kursansvariga för programmeringskursen har definitivt inte saknat insikt i att det kan orsaka problem i kunskapssökandet, om studenten saknar förmåga eller verktyg att formulera vad det är hon/han letar efter. Å andra sidan har MT-programmets studenter haft helt andra möjligheter än IT-programmets till kontinuerlig handledning. Visserligen sägs det i programförklaringen att "vi är mobila" tack vare de bärbara datorer som studenterna utrustas med, vilket *"innebär att du kan befinna dig var som helst i världen och ändå känna dig hemma"*[17]. Denna möjlighet till rörlighet används av många studenter såtillvida att de inte alltid befinner sig på högskolan utan istället väljer att arbeta någon annanstans. Det faktum att de kan åka till högskolan när de vill och där få hjälp antingen av klasskamrater eller av handledare utgör dock en väsentlig skillnad mot den distansundervisning som kännetecknar IT-programmet.

När det gäller arbete vid sidan av studierna så förekommer detta även vid MT-programmet om än inte i samma utsträckning som vid IT-programmet. De studenter som arbetar tycks heller inte ha haft några problem att hinna med kurserna, vilket delvis kan bero på att de besitter förkunskaper inom berörda områden. Några har datatekniska studier med sig i bagaget, andra arbetar inom områden som relaterar till de som studeras. Dessutom kommer MT-studenterna inför andra och tredje året att få välja vilka/ vilken typ av kurser de vill gå. Det innebär att det individuella valet sker betydligt tidigare än vid IT-programmet där man i princip inte kan välja kurser förrän inför det sista året.

[17] Ur välkomstbrev till de studenter som sökte utbildningen inför höstterminen 2000. Framgår även av utbildningskatalogen för höstterminen 2001/02.

3. Genomförande

Det empiriska materialet utgör basen för mitt arbete. I nedanstående kapitel redogör jag dels för hur materialinsamlingen gått till, dels för min bearbetning av det.

3.1 Beskrivning av intervjuarbetet

Jag har intervjuat åtta studenter från MT-programmet och sju från IT-programmet. Av de intervjuer som gjordes med studenter från IT-programmet genomfördes sex muntligen och en skriftligen. Fördelningen mellan män och kvinnor bland de intervjuade studenterna är i stort sett jämn. Åldersfördelningen är också relativt jämn men med en viss slagsida mot något "äldre"[18] studenter vad beträffar MT-programmet. Bland inblandade lärare har jag intervjuat fyra av fem från IT-programmet[19] och båda de lärare som ansvarade för MT-programmets programmeringskurs samt en lärare som gjorde ett par inhopp som föreläsare i programmeringskursen. Därutöver har jag intervjuat en av programmeringslärarna vid MDA-programmet[20] i Ronneby. I materialet använder jag ofta lärare och handledare synonymt om jag inte finner det befogat att betona skillnaden. Skälet till detta är jag på så sätt lättare bevarar de iblandades anonymitet. Dessutom tycks skillnaden ur studenternas synvinkel ha en marginell betydelse.

De frågor jag ställde till de individer som ställde upp och lät sig intervjuas, utgick från gemensamma mallar - en med frågor till studenterna och en med frågor till lärarna. Jag följde dock inte dessa mallar slaviskt utan lät intervjuerna utvecklas och formas utifrån sammanhang och person. Mallarna fyllde dock en viktig funktion vad gällde att på ett kortfattat sätt förbereda dem jag skulle intervjua om karaktären på frågorna. Frågorna i mallarna har utformats med utgångspunkt från de övergripande frågeställningar jag formulerade i problemformuleringsfasen. De har reflekterats genom delar av den litteratur jag läste i denna fas. De har också diskuterats med olika referenspersoner som utgjort ett viktigt stöd för detta arbetes framväxt.

Jag genomförde två intervjuomgångar med studenterna. Den första intervjuomgången genomfördes under oktober månad, vilket innebar att studenterna hade börjat komma in i programmeringen, men fortfarande brottades med en hel del svårigheter av nybörjarkaraktär. Den andra intervjuomgången genomfördes som en uppföljning efter att programmeringskursen avslutats i mitten av januari. IT-studenterna hade då inlett två nya kurser - en 10-poängskurs i diskret matematik som löper över hela terminen och en 5-poängskurs i systemutveckling som läses under halva terminen. Studenterna vid MT-programmet hade börjat läsa två helterminskurser på 5 poäng - en i videoproduktion och en i datastrukturer och algoritmer. Därutöver hade de börjat läsa en halvterminskurs på 5 poäng i videoteknik. I några av de uppföljande intervjuerna återspeglas de första intrycken av dessa kurser i förhållande till de som genomfördes under höstterminen. Två studenter deltog inte i den uppföljande intervjun, en från vardera programmet[21].

[18] Äldre i förhållande till genomsnittsåldern hos kursdeltagarna.
[19] Totalt var sex lärare inblandade om man räknar kursansvarig som dock inte deltog i undervisningen.
[20] MDA står för Människor, Datateknik, Arbetsliv.
[21] Den ena av studenterna hade bestämt sig för att avsluta utbildningen och bidrog ändå till materialet genom att förklara orsakerna till avhoppet i ett e-postmeddelande.

Samtliga intervjuer har bearbetats genom att jag lyssnat igenom de inspelade banden och loggat materialet i enlighet med den metod Sharan B Merriam (1994) rekommenderar. Metoden innebär att man gör kortare (ej ordagranna) anteckningar av den intervjuades svar och kommentarer och kompletterar dessa med egna reflektioner. Utskriften av logganteckningsarna har gjorts i direkt anslutning till genomlyssningen och är fylligare än anteckningarna i sig. Jag har alltså fyllt ut texten med bindeord och sammanhang som intervjuloggarna ger en antydan om (ungefär som stödanteckningar från en föreläsning väcker sammanhanget till liv för den som skrivit anteckningarna). Texten blir därmed lättare att läsa även för andra än mig själv. Jag har dock inte vinnlagt mig om att utforma en grammatiskt korrekt text utan karaktären av telegramstil är uppenbar. I den mån jag återger citat i den följande texten så är dessa i princip ordagranna. I några fall har jag strukit utfyllnadsord för att tydligare framhäva det som sägs. Var och en av de intervjuade studenterna har tagit del av sina utskrifter. De intervjuade lärarna har inte tagit del av sina utskrifter utan endast beretts möjlighet att ta del av berörda textpartier i detta arbete. Skälet till denna synbara diskriminering är att de intervjuer jag genomförde under hösten tog en avsevärd tid att bearbeta. Att skicka utskrifterna från dessa för en skriftlig kommentar drygt ett par månader efter genomförd intervju bedömde jag som mindre lyckat. I studenternas fall fyllde utskrift-erna funktionen av att återknyta till höstens samtal. I lärarnas fall medförde avsaknaden av uppföljande intervju att ingen sådan naturlig koppling fanns.

3.2 Analysmetod och metodologiska utgångspunkter

Analysen av intervjumaterialet har skett successivt och kontinuerligt. Framförallt har den skett parallellt och i dialog med den läsning jag gjort (se kapitel 4). Frågeställningarna i de mallar jag använde vid intervjuerna bildar i sig vissa teman som jag försöker belysa. Intervju-resultaten har också visat på tendenser som bildat nya teman.

I denna tematiska genomgång har jag haft tydliga utgångspunkter. Jag har t ex sett det som självklart att det generella och det specifika måste belysas parallellt för att det ska vara möjligt att förstå övergripande strukturer såväl som enskildheter. Jag har försökt behålla ett brett och mångsidigt perspektiv, men är också fullt medveten om mina egna erfarenheters och förförståelsers betydelser för hur jag genomfört och analyserat arbetet. De förhållningssätt jag använt mig av är förankrade i *genusforskningsteorier* kring de vetenskapliga kunskaps-processerna[22]. Inom dessa problematiseras bl a frågor kring metod- och teoriinnehåll inom olika discipliner. Framförallt kritiseras den positivistiska kunskapssyn som bildar utgångspunkt för påståenden om kunskap som "objektiv". Istället framhävs vikten av att tydliggöra de egna utgångspunkterna och vems respektive vilken verklighet man anser sig skildra. Den centrala utgångspunkten i min analys har varit frågor kring synen på kunskap och kunskapsprocesser i formandet av programmeringskunskap. De epistemologiska frågeställ-ningarna bildar det ramverk som omger mitt arbete. Teorier och tillvägagångssätt förankrade i genusforskningen utgör min metodologi. Programmeringskunskap i högskolemiljö är mitt tillämpningsområde.

[22]Trojer, et al: *Genusforskningens relevans* s. 10 ff.

3.3 Presentation av materialet

När jag har bearbetat materialet har jag utgått från de frågeställningar som finns presentade i bilagorna 2-4. Rubriksättningen och dispositionen i nedanstående text är dock en annan, men den ämnesmässiga anknytningen är bibehållen.

Utbildning och förhållande till matematikämnet

De intervjuades utbildningsbakgrund respektive yrkeserfarenhet varierar i stor utsträckning - vad beträffar såväl innehåll som omfång. Teknisk/ naturvetenskaplig gymnasiebakgrund finns representerad bland både män som kvinnor liksom samhällsvetenskaplig/ekonomisk. Förhållandet till matematik skiftar, och tycks inte ha något direkt samband med vilken typ av gymnasieutbildning studenten har. Tvärtom, som en student med teknisk bakgrund påpekar:

> *"Jag har aldrig gillat matte trots att jag klarat mig ganska bra ... .* ***- Har du funderat på varför?*** *- Tror det beror på sättet som det lärs ut ... jag tycker om att lösa problem, men inte att tvingas jobba med samma typ av uppgift flera gånger"*

En annan student med liknande gymnasiebakgrund bedömer matematik som "lätt". Det var dock först efter att ha genomgått ett test[23], som bekräftade bl a hans numeriska och logiska förmågor liksom goda matematikkunskaper, som han vågade satsa på en tekniskt inriktad utbildning som föregick den han går nu.

En tredje student hade efter långt studieuppehåll läst in nödvändiga matematikkunskaper på Komvux. För henne blev detta en kick eftersom det gick så oerhört bra och hon upplever matematikämnet som enormt roligt.

Upplevelsen av matematik som något oerhört roligt delas också av den student som har genomfört ett års studier vid en datautbildning där komplettering av matematikkunskaperna ingick[24]. Hon tycker dessa kunskaper har hjälpt henne när hon försökt lära sig funktionsorienterad programmering, i synnerhet kurserna i talrepresentation och algoritmteori som föregick programmeringskursen vid den föregående datautbildningen. Det hon uppfattar som lätt nu är att förstå de delar av syntaxen som har med variabeldeklaration och villkorsuttryck att göra. Samtidigt påpekar hon att hon nog inte hade förstått det heller om hon inte haft med sig de kunskaper hon har i programmering.

Bland de intervjuade lärarna är det en som lyfter fram den betydelse medhavda matematik-kunskaper (på högskolenivå) haft för henne då hon själv lärde sig programmera.

> *"För mig var det oerhört betydelsefullt för att jag skulle förstå regelverket…är inte säker på att jag skulle ha klarat programmeringen annars."*

Hon säger samtidigt att hon inte tror det är så för alla, men att det var så för henne.

Slutligen en student som hävdar att *"java är på så hög nivå att man slipper bry sig om de matematiska uträkningarna. Datorn gör det åt en."*

När man diskuterar matematikämnets betydelse för förvärvandet av programmeringskunskap är det viktigt att ta hänsyn till både vilken typ av matematik det gäller *("det är skillnad på*

[23] Testet genomfördes av arbetsmarknadsinstitutet.
[24] Matematikämnet kompletteras till gymnasiets E-nivå och därefter läses Diskret matematik på högskolenivå.

sannolikhetslära och att arbeta med formler", som en student påpekar) och hur den enskilda matematikundervisningen sett ut. För den som konfronterats med algoritmteori, boolesk algebra, mm i sin tidigare matematikundervisning torde åtminstone delar av begreppsapparaten i programmering kännas bekant. Med 3-årig s k C-matematik som grund får man dock räkna med att även en del av dessa begrepp är nya. Vidare verkar det som att de som behärskar dessa kunskaper bland de intervjuade studenterna möjligen haft en fördel vad gäller den teoretiska förståelsen och i bästa fall också vid hanteringen av de specifika situationer som berörs. I övrigt tycks de inte ha några fördelar vad gäller förståelsen av hur man bygger upp ett program i sin helhet. Dessutom är det kanske så att sambanden måste synliggöras för att bli användbara. En av de studenter som hade läst programmering som tillval på gymnasiet påpekade att ingen någonsin förklarade några samband mellan det de hade lärt sig i matematiken och det de fick lära sig i programmeringen. Han insåg att sådana samband rimligen måste finnas, men kunde helt enkelt inte upptäcka dem på egen hand. Gymnasietidens programmering upplevde han trots detta som fullt förståelig och framförallt rolig. Problemen kom när han började högskolan.

Frågorna utbildningsansvariga bör ställa sig är kanske:

- Vad innebär det egentligen att inneha goda matematikkunskaper?
- Finns det något automatisk samband mellan bra betyg i matematik och väl utvecklad problemlösningsförmåga? Att döma av den citerade studenten i inledningen av detta avsnitt så är det ingen självklarhet att man får träna den förmågan under gymnasietidens matematikstudier.
- Är matematikkunskapernas relevans kanske kopplade till vilken typ av programmering det gäller (t ex funktionsorienterad respektive objektorienterad)?
- Är det den matematiska sidan av programmeringen som orsakar problem?

Matematikkunskapernas betydelse eller obetydelse för förvärvandet av programmerings-kunskap tål säkert att utredas vidare. Ska man göra det bör man dock sätta in frågan i ett större sammanhang som frågeställningarna ovan indikerar. För mitt arbete nöjer jag mig med att konstatera att bland de studenter jag har intervjuat går det inte att påvisa några samband mellan förkunskaper i matematik och lättheten eller svårigheten med programmering. Det är snarare så att svårigheterna tycks vara lika stora för dem som enbart har med sig de förkunskaper som krävs för att bli antagen som för dem som har betydligt mer matematikkunskaper med sig i bagaget.

Vad innebär det att kunna programmera … ur studentperspektiv?

Föreställningarna kring vad det innebär att programmera varierar liksom förväntningarna kring vad utbildningen och i synnerhet programmeringen ska leda till. Några hade enorma förväntningar på just programmeringskursen, andra såg den mer som ett nödvändigt steg på vägen. De allra flesta hoppades dock att utbildningen ska leda till ett intressant arbete framöver.

> *”Jag vill utveckla mig själv inom ett område som verkar intressant och där det finns hyfsat goda utsikter att få ett jobb.”*

> *”Det här är början på ett nytt liv…jag har alltid tyckt att det varit roligt med data. Jag hoppas få ett jobb som jag trivs med…det ska kännas roligt att gå till jobbet.”*

De som provat på programmering tidigare gav i regel uttryck för högre förväntningar än de som inte hade någon erfarenhet. Det fanns dock exempel också på motsatsen. Några av de studenter som programmerat mycket uttalade att de även ville hålla på med andra saker. Ett par av de studenter som helt saknade programmeringserfarenhet formulerade farhågor om att det skulle bli svårt. Detta har dock inte automatiskt inneburit att de tappat sugen eller haft planer på att ge upp programmerandet.

> *”Jag har aldrig känt att jag har tappat greppet helt. Även om det varit himla svårt så tycker jag ändå det har varit roligt. […]Visst önskar jag att kursen varit upplagd på ett annat sätt …hade säkert varit lättare då, […] men jag ser ändå fram emot nästa kurs (Datastrukturer och algoritmer).”*

Hur studenterna ser på innebörden av programmering varierar, men det finns ett tydligt samband mellan tidigare erfarenheter och vad man betraktar som kärnan i programmerandet liksom vad man förväntar sig av programmeringskursen.

> *”Att programmera innebär för mig att jag kan utveckla och skapa möjligheter för andra personer. Jag vill förenkla för användaren. […] Jag vill t ex lära mig jobba med interaktivitet…lära mig tekniken.”*
>
> *”Programmering innebär skapande, men kan även vara ’bara’ problem-lösning…det är ett sätt att uttrycka sig kreativt.”*
>
> *”Jag har egentligen ingen konkret bild av vad det innebär att programmera och vad man ska ha lärt sig. När andra frågar tycker jag det är svårt att förklara.”*
>
> *”Jag har prövat lära mig programmera en gång tidigare och fick då en bild av programmering som något statiskt och okreativt. […] Jag skulle vilja få lov att experimentera och leka lite mer.”*

De två första citaten ger exempel på studenter som har arbetat ganska mycket med datorer och som har tidigare erfarenheter av programmering. Det tredje citatet utgör exempel på en nyfiken, men ganska osäker, nybörjare som saknar referensramar. Slutligen ett exempel på en student med negativa erfarenheter och förhoppningar om att få anledning att revidera sin negativa bild.

Det var få av de intervjuade som angav att programmering främst handlar om problemlösning. De flesta angav kommunikation med datorn, skrivandet av kod e t c som utmärkande drag för vad det innebär att programmera. Förväntningarna på vad man ska ha lärt sig efter den första introduktionskursen varierade också. De flesta påpekade att de inte hade någon som helst bild av vad som betraktas som det officiella målet med kursen. Många uttryckte besvikelse över att de inte lärt sig mer.

> *”Innan jag började kändes det som att man skulle behärska programmering …eller i alla fall få en ordentlig grund att utgå ifrån. Som det är nu känns det mer som att man ska få en enkel överblick.”*

I de officiella målbeskrivningarna framgår att målet med respektive programmeringskurs är att ge de studerande:[25]

- grundläggande kunskaper om programmeringsspråket java

[25] Beskrivningen från ddv 115:s målbeskrivning av de moment som relaterar till javaprogrammering. Dpt 100 har samma målbeskrivning, men med tillägget att de studerande också ska lära sig projekthantering.

- grundläggande kunskaper om objektorienterad programmering
- förståelse för programmeringscykeln (editering, kompilering, länkning, exekvering, interpretering)
- förståelse för programutvecklingscykeln (specificering, implementering, testning, felsökning)
- orientering om relationen mellan java och www
- orientering om principer för dokumentering av program

Vad grundläggande kunskaper respektive orientering innebär framgår inte närmare än att de moment som ska ingå specificeras, t ex objekt, klasser, arv, variabler, standardbibliotek (API:et) m m.

Vad innebär det att kunna programmera … ur lärarperspektiv?

De lärare jag har intervjuat framhöll i de flesta fall problemlösningen som det mest kännetecknande för programmering och betonade vikten av att förmedla denna syn till studenterna.

> *"Det som utmärker en god programmerare är förmågan att se hur man kan lösa problemet…kunna bena ut vad som behöver göras - bryta ner problemet och sen pussla ihop lösningarna."*

Det var dock ett par av lärarna som betonade andra faktorer som de mest utmärkande för programmerandet, t ex vikten av att lära sig ta reda på den information man behöver för att lösa ett problem respektive programmeringen som en datororienterad aktivitet.

> *"Att programmera innebär att kunna ge instruktioner till datorn. Man måste känna till hur det fungerar, vad som är möjligt e t c. ...Jag tror det är viktigt att ägna mycket av sin fritid åt programmering om man ska bli duktig. Man måste hålla på med programmering hela tiden - annars glömmer man."*

Synen på programmering som en datororienterad aktivitet framkom också hos många av studenterna och det är säkert inte utan betydelse att den lärare som står bakom citatet ovan befinner sig i den dubbla rollen av att vara både lärare och student. Samtidigt behöver inte det faktum att man inte nämner problemlösning som aktivitet innebära att man inte ser den delen av programmerandet. Det kan lika gärna handla om att man tar den som så självklar att man glömmer bort att nämna den. Enligt de intervjuade lärarna är det dock oftast just problemlösandet som orsakar problem bland nybörjarstudenterna.

> *"Många har problem just med problemlösningen, att bryta ned problemet i mindre bitar. De får helt enkelt ingen bild av hur saker och ting hänger ihop.[...] jag försöker lyfta fram det arbetssättet eftersom jag har upptäckt att det inte är något som är naturligt för alla."*

Syftet med programmeringskursen formulerades något olika av lärarna för de olika programmen. De ansvariga vid MT-programmet menade att det viktigaste är att skapa motivation hos studenterna att lära sig själva. Målet är att väcka intresse för programmering.

> *"Målet är att studenterna ska kunna ta reda på det man behöver veta för att göra det man ska. De ska veta var man ska leta, t ex genom att lära sig använda API:et. […]Vi har medvetet försökt lämna programmeringstekniska begrepp utanför - tanken är att studenterna ska lära sig göra saker och sen lära sig vad de heter. Vi har försökt få kursen att kretsa kring praktiken - och därför velat lämna det teoretiska utanför."*

Lärarna vid IT-programmet betonade betydelsen av att studenterna lär sig "tänkesättet"[26] kring programmering i allmänhet och objektorienterad programmering i synnerhet. Vikten av att de får en överblick över syntaxen lyftes också fram.

> *"Efter en termin bör man ha fått en översikt av ett programmeringsspråk och man bör ha lärt sig tänka i termer av program, i logiska sekvenser samt hur man plockar ihop de här delarna till ett program."*

Samtidigt påpekade ett par av dem svårigheten med att lära sig dessa två delar - tänkesätt (problemlösning) och syntax - samtidigt.

> *"Med den begränsade tid som är, så är det alltid väldigt svårt att hinna med båda delarna varför det gärna blir övervikt åt ena hållet."*

> *"Jag tror egentligen man skulle fokusera på problemlösningen först, men det är svårt att lära sig det parallellt med syntaxen - blir väldigt mycket att ta till sig. Samtidigt är tänkesättet det som är det svåra. Koden i sig är bara ett språk som man lär sig utantill."*

Den av de intervjuade lärarna med längst yrkeserfarenhet påpekade att alla delarna behövs och att det är bra om man kan lära sig dem parallellt för att därigenom skapa en förståelse för hur de hänger ihop. Problemlösningen är grunden som allting utgår från, men för att skapa ett program som löser problemet måste man också förstå syntaxen. För att förstå är det viktigt att skapa utrymme för någon form av reflektion, t ex genom att diskutera olika lösningar i seminariegrupper eller genom att skriva rapporter där man förklarar vad man har gjort och varför.

Vilka bilder har studenterna av lärarnas syn på programmering?

Studenternas syn på hur lärarna ser på programmering ser olika ut vid de två programmen. Vid **IT-programmet** var det många som tyckte det var svårt att uttala sig om vilken syn lärarna har , bl a med hänvisning till att man inte träffat dem så ofta[27]. Bland dem som hade någon uppfattning återkom uttryck som "viktigt", "tidskrävande", men också "naturligt".

> *"Deras syn på programmering är att det är något viktigt…något som man måste gå in för att lära sig."*

[26] En av lärarna definierar tänkesättet som bestående av tre grundkomponenter - sekvens, iteration och selektion - som man måste lära sig hantera.
[27] Det var heller ingen som hade skapat sig en radikalt annorlunda bild vid den uppföljande intervjun.

"De tar det naturligt…ger en känsla av att programmering är något som vem som helst kan lära sig som har intresset."

Vid **MT-programmet** återkom många till det stora intresset för programmering som de uppfattat hos de kursansvariga lärarna, i synnerhet intresset för spelprogrammering.

"De är väldigt spelfixerade, men bra killar …verkar själva tycka att det är kul…Kan känna att de saknar distans till programmeringen."

"Det märks att de kan mycket och är väldigt intresserade. Java är nog nåt´ man ska skriva bibeln med om de fick bestämma(skratt)."

Det intressanta att fundera över är hur studenterna påverkas av de föreställningar kring programmering som lärarna förmedlar (eller som studenterna uppfattar att de förmedlar). Blir de inspirerade eller tappar de motivationen? Klart är att det har betydelse vilka signaler studenterna upplever att de får. Föreställningar om programmering som något viktigt och tidskrävande, men också naturligt, har sannolikt haft betydelse för att de flesta av de intervjuade IT-studenterna ägnat mycket tid åt programmeringskursen. Men några tycks också ha blivit skrämda och fått uppfattningen att om man inte klarar den första programmeringskursen så klarar man inte resten av utbildningen. Det fanns även studenter som gav uttryck för en avsaknad av lekfullhet i lärandet - de uppfattade programmeringskursen som alltför resultatinriktad. Denna önskan att programmeringskunskap förmedlas på ett mer lekfullt sätt finns hos studenter vid båda programmen. MT-studenterna tycks dock tolka lärarnas signaler väldigt olika. En del tar relativt lätt på programmeringskursen och ser den heller inte som nödvändig för den fortsatta utbildningen. Andra tar däremot väldigt seriöst på programmeringen - antingen för att de faktiskt är eller har blivit väldigt intresserade[28]- eller för att de har en bild av att programmering är någonting man "bör" kunna.

Vad är det som är så svårt?

"Allting är svårt när man inte kan"

Citatet från en av de intervjuade studenterna ger en bra sammanfattning av den frustration många upplever i förvärvandet av ny kunskap. För att problemet ska tas på allvar är det dock viktigt att försöka gå lite mer på djupet kring vad det är som är så svårt och varför det är det. I den första intervjuomgången formulerade studenterna en mängd exempel på svårigheter. Så småningom blev det tydligt att vissa teman återkom, om än i lite olika skepnader:

- **Osäkerhet kring själva uppgiften:**

"Det svåraste med labbarna är att förstå vad slutresultatet ska bli…man har ingenting att relatera till - vet inte vad man ska sträva mot".

"När jag läser labbarna känner jag att jag inte vet vad jag ska göra."

"Det är svårt att hitta relevanta exempel i böckerna som relaterar till labbarna eftersom man inte vet vad man ska göra, vad man ska åstadkomma. Jag kan inte veta att på sidan 277 i boken finns ett relevant exempel."

[28] Detta var särskilt tydligt vid den uppföljande intervjun då programmeringskursens projekt var avslutade. Just dessa projektarbeten verkade ha fått igång en del studenter som inte verkat så motiverade i början av kursen.

- **Svårigheter att knyta ihop teori och praktik:**

  "Jag förstår när jag läser och på föreläsningar, men sen när jag ska skriva ner det vet jag inte hur. ... Föreläsningarna stämmer inte heller alltid med labbarna."

  "Vi går igenom en kod och sedan ska vi själva göra något annat…"

  "Det svåraste med första labben var att förstå allting…- Nu ska du deklarera attribut, nu ska du skapa ett objekt, men hur gör man? Jag förstod inte t ex vad en klass var, hur man skulle använda allting, hur man skulle sätta ihop det.[…] Jag tror den bristande introduktionen varit främsta orsaken att det kändes svårt, för när man väl hade gjort den insåg man att den nog inte var så svår egentligen."

  "Det är svårt koppla ihop bokens sätt att presentera java med hur handledarna presenterar ämnet. …Utförandet är svårt."

- **Svårigheter att hinna med och/ eller att greppa allting:**

  "Jag vet inte hur jag ska gå tillväga, var jag ska börja […] labbarna är stora och tempot alldeles för högt. […] Dessutom saknar jag möjligheter att fråga, det finns för lite tid till det.[…] jag måste få göra för att förstå… och få feedback."

  "Det känns som att försöka mura en vägg genom att mura in en bit här och en där. Till sist rasar hela muren eftersom det inte fanns någon grund … och man har kanske inte ens hunnit uppfatta att det var en mur det skulle bli.."

  "Det känns som vi har hoppat rakt in[…] I början var det t ex som att man glömde bort att inte alla jobbat i dos, kompilerat, o s v - allt sånt gicks igenom i ett himla tempo"

  "Det är svårt se logiken…och helheten. Objektorientering känns luddigt. […]Allting har blivit så resultatinriktat - man hinner inte reflektera (över det man gör)"

- **Svårigheter med syntaxen:**

  "Det är svårt att komma in i språket…allting blir för stort, jag förstår inte helheten…"

  "Svårast är att veta i vilken ordning allt kommer och hur man binder ihop allting."

  "Jag tycker det är svårt att skriva kod och att hitta fel i den egna koden."

- **Svårigheter att arbeta med färdig kod** (gäller enbart MT-programmet):

  "Det är för mycket färdig kod som det är nu…får dessutom ingen förklaring. Svårt att göra något när man inte förstår den färdiga koden."

  "Kul labbar, men för hög nivå på uppgifterna. Det har tagit 80-90% av tiden att förstå den färdiga koden. När man väl gjort det har det varit ganska lätt…"

  "Resultaten av labbarna har varit roliga, men det är svårt att förstå vad som händer, ...svårt veta var i koden man ska gå in."

- **Svårigheter att skapa utrymme för javakursen i relation till övriga kurser:**

> *"Det blir för långa hopp mellan gångerna…man hinner aldrig gå på djupet med labbarna."*
>
> *"Jag hinner inte lägga ner den tid jag skulle vilja p g a att företagsekonomin tar så mycket tid…jag hinner inte läsa på och fördjupa mig…"*
>
> *"Jag tycker inte om att läsa flera kurser parallellt - känner mig splittrad. Hade kanske varit lättare om det inte varit så många inlämningar hela tiden (i företagsekonomi). Nu känns det som man hela tiden jagar deadlines."*
>
> *"Javan är dåligt integrerad med övriga kurser, känns ibland som att oj, nu är det torsdag - bäst att göra lite java"*
>
> *"Det är okej att läsa två kurser parallellt, men ibland har det varit tre och nästan fyra eftersom en del kurser sträcker sig över hela terminen och några bara över halva. Det är svårt att prioritera när det blir så - lätt att javan hamnar i skymundan"*

Efter den första intervjuomgången var det tydligt att många av studenterna vid såväl IT-programmet som MT-programmet upplevde att de inte fick någon ordentlig grund att stå på. Många tyckte att uppgifterna var alldeles för stora och de upplevde att det var svårt att förstå inte bara hur man skulle göra utan även vad. Flera efterlyste mindre övningar som gick igenom ett moment i taget. Många upplevde också stora glapp mellan de teoretiska genomgångarna och laborationerna. Några få uttryckte svårigheter med specifika koncept som t ex arv. De som gjorde det hade i regel tidigare erfarenheter av programmering och brottades med att förstå skillnader mellan funktions- och/ eller procedurorienterade språk och objektorienterade som java. *"Det är svårt att se sammanhangen mellan det gamla och det nya"*, som en student uttryckte det.

De intervjuade lärarna bekräftade studenternas svårigheter som framkommer i citaten ovan. I synnerhet svårigheter med att knyta ihop teori och praktik liksom med att se mönstren i programmerandet eller komma på hur man ska "börja". Flera av lärarna uppfattade också att studenterna hade svårigheter med att förstå hur kommunikationen mellan klasser fungerar och därför t ex inte kunde förstå vitsen med att ha en separat testklass eller med att placera varje klass i en egen fil. Många av studenterna hade också påtalat detta och refererade då till kurslitteraturen där exemplen utgjordes av enstaka filer.

Vad var svårt i backspegeln?

Vid de uppföljande intervjuerna hade en viss förändring skett och de flesta ansåg att de (trots allt) hade lärt sig en del.

> *"Från att aldrig ha sett java så har jag ju lärt mig en del. Jag kan skriva en klass, göra en metod som anropar en annan klass o s v, men jag tycker inte jag kan säga att jag förstår riktigt."*

De svårigheter studenterna formulerade var nu mer specifika. Det var t ex många som relaterade till sitt användande av javas API.

> *"Jag har använt API:et, men jag förstår det inte riktigt. Jag vet inte vad man ska titta efter, hur man ska tänka och leta."*

Alla studenter hade använt API:et i någon utsträckning, men de flesta upplevde liknande svårigheter som den citerade studenten ovan. Felhantering var också något som flera av studenterna upplevde som något diffust. Vad gäller objektorientering som princip var det ungefär lika många som tyckte det kändes *"luddigt"* som tvärtom. Bland MT-studenterna relaterade flera sin förståelse till de extrainsatta föreläsningarna i början av hösten som behandlade objektorientering.

> *"Han drog igång tanken med det här objektorienterade och han gjorde det himla bra…men jag hade behövt lite mer av det."*

Det var betydligt fler av MT-studenterna än IT-studenterna som beskrev att de hade svårigheter med syntaxen vid den uppföljande intervjun. Möjliga förklaringar till detta kan dels vara laborationernas olikartade utformning. Många MT-studenter uttryckte svårigheter med att läsa färdig kod och såg sannolikt inte syntaxen för all kod. Dels spelade säkert tiden och kursupplägget in. IT-studenterna hade två kurser att koncentrera sig på, MT-studenterna tre. Dessutom fokuserade IT-programmets javakurs betydlig mer på att studenterna skulle lära sig terminologin, vilket sannolikt bidrog till en ökad förståelse även av syntaxen.

Bland dem jag intervjuat har de flesta blivit godkända i javakursen, men inte alla[29]. Det i sig är naturligtvis skäl till eftertanke om man betänker att alla arbetat med kursen på ett seriöst sätt. Samtliga som jag har intervjuat har lagt ner förhållandevis mycket arbete på kursen, även om variationerna naturligtvis varit stora. Vid IT-programmet har alla lagt ner 15-20+ timmar i veckan. Vid MT-programmet har studenterna vikt en dag per vecka till programmering. I samband med projektarbetet la några ner betydligt mer tid. Vad som är desto allvarligare är att även bland dem som blivit godkända uttryckte flera en stor osäkerhet när den uppföljande intervjun gjordes.

> *"Jag har svårt att se mönstret eftersom ingen har kunnat beskriva det på ett bra sätt."*

> *"Jag känner mig fortfarande väldigt osäker…tycker det är svårt att veta hur jag ska skapa ett program, hur jag ska tänka."*

Det var bara en av de studenter som inte hade någon programmeringserfarenhet sedan tidigare som upplevde att hon "kommit över tröskeln". Men det har inte skett utan ansträngning.

> *"Det där med att plugga 8 timmar det finns inte. Jag gör uppehåll på eftermiddagen när barnen kommer från skolan. Sen sitter jag på kvällen igen, och ofta på helgerna. Har sovit dåligt och känt mig konstant stressad. Men det är också krav jag lägger på mig själv. Jag ska klara av det, något annat existerar inte."*

[29] De ansvariga för MT-programmets programmeringskurs hade en uttalad målsättning att skapa förutsättningar för att alla skulle kunna bli godkända genom att utgå från vars och ens förutsättningar. Kriteriet för att bli godkänd var att studenten (ensam eller i samarbete med andra studenter) skulle ha gjort ett åtminstone delvis fungerande program under projektarbetet. Var och en skulle också kunna förklara hur de hade gjort och varför för att bli godkända. Vid IT-programmet fanns en uttalad målsättning att fler studenter än vid tidigare års programmeringskurser skulle ges förutsättningar för att bli godkända genom att skapa en mer genomtänkt struktur i kursupplägget. Att skapa en röd tråd genom alla laborationer var en del av denna strategi liksom att på ett tidigt stadium försöka fånga upp de studenter som man misstänkte hade problem med programmeringen.

Hur allvarligt studenterna ser på det faktum att de inte riktigt fått grepp om javan varierar - framförallt mellan programmen. IT-studenterna uttryckte att de fått uppfattningen att javakursen är oerhört viktig för den fortsatta utbildningen.

> *"Klarar du inte javan så klarar du inte resten av utbildningen har de sagt. Det gör ju att man känner sig lite stressad."*

Bland MT-studenterna var reaktionerna mer varierade. Några kände sig onekligen oerhört stressade av att inte ha "greppat" javan. Andra uttryckte att de visserligen hade hoppats att få lära sig mer, men att de ändå var rätt nöjda med det de gjort (e.g projekten).

> *"Jag är nöjd med programmeringskursen på så sätt att jag tycker jag förstår det jag har gjort. Jag har visserligen fått en del hjälp, men i stort sett har jag lärt mig själv."*

Båda programmen har en uppföljande javakurs - Datastrukturer och algoritmer. IT-studenterna måste dessutom använda java i ytterligare minst en kurs[30]. MT-studenterna kan däremot välja att inte läsa ytterligare programmering, vilket de intervjuade visade sig vara väl medvetna om.

> *"Java är nog inte riktigt min grej, tror inte det är något jag kommer att hålla på med i fortsättningen."*

Om påståendet ska ses som ett uttryck för att vederbörande har andra intressen eller om det snarare handlar om en försvarsmekanism för att hantera en besvikelse över att inte ha "lyckats", kan naturligtvis diskuteras. Kanske är det ett uttryck för båda delarna.

Varför är det så svårt?

Vid båda programmen angav studenterna, vad de uppfattat som, bristande introduktion som ett viktigt skäl till att det varit svårt. Det de saknat är framförallt relevanta kodexempel för olika typer av problem. I synnerhet har många saknat exempel som relaterar till laborationerna. Många har också upplevt att laborationerna varit alldeles för stora och omfattande.

> *"Det hade varit bra med många små övningar som byggdes på stegvis och som löpte parallellt med föreläsningarna.[…] Som det är nu tvingas man ta jättekliv…"*

Små övningar som på ett tydligt sätt illustrerar sambandet mellan input och output efterlystes av de flesta. Samtidigt menade många att laborationerna i sig säkert var bra, men att de borde introduceras på ett något senare stadium av kursen, efter att grundkoncepten bearbetats ordentligt.

> *”Jag måste få känna att jag får ordentlig kläm på tänkesättet innan jag går vidare. Jag skickar in något och får ut någon form av resultat.[…] Jag måste få skriva mycket kod själv, små enkla program som ger en överblick över hur de olika delarna hänger ihop.”*

[30] De studenter som läser två år har en databaskurs där java ingår. De som läser tre år kommer att läsa denna plus ytterligare minst en kurs där java ingår som en väsentlig beståndsdel.

Vid den uppföljande intervjun ombads studenterna att försöka beskriva hur de skulle vilja lära sig programmera. Mindre steg med ökande svårighetsgrad, utgå från grunden, grundliga förklaringar med många konkreta exempel, många tillfällen till övning, bättre feedback i form av handledning och utvecklingssamtal och mer tid (större koncentration) var några av de aspekter som lyftes fram. En av studenterna jämförde med hur hon lärde sig att sy.

> *"Jag har gått från att fålla en handduk till att sy en brudkänning. Har lärt mig stegvis och systematiskt […] och i min egen takt. Jag har tyckt det varit roligt också…"*

Vid **IT-programmet** har många påpekat hur en exempelklass med titeln Person fick polletten att trilla ned när de skulle lösa första laborationen. De flesta menade dock att de kunde fått den på ett tidigare stadium, eftersom inga andra exempel fanns att tillgå (t ex i kursboken). Om de två första laborationerna upplevdes som omfattande så tyckte många att framförallt den tredje, men även den fjärde var gigantiska.

> *"Stegen mellan labbarna (2 och 3 respektive 3 och 4) har varit enormt stora. ...Flera saker som var nya ...flera nya metoder som vi inte gått igenom. Man fick ingen ledtråd heller hur man skulle göra."*

Vid IT-programmet har många också uttryckt frustration över bristande läsanvisningar. Många har påpekat att de som distansstudenter är hänvisade till att lära sig med hjälp av kurslitteraturen, men att det behövs en bättre struktur eftersom *"ingen kan läsa hela boken och minnas allt som står i den".* De flesta riktade sin kritik mot uppläggningen av kursen, inte mot de enskilda lärarna. Bland dem som haft möjlighet att delta vid föreläsnings- och/ eller laborationstillfällen har de flesta sagt att de varit väldigt nöjda med föreläsningarna och att handledarna varit till stor hjälp vid laborationerna. Det de saknat har varit samordning mellan den teoretiska och praktiska delen.

> *"Det skulle behövas någon som ansvarar för helheten. Kursansvarig ska vara någon som håller på med java och är direkt involverad i kursen."*

De som inte haft möjlighet att resa till Ronneby har gett uttryck för en större frustration än de som varit där. En del har störts av föreläsningsanteckningar som blivit försenade eller varit ofullständiga. Några uppfattade att handledarna gjort olika bedömningar av inlämnade laborationer. Många uttalade svårigheter med att formulera frågor i det virtuella klassrummet:

> *"Jag tycker det är svårt att formulera frågor i (det virtuella) klassrummet. Frågorna omfattar oftast så mycket…och så kommer man på tusen följdfrågor."*

Det var också flera som efter genomgången kurs uttryckte tvivel över det lämpliga i att bedriva programmeringskursen som en distanskurs.

> *"Tänk om man skulle försöka lära sig köra bil på distans, är inte det samma sak egentligen?"*

De inblandade lärarna - i synnerhet labbhandledarna - ansåg att kursintroduktionen inte blev vad den borde. De menade att den första laborationen kom alldeles för tidigt. Studenterna fick ingen möjlighet att träna på mindre exempel, trots att denna kritik har varit återkommande för IT-programmets javakurs. Skälen handledarna framförde var framförallt avsaknaden av någon med en samordnande funktion, d v s en kursansvarig som är direkt involverad i kursens uppläggning. Detta medförde bl a att ingen fanns som kunde ta på sig rollen av att i god tid tänka igenom kursupplägget. När handledarna kopplades in fanns inget tidsmässigt utrymme

för att göra mer än vad som var absolut nödvändigt för att kursen skulle gå att genomföra. Ett par av handledarna uttalade också att de upplevde vissa hinder i kontakterna med den externa föreläsaren. Framförallt uttryckte de svårigheter med att komma överens om former för hur samordningen av föreläsningar och laborationer skulle ske.

En av handledarna vid IT-programmet hade erfarenheter av både distansundervisning och undervisning "på plats". Hon påpekade att hon upplevde det som betydligt enklare att uppfatta vilka problem de studenter hade som hon träffade regelbundet jämfört med dem hon mestadels hade kontakt med via e-post. För den senare kategorin var hon ofta tvungen att "gissa sig till" problemen, vilket kunde bidra till att dessa studenter i praktiken gavs sämre förutsättningar.

Vid **MT-programmet** riktades mer kritik mot de enskilda lärarna. Både i intervjuerna och vid utvärderingen efter avslutad kurs framkom kritik som framförallt berörde den pedagogiska förmågan. Samtidigt framhöll de flesta att de upplevde att de kursansvariga lärarna hade gedigna kunskaper, men att de hade svårt att förmedla dem.

> *"Det är så självklart för dem…de kan det så bra att de har svårt att förklara."*

Bilden har dock inte varit helt entydig. Många har uttryckt att de uppfattat att de ansvariga lärarna verkligen ansträngt sig och att de visat en stor öppenhet för att förändra saker till det bättre. De studenter som varit mycket intresserade av datorspel har också uppskattat det enorma spelengagemang som lärarna visat.

> *"Det märks verkligen att de tycker det är kul, och att de hållit på mycket själva."*

De studenter som inte varit lika spelintresserade har snarare uppfattat dessa egenskaper som negativa.

> *"De är oerhört spelfixerade, de verkar bara tycka om spel."*

Flera av dem som uttryckt ett stort missnöje över javakursen har framhållit den lärare som på begäran gjorde några enstaka inhopp som ett föredöme.

> *"Han förmedlar budskapet att alla kan förstå. Han kan förmedla samma sak på flera sätt…märker han att någon inte förstår, tar han om det- men från en annan vinkel […] Han gör så att alla blir delaktiga."*

Vid den uppföljande intervjun fanns fortfarande ett missnöje hos många MT-studenter över att inte fått en ordentlig grund att stå på. Samtidigt var det flera som upplevde att kursen blev bättre när de började arbeta med projekten.

> *"De (lärarna) fungerar betydligt bättre som handledare…när man frågar om nåt´ som har med det egna projektet att göra kan de alltid svara […] och de är oerhört tillmötesgående".*

Just möjligheten till individuell handledning var det många MT-studenter som efterlyste vid den första intervjun, men då fanns inte någon sådan inplanerad även om en del improviserade lösningar gjordes. Bl a engagerades några av de studenter som läst java tidigare för att hålla extralektioner och de hjälpte även sina klasskamrater med individuella frågor.

Orsakerna till att javakursen upplevts som svår av många får även andra förklaringar än dem ovan. Många av MT-studenterna har t e x upplevt det som positivt att läsa java under en längre period, men negativt att lektionerna varit så utspridda.

> *"Javan har blivit för utsträckt med bara en träff i veckan. De andra kurserna tar över och varje gång man sätter sig med javan måste man ladda om - vilket tar tid."*

Den stora spridningen i kunskapsnivå framhålls också som en möjlig förklaring, liksom att det är mycket upp till studenterna själva att lära sig.

> *"Det är ett problem att de som är duktiga tycker att det är för 'lätt'. Det gör att man känner sig nedtryckt när man inte kan."*

> *"Det är för mycket distanskurs över det hela - man hänvisas hela tiden till andra elever. 'Han kan…' ...för det är oftast en han."*

Den senare kommentaren relaterar till programmet i sin helhet. Studenten i fråga var mycket frustrerad över allt hon förväntades antingen känna till sen innan eller plocka upp på kort tid - alltifrån användningen av specifik programvara till färdigheter i programmering.

Alla som jag intervjuat vid MT-programmet har uttalat sig positivt kring att få arbeta med projekt i programmeringskursen så som man gjorde under andra halvan av terminen. De svårigheter studenterna haft kring detta har mer handlat om att komma på vad man kan göra som är på en rimlig nivå. Det gällde att komma på realistiska projekt både utifrån den tid man haft till sitt förfogande respektive i förhållande till den egna kunskapsnivån. Många har saknat vägledning just vad gäller att hitta en rimlig begränsning av den egna uppgiften.

De kursansvariga för MT-programmets javakurs har uppfattat att många av studenterna haft svårigheter med att läsa den färdiga koden. Vid intervjun påpekade de dock att en del av studenterna uttalat sig positivt kring att arbeta med färdig kod. De studenter som varit positiva har i regel haft viss programmeringserfarenhet sedan tidigare. De ansvariga lärarna har också medvetet arbetat i ett högt tempo. Skälen till detta har varit flera. Dels ville man låta studenterna komma igång med det praktiska så snart som möjligt. Den programmeringsundervisning som de kursansvariga lärarna själva genomgått fokuserade på den teoretiska sidan av programmeringen hela första månaden, vilket de uppfattade som "oerhört tråkigt". Dels var avsikten att genom att ha en snabb introduktion skulle man skapa utrymme för en mer diversifierad fortsättning av kursen. De studenter som ansåg sig ha förvärvat nödvändiga grundkunskaper skulle kunna börja arbeta med projekt direkt efter introduktionen. De som behövde bygga på grundkunskaperna ytterligare skulle kunna fortsätta ägna tid åt detta och sedan avsluta med ett mindre projekt. De ansvariga lärarna uttalade också svårigheten med att finna en balanspunkt mellan *"chock och tydlighet i undervisningen"*. De påpekade också att de avsåg att ordentligt fundera igenom vad som fungerat och inte under höstterminen inför den uppföljande kursen i Datastrukturer och algoritmer.

Är det då ingenting som är "lätt"?

Vid den första intervjun var det i princip ingen av de studenter som saknade tidigare erfarenheter av programmering som kunde peka på något som de tyckte var "lätt" i programmeringen. Denna bild bekräftades också av de intervjuade lärarna.

> *"Allting är mer eller mindre svårt för dem som är nybörjare, eftersom det rör sig om en helt ny begreppsvärld."*

De som hade viss programmeringserfarenhet kunde dock lyfta fram vissa aspekter av programmeringen som de upplevde som "lätta". Grundläggande syntax - som att deklarera variabler, skriva enklare metoder, konstruera och förstå olika typer av styrstrukturer (if- satser respektive for- och while-loopar) - tycktes inte orsaka några större problem för dem med viss erfarenhet av programmering. Även nybörjarstudenterna upplevde dessa aspekter som relativt lättförståeliga när den uppföljande intervjun gjordes. Många av MT-programmets studenter upplevde också att "polletten trillade ner" i samband med att man gick igenom programmering av grafiska användargränssnitt. IT-studenterna hade större problem med detta, men flera studenter påpekade att de trots allt upplevde det som stimulerande att *"det händer lite på skärmen"*. IT-studenterna hade heller ingen genomgång av hur grafiska gränssnitt programmeras utan fick läsa sig till dessa kunskaper. De hade också kravet på sig att skapa ett gränssnitt som gick att använda till den (förhållandevis omfattande) laboration de hade gjort innan där in- och utmatning hanterades via dos.

Flera av de intervjuade lärarna lyfte fram fördelarna med att introducera grafiska användargränssnitt på ett relativt tidigt stadium i kursen.

> *"Gränssnittet är en bra början därför att det är så konkret. Det går att rita upp på tavlan, vilket gör att alla kan delta i diskussionen. […] Att börja med att titta på hur gränssnittet ska se ut kan vara ett bra sätt att skapa ramarna. Man får en diskussion kring vad man gör eller ska göra och varför."*

Lärarna påpekade också vikten av att ändå börja med att gå igenom grundkomponenterna för att skapa en förståelse för vad som händer. Exemplen bör heller inte vara för stora.

> *"Jag har precis gjort ett exempel med en termometer. Man kan börja göra den i dos och sen skapa ett enkelt användargränssnitt till…Man kan använda programmet till att diskutera vad som händer och hur allting hänger ihop. Som det är nu blir det ofta väldigt mycket på en gång när man börjar med grafiska gränssnitt…lyssnare och annat. Termometerexemplet går att leka lite med…"*

En annan aspekt av "lätthet" i programmerandet är förhållandet mellan typen av utbildning och innehållet i programmeringskursen. Ett par av lärarna påpekade att kraven på *"vad man ska klara av"* måste anpassas till typen av kurs (t ex om det är en grundkurs eller en påbyggnadskurs) och vilken betydelse kursen har för den övriga utbildningen. En synpunkt var att inte alla studenter nödvändigtvis behöver lära sig samma saker. Kunde man dessutom lyckas med att skapa exempel som anknyter till utbildningen i fråga och som av studenterna uppfattas som användbara, skulle detta sannolikt öka både motivationen och upplevelsen av *"att det kanske inte är så svårt"*.

Olika förhållningssätt i förvärvandet av programmeringskunskap

> *"Det verkar som att distansstudenterna läser mycket mer. De som läser på plats verkar läsa när de måste. Samtidigt märker jag att de som har läst i boken innan förstår mycket snabbare vad jag pratar om på lektioner och föreläsningar."*

Att distansstudenterna generellt sett läser mer än studenter som läser på plats är kanske inte så förvånande med tanke på att de är mer tvingade att lita till sin egen förmåga. De har inte samma möjligheter att fråga andra utan har att välja mellan renodlad "trial and error" eller en kombination av det och införskaffande av kunskaper via skriftliga källor. Samtidigt uttryckte flera av de studenter jag intervjuade, som använde läsning som huvudsaklig "metod", att de upplevde svårigheter med att översätta det de läser till praktik. De upplevde sig ofta ha en väldigt god teoretisk förståelse, men visste inte hur de skulle omsätta den teoretiska kunskapen i handling. Andra studenter uttryckte att läsningen fyllde en viktig funktion för att bekräfta det de tyckte sig ha förstått när de prövat lösa ett programmeringsproblem på egen hand.

"När jag prövat någonting vill jag gärna läsa i boken för att se om jag tänkt rätt."

En intressant kommentar jag fick från en student vid IT-programmet med tidigare programmeringserfarenhet var att han upplevde att kursens uppläggning bidrog till att skapa ett beteende där man "bara" testar sig fram. Hans uppfattning var att ett sådant förhållningssätt riskerade att minska förståelsen för programmering som en problemorienterad aktivitet. Det ideala fallet av programmering såg han som då man löste problemet på papper för att sedan skriva in det i datorn. På så sätt försäkrade man sig om att man faktiskt *"begrep programmet"*.

Huruvida man föredrar att arbeta ensam eller i grupp med programmeringen varierar. Distansstudenterna arbetade föga överraskande mer på egen hand, men det var långt ifrån alla som trivdes med det. En del har haft möjlighet att bilda lokala studiegrupper, andra har tagit tillvara träffarna i Ronneby respektive Höganäs för att diskutera med och fråga andra. En mindre andel utnyttjade det virtuella klassrummet för att ställa frågor. Många av MT-studenterna har jobbat parvis. De med programmeringserfarenhet har i regel arbetat på egen hand. Flera av dem med erfarenhet har dessutom fungerat som informella handledare åt andra studenter. Samtliga studenter vid de båda programmen uttryckte ett stort behov av att "själva förstå". Det var dock stor variation på hur man föredrog att skapa denna självförståelse. En del måste få sitta och fundera på egen hand innan de kunde möta andra i en diskussion. För andra var det precis tvärtom - de måste först få diskutera med andra innan de kunde sätta sig och arbeta på egen hand. Ytterligare andra behövde en ständig växelverkan mellan dessa två förhållningssätt. Bland flera av IT-studenterna fanns en uttalad frustration över att inte alltid ha möjlighet att välja arbetssätt p g a svårigheten att hitta samarbetspartners inom det geografiska närområdet. Vid båda programmen uttryckte de studenter som saknade programmeringserfarenhet vikten av att någon förklarar de grundläggande koncepten och visar på olika exempel innan man som student försätts i situationen av att pröva sig fram på egen hand.

"Det går inte att sätta sig ner och programmera om man inte har en aning om vad man kan göra och hur."

Generellt sett finns en uppfattning bland både män och kvinnor som jag intervjuat eller samtalat med, att kvinnor har ett större behov av att förstå helheten än män. Många upplevde att män i allmänhet har lättare att acceptera att *"så här är det",* medan kvinnorna i regel har ett större behov av att veta varför. Både lärarna och studenterna upplevde att kvinnorna frågar mer. Några menade dock att det var svårt att avgöra huruvida det berodde på åldern eller könet eftersom många av de kvinnliga studenterna dessutom var lite äldre än genomsnittet. Karaktären på frågorna uppfattas dessutom vara densamma oavsett kön. En av de kvinnliga handledarna hade lagt märke till att hon fick mer frågor av "bagatellartad" karaktär än sin manlige kollega. Om det berodde på att de hade olika sätt att besvara frågor eller om det

berodde på att studenterna vågade ställa ”bagatellartade” frågor till henne tyckte hon däremot var svårt att svara på.

En annan handledare påpekade att hon upplevt att de kvinnliga studenterna varit mer aktiva och ivriga att lära sig än de manliga. Trots det är de duktigaste programmerarna hon känner män. Hon har funderat mycket över varför det är så och fått en allt starkare känsla av att det till stor del har med självförtroendet att göra. *"Killarna verkar ofta mer övertygade om att de ska klara av det."* Flera av lärarna uppgav också att de upplevde att de kvinnliga studenterna i allmänhet förde anteckningar i större utsträckning än de manliga och att de läste mer innan de satte sig vid datorn. Det var också flera av lärarna som upplevde att det var fler kvinnliga än manliga studenter som gav upp programmeringen, men också att det var fler manliga studenter som ägnade en stor del av sin fritid åt programmering.

Vad gäller ålderns eventuella betydelse för förvärvandet av programmeringskunskap upplevde en del av lärarna att de studenter som hade några års arbetslivserfarenhet i regel var mer självgående än andra, i synnerhet i jämförelse med dem som kom direkt från gymnasiet. En av lärarna menade också att de yngsta studenterna ofta verkade ha det jobbigast. Något han trodde kunde bero på att de ofta har så mycket annat omkring sig och var ovana att fokusera på en eller ett fåtal saker. Detta var särskilt märkbart i början av utbildningen. En annan lärare påpekade att man heller inte fick underskatta den turbulens det innebär för många att börja vid högskolan - i form av acklimatisering till en ny social (och ibland också geografisk) miljö. Att detta kunde sätta sina spår under första terminens studier menade han inte var särskilt märkligt.

Den uppfattning jag fick efter genomförda intervjuer var att de studenter som varit ute en längre tid i arbetslivet ställde högre krav på utbildningen i sin helhet. Flera av dem såg utbildningen som sin chans att bryta med ett arbetsliv som de upplevde hade gått i stå. Dessa personer la också ner oerhört mycket tid på såväl javakursen som övriga kurser. De hade i regel familj, men trots att jag anade att flera av dem slet ganska hårt för att få dessa två delar av tillvaron att gå ihop, var det ingen som sa något om detta annat än som en vag antydan. Den enda student som öppet uttalade krav på att utbildningen måste ge utrymme även för en social tillvaro var relativt ung och utan barn, men med en stor familj i övrigt. *"Det är lite arbete det också."*

En av lärarna sammanfattade de olika förhållningssätt till programmeringskunskap han konfronterats med på följande sätt:

> *"Antingen tycker man det är kul och vill jobba jämt eller så tycker man det är tråkigt och gör bara det man måste för att klara labben"*

Han berättade hur en del studenter under hans egen studietid så gott som gav upp programmeringen och i princip kopierade kod från andra. Själv uppfattade han det som en stor frihet att kunna programmera. Han tyckte dock inte att det var viktigt att kunna allt utan att det är mer en fråga om att lära sig att ta reda på det man behöver veta för att lösa en bestämd uppgift. Det är också ett förhållningssätt han försöker förmedla till de studenter han träffar. Han påpekade också att han tror det är viktigt att se ett resultat av det man gör för att få upp motivationen om man aldrig hållit på med programmering tidigare.

En svårighet i mötet mellan nybörjare i programmering och mer erfarna lärare och studenter kan vara att de som hållit på ett tag faktiskt glömmer vad som var problematiskt när de själva lärde sig programmera:

"De som har programmerat mycket upplever det ofta som något lätt och därför tenderar de att förringa vilken tid det faktiskt tar.[…] Jag tror det kan vara en bidragande orsak till att många nybörjarstudenter försöker gripa sig an hela uppgiften med en gång. […] De tror de ska kunna smälla ihop en labb på en eftermiddag, trots att de aldrig hållit på med programmering tidigare…"

Tidigare datorvana och erfarenheter av programmering

Betydelsen av tidigare erfarenheter av datoranvändning respektive förkunskaper i programmering utgör en ganska komplex bild. Datorvana är ett vagt begrepp och jag försökte därför delvis definiera vad jag själv avser genom att specificera begreppet med frågor om vilken vana vederbörande hade av att arbeta i dos-miljö respektive installera program samt i vilken utsträckning han eller hon brukade lägga ner tid på att själv försöka lista ut den mest ultimata användningen med dessa program. Jag frågade även om i vilken utsträckning den intervjuade ägnade tid åt att spela datorspel.

Bland de studenter jag har pratat med var variationen stor vad gäller både inriktningen på den tidigare datoranvändningen och omfattningen av densamma. Den minsta gemensamma nämnaren utgjordes av erfarenheter av att arbeta i Windowsmiljö, med ordbehandling, skicka e-post och surfa på Internet. Bland de intervjuades klasskamrater fanns dock personer som saknade även en del av dessa erfarenheter när de började utbildningen och några av de intervjuade påpekade vid det första samtalet att deras tidigare datoranvändning inte varit särskilt omfattande. Cirka hälften av de intervjuade studenterna hade någon erfarenhet av dos innan de började utbildningen. Dessa personer hade antingen hållit på med spel i dos-miljö eller programmering. Några hade hållit på med både och. Bland dem som bara kommit i kontakt med dos i samband med att de *spelat* dataspel inskränkte sig dock erfarenheten till att som en student formulerade det *"ha upplevt dos"*. Mest tydligt var att det fanns en grupp manliga studenter som hade ganska mycket erfarenhet av att ha arbetat med mikrodatorer som Commodore-64 och/ eller Amiga. En av de kvinnliga studenterna har som barn spelat spel på Commodore-64, men påpekade att hon då bara *"lärde sig att trycka på den och den knappen"*.

Spridningen i kunskapsnivå beträffande vilka erfarenheter i datoranvändning och programmering studenterna hade med sig, var speciellt märkbar vid MT-programmet. Ungefär halva klassen visade sig ha någon form av programmeringserfarenhet sedan tidigare och ett tiotal av dessa hade erfarenhet av just java[31]. Detta skapar en stor spännvidd och kan i sig bidragit till svårigheter med att hitta rätt nivå för undervisningen. Vid utvärderingen av programmeringskursen framkom att ungefär halva klassen var relativt nöjd med kursen, medan andra halvan inte var det.

Den stora spridningen i kunskapsnivå påtalades av en del lärare. Det var dock ingen - varken bland studenterna eller lärarna - som ansåg att förkunskapskraven borde ändras. Däremot påpekade flera att de s k preparand-dagarna i början av terminen kunde tillvaratas bättre för att se till att alla åtminstone fick grundläggande kunskaper i filhantering, användning av dos och rent allmänt i att hantera Windows. Det var också uppenbart att en del av de problem som vissa studenter upplevde i början av utbildningen snabbt passerades. De med minst erfarenheter av datoranvändning var också de som tog de största sprången och upplevde en avsevärd förbättring efter den första terminen.

[31] Enligt vad som framkom vid en handuppräckning i samband med den muntliga utvärderingen.

Att försöka tillvarata mer erfarna studenters kunskaper sågs också som något positivt, men flera pekade på de svårigheter det innebär att hitta en balans mellan att låta dessa studenter hjälpa till och att få dem att känna att de ställs inför stimulerande utmaningar.

> *"Det är inte alltid roligt att 'bara' vara den som hjälper andra."*

En av lärarna menade att det i realiteten finns en hel del osynliga krav på att studenterna ska kunna vissa saker när de kommer till högskolan, t ex vad gäller datoranvändning. Han menade att dessa krav ibland skapades för att läraren var alltför ambitiös. *"Det är så mycket man vill hinna med."* En hög ambitionsnivå kunde bidra till att tempot drogs upp, vilket i praktiken försvårar för de mer ovana studenterna.

Under intervjuandets gång uppenbarade sig en tydlig tendens; intresset för att spela datorspel och erfarenheter av Commodore-64 och/ eller av att ha kopplat upp sig i nätverk för att spela datorspel i större eller mindre grupper, är en erfarenhet som delas av flera av de manliga studenterna och av deras manliga lärare, men i princip ingen av varken de kvinnliga studenterna eller de kvinnliga lärarna har dessa erfarenheter. Generellt sett hade de manliga studenterna och lärarna en helt annan erfarenhet av att *leka* med datorer än deras kvinnliga studiekamrater och kollegor. De kvinnliga studenter som hade en djupare erfarenhet av datoranvändning än den som utgör vad jag ovan benämner minsta gemensamma nämnare, hade i regel förvärvat denna genom sitt yrkesliv och/ eller tidigare utbildning. Den erfarenhet av datoranvändning som kännetecknades av att ha utvecklats i yrkeslivet var i flera fall inriktad på en utbredd vana vid att använda färdiga applikationer.

> *"Jag tror inte det finns nån´ (kille) i min generation, som inte har den bakgrunden (erfarenhet av Commodore-64) om de varit det minsta intresserade av datorer."* [32]

Citatet ovan antyder att när det gäller vissa typer av erfarenheter kanske det också är en generationsfråga, vilket bekräftas av de svar och kommentarer jag fått i intervjuerna. De studenter som avslutat gymnasiet under början av 80-talet saknar - oavsett om de intresserar sig för datorspel eller ej - i princip erfarenhet av Commodore och Amiga, vilket kan förklaras av att företaget som tillverkade dessa datorer gick i konkurs i början av 90-talet[33]. En del av dem som har fått utlopp för sitt intresse för datorspel genom att använda sig av dessa grand old men bland mikrodatorer, har även blivit intresserade av att försöka ändra dessa spel eller andra applikationer vilket fört in dem på programmering. Det är m a o inte ovanligt att dessa spelintresserade killar, som det i huvudsak rör sig om, också gjort sina första försök i programmering på ett ganska lekfullt sätt.

Vilken betydelse har tidigare erfarenheter av datorer och programmering?

En fråga jag ställt mig under arbetets gång är *vilken* betydelse de förkunskaper en del studenter burit med sig har haft för deras möjligheter att lära sig programmera i java? Är det så att Commodore-64 användarna har det betydligt lättare än andra? Innebär förkunskaper i någon form av programmering automatiskt att det är lättare att lära sig alla former av programmering? Det är frestande att svara ja på de två sista frågorna. Samtidigt har kanske

[32] Citat från manlig lärare.
[33] Enligt ett inslag i ungdomsprogrammet Cyber från mars 2001 håller dock dessa datorer på att få en renässans hos en yngre generation (pojkar) som använder dem just för att spela spel. Gamla exemplar plockas fram från förråden i de föräldrahem där de finns eller cirkulerar på begagnatmarkanden.

just de frågorna - och svaren de har genererat från olika studenter - mer än några andra, fått mig att inse betydelsen av sammanhanget. Flera av de studenter som har relativt lång erfarenhet av datorer, inklusive programmering, uttryckte stora svårigheter med att tillägna sig de nya kunskaperna.

> *"Gamla kunskaper (i Pascal och Basic) har legat i vägen… Jag har försökt bekämpa gamla kunskaper, men de har hela tiden funnits i bakhuvudet, t ex globala variabler som används i Basic, eller att skriva program i en enda fil. De gamla kunskaperna har inte varit till någon hjälp.* ***- Tror du att du hade varit hjälpt av att ha en lärare som haft kunskap i flera programmeringsspråk och som kunnat jämföra?*** *- Nej, det är jag själv som måste göra detta i mitt huvud. Lära av det gamla och få in det nya. […] Jag förstår när jag läser i boken, men sen när jag ska skriva program är det vanan som kommer fram. Jag har inte fått kunskaperna i java att sitta kvar i minnet."*

Den citerade studenten formulerar svårigheten med att bli av med gamla programmeringsvanor. Det är förvisso en svårighet som är av en helt annan karaktär än att inte alls förstå. Den visar dock på att tänkesätten mellan de procedur- och funktionsorienterade språken skiljer sig markant från de objektorienterade, vilket i sig kan orsaka svårigheter. Samtidigt uttrycker den här studenten att det är en process han själv måste gå igenom, d v s det är mer en fråga om tid. Min reflektion är att bristande tydlighet kring vad som kännetecknar just objektorienterad programmering också kan bidra till svårigheten. Om dessa kännetecken görs tydliga för studenten är det lättare för denne att själv upptäcka skillnaderna.

Det är dock inte alla som upplever de gamla kunskaperna som ett problem. En student upplevde att det var tack vare de gamla kunskaperna som hon över huvud taget förstod någonting alls. En annan påpekade hur han kände igen en laboration från när han lärde sig ett annat programmeringsspråk. Det innebar att han kände igen själva problemet och alltså i princip visste hur det skulle lösas. Det som återstod var att översätta problemlösningen till javakod, d v s att skapa en syntaktiskt korrekt kod. Denna del av programmeringen ska definitivt inte underskattas. Det är dock en väsentlig skillnad på att "bara" lösa denna del av programmeringen och på att först lista ut vad som är själva problemet och hur detta kan lösas för att först därefter kunna ta itu med syntaxen.

Bedömning av programmeringskunskap

Oavsett hur vi som studenter har tillägnat oss våra kunskaper i programmering kommer vi till en punkt där dessa kunskaper ska bedömas. De examinationsformer som användes vid de respektive programmen var av två olika slag. Vid IT-programmet utgjorde de inlämnade laborationerna tillsammans med en skriftlig tentamen bedömningsunderlag för betygssättning. Vid MT-programmet genomfördes en s k codereview av de genomförda projekten. Denna kodgenomgång utgjorde underlag för bedömning och kompletterades i tveksamma fall med en granskning av genomförda laborationer. Uppfattningarna om de olika examinationsformerna skiljer sig inte överraskande åt. Studenterna vid IT-programmet ifrågasatte inte i något fall själva idén med att använda laborationer som underlag för bedömning. Meningarna var mer delade vad gäller den skriftliga tentamen. En del såg det som den enda vettiga examinationsformen. Några uttryckte att tentamen fyllde en viktig funktion för att få dem att läsa igenom kurslitteraturen ordentligt samt för att arbeta med övningarna i självstudietestet. Andra menade att självstudietestet var tillräckligt som examinationsform eller möjligen kunde

kompletteras med någon form av hemtentamen. Någon form av muntlig redovisning och/ eller redovisning i datasal framhölls också som en möjlighet.

> *"Jag tycker att det viktigaste är väl egentligen att man ska klara av att programmera. Vissa begrepp ska man förstås också kunna …men att skriva kod på papper är det ingen som gör i verkligheten, man borde hellre ha en del där man fick sitta vid en dator och skriva… Kanske kunde man haft en muntlig del, t ex någon form av redovisning. Kan man förklara har man ju i regel lärt sig…"*

Vid MT-programmet verkade alla positiva till själva idén med att ha en kodgenomgång. Några framförde synpunkter på att den kanske kunde kompletteras med en skriftlig del. Hemtentamen och parskrivning framfördes också som möjliga alternativ. Flera var kritiska till att redovisningstiden var så kort och att en del distribuerade sina program alldeles för nära inpå redovisningstillfället. En av de studenter som hade erfarenhet av skriftlig tentamen i programmering tillhörde dem som var mest entusiastiska.

> ***"- Vad tycker du om examinationsformen?*** *- Den bästa jag haft! Skulle kanske varit lite tuffare så att lärarna kunde kolla att man verkligen förstått, men det hade krävt mer tid. Nu är det nog några som slunkit igenom. …Men det var himla skönt att slippa tenta …vem vill skriva en tenta i java?! Hur gör man förresten en tenta som är rättvis? Man kommer ju aldrig att jobba på det viset. Projektet med tillhörande code review är i alla fall ett närmande till hur det ser ut i arbetslivet där man måste kunna prestera något och motivera sina lösningar. Dessutom lär man sig himla mycket på att förklara vad man gjort och varför."*

En av nybörjarstudenterna som brottats med stora problem uttryckte hur redovisningen fick henne att upptäcka att hon faktiskt lärt sig mer än hon trodde.

> *"Först tänkte jag att det här kommer aldrig att gå - jag kan ju ingenting. Dessutom är det en sak att förstå sitt eget, men andras? …men det gick ju!"*

Bilden av examinationen som någon form av prestation varierade också bland studenterna. De flesta verkade anse att det viktigaste var att man lyckades skapa en grundförståelse och att denna kunde se något olika ut. Samtidigt fanns bland vissa av MT-studenterna en tydlig tendens till oro över att vissa "fuskade" eller blev godkända ”alldeles för lättvindligt”. Oron verkade i flera fall bottna i någon form av avundsjuka relaterad till det enorma arbete man själv upplevde sig ha lagt ner. Ett par studenter påpekade dock att problemet - som de såg det - var att ett ”allför lättvindligt godkännande” av studenter skulle slå tillbaka mot dem vid senare kurser. Denna oro tolkar jag som ett uttryck för ett ”traditionellt sätt” att se på undervisning i den bemärkelsen att läroprocessen tenderar att betraktas som instrumentell. Alla studenter förväntas lära sig samma saker i samma takt.

Att som lärare tvingas bedöma andra verkar inte heller helt problemfritt. I synnerhet inte för dem som befinner sig i den dubbla rollen av att vara både student och lärare. *"Man hamnar lätt i en pressad situation",* som en lärare uttryckte det. Svårigheterna förstärks också när förkunskaperna varierar mycket i gruppen. Dessutom påpekade ett par av lärarna att för att göra en helhetsbedömning krävs det att man som lärare hunnit lära känna studenterna någorlunda vilket är svårt om man inte träffar dem så ofta och grupperna dessutom är stora. Systemet med 5-poängskurser sågs inte heller som helt oproblematiskt *"eftersom det tar så olika lång tid för olika individer att lära sig."* En av lärarna vid IT-programmet menade att vid en nybörjarkurs i java var det inget onormalt om bara 6 av 60 studenter klarade första tentan. På min fråga om han inte upplevde det som problematiskt eftersom det i praktiken slog

ut många studenter (p g a att de riskerade komma ohjälpligt efter i studierna och i förlängningen få sina studiemedel indragna) svarade han ja, men att han heller inte såg någon riktig lösning på problemet.

"Jag tror det krävs radikala förändringar i undervisningen för att ändra på detta."

Hur ser lärarrollen ut?

Av de intervjuade lärarna har alla utom en lärt sig programmera java som studenter vid BTH. Av de fyra manliga lärarna hade tre erfarenhet av programmering när de började vid högskolan. Alla visade dock i intervjuerna insikt om att de flesta trots allt är nybörjare i programmering när de börjar vid högskolan och att utgångspunkten därför måste vara att förmedla grundläggande kunskaper. Av de lärare som varit direkt inblandade i javaundervisningen vid något av programmen hade tre någon form av erfarenhet av detta sen tidigare.

Lärarna vid **MT-programmet** rekryterades till arbetet p g a personliga kvalifikationer. De ansågs väl lämpade för uppdraget - att utforma en ny typ av laborationer. Som tidigare nämnts utökades uppdraget allteftersom, för att så småningom - med kort varsel - resultera i att de blev kursansvariga. Vid intervjuerna med dessa lärare framkom att omställningen blev plötslig och att det ibland har varit svårt att få tiden att räcka till för att genomföra uppdraget - i synnerhet som de haft sina egna studier att ta hänsyn till vid sidan av. Arbetet menar de kräver en heltidstjänst, som dock kan vara uppdelad på fler personer. Det skulle t ex möjliggöra skapandet av fler övningar till varje moment och ge mer utrymme för handledning och fler lektioner. För att hinna med att undervisa på önskat sätt - i mindre grupper med ca 15 studenter i varje - har de kursansvariga haft lektioner i fyra stycken tvåtimmarspass. Dessa lektioner har hållits på torsdagar och för de kursansvariga inneburit arbete från 8-17, med en timmes lunch. Därutöver har det inte funnits utrymme för några pauser vilket bidragit till att de känt sig ganska slutkörda efteråt. Samtidigt framkom att de var mycket nöjda med det förtroende programansvarige visade dem. Vid uppläggningen av kursen har de upplevt det som en stor fördel att ha så fria händer som de haft och att inte hela tiden behöva få sina idéer sanktionerade av andra. De har också påtalat fördelarna med att kontinuerligt kunna diskutera idéer såväl som problem med varandra. På min direkta fråga om de skulle uppskattat någon form av mentorskap, har de svarat ja. De menade också att det kunde vara positivt om alla lärare och handledare inom ett visst ämnesområde samlades då och då för gemensamma diskussioner. Detta förslag har även tagits upp av andra lärare och vid MDA-programmet har man börjat tillämpa det med alla som på något sätt är involverade i programmeringskurserna. Lärare och amanuenser har gemensam arbetslunch en dag per vecka och labbhandledarna deltar tillsammans med kursansvarig vid utformningen av laborationer. Skälen man framför är att man uppfattar det som betydelsefullt att amanuenserna känner sig delaktiga och att det skapas en atmosfär av att kurserna är alla inblandades angelägenhet - man ska kunna gå in och diskutera med vem som helst. I förhållandet mellan amanuenser och etablerade lärare betonas också vikten av det stöd de etablerade lärarna kan ge amanuenserna.

Att detta arbetssätt inte är förhärskande vid hela BTH framkom också i intervjuerna. Just olika typer av mentorskap, skapande av en tydligare pedagogisk linje i undervisningen och tillvaratagande av mer etablerade lärares erfarenheter efterlystes. Av dem som varit anställda som amanuenser upplevde alla att de i mångt och mycket kastades in i lärarrollen och att det medförde att de gjorde samma misstag som *"alla andra"*. Misstag som i viss utsträckning hade gått att undvika om introduktionen i lärarrollen varit en annan.

De inblandade lärarna vid **IT-programmets javakurs** var alla utom en anställda som amanuenser. Den som inte hade en amanuenstjänst var den externt rekryterade föreläsaren. Alla amanuenserna uttryckte frustration över att det inte fanns någon kursansvarig som var direkt inblandad i kursen och kunde ta ansvar för dess utformning. Helst såg man att kursansvarig och föreläsare hade varit samma person. Ett par av dem påpekade att de hade kunnat göra mer för utformningen av kursen om de blivit anställda i ett tidigare skede. Normalarbetstiden för en amanuens med studier vid sidan av är tio timmar per vecka. En av amanuenserna arbetade dock något mer och ansvarade förutom för handledning i Ronneby också för javaundervisningen i Höganäs. Alla amanuenserna har lärt sig javaprogrammering under sin utbildning vid BTH. En av dem hade läst lite Basic i samband med en tidigare utbildning och dessutom arbetat i många år som lärare i bl a matematik inom grundskolan. Alla såg det som en fördel att ha sina egna erfarenheter av att lära sig programmera i färskt minne. Ingen var speciellt nöjd med uppläggningen på den egna javakursen och såg det som ett viktigt mål att försöka skapa bättre förutsättningar för dessa studenter. För att åstadkomma detta ägnade man mycket tid åt att försöka fånga upp studenterna och deras frågor. Strategin att låta kursansvarig gå ut med ett personligt brev till alla som inte skickade in den första laborationen i tid var t ex ett led i dessa ansträngningar. Amanuenserna var fullt medvetna om bristerna i årets javakurs, bl a avsaknaden av en mjuk introduktion och den bristande samordningen mellan föreläsningar och laborationer. Avsaknaden av en tydligare koppling mellan javakursen och utbildningen i sin helhet påtalades också.

"Det är viktigt att det skapas en röd tråd även med andra kurser."

Amanuenserna ansåg t ex att det hade varit önskvärt att introducera en del av analys- och modelleringstänkandet redan i denna första kurs, för att understryka programmeringens problemlösande karaktär.

Att de inledande programmeringskurserna ges på distans för IT-programmet menade inblandade lärare kräver andra förutsättningar än för de kurser som ges "på plats". Bl a uttryckte ett par av dem nödvändigheten av att försöka hitta elektroniska interaktiva verktyg som kan användas i inlärningsprocessen. Självstudietestet som användes sågs som ett betydelsefullt steg på vägen även om ambitionen var att hitta ett svenskt verktyg. Önskemålet var att hitta verktyg som kunde användas även i själva programmeringsprocessen - med en mer välutvecklad feedback än de gängse utvecklingsverktygen. Dessa är i regel skapade för professionella användare, inte för nybörjare. Vikten av att inse att distansundervisning inte kan användas som ett sätt att spara in lärartimmar påtalades också. Snarare menade man att distansundervisning kräver fler lärartimmar eftersom studenterna är mer utlämnade än de som befinner sig på högskolan mer eller mindre dagligen. Nödvändigheten av att finna former för att underlätta för studenterna att bilda lokala studiegrupper framhölls också. Gruppen i Höganäs framställdes av alla som ett positivt exempel.

Pedagogiska förhållningssätt

Flera av de intervjuade lärarna framhöll fördelarna med att arbeta i par. Dels därför att de upplevde det som positivt att ha någon att diskutera och bolla idéer med, dels därför att de upplevde att de i regel hade olika sätt att beskriva saker på. Genom att arbeta i par kunde de lära sig av varandras förklaringssätt, men också lättare tillfredsställa studentgruppens behov av varierade förklaringsstilar. Nackdelen som någon påpekade var att det möjligen kunde upplevas som förvirrande för studenterna att få olika och kanske rentav motsägelsefulla svar.

En av de intervjuade lärarna framhöll särskilt vikten av att våga släppa taget i sin lärarroll. Han menade att ämnesmässiga kunskaper var en förutsättning för att undervisa, men också förmågan att lyssna in studenterna och att våga erkänna att man inte kan allt. Genom ett sådant förhållningssätt ansåg han att man bättre uppmuntrade studenternas förmåga att själva söka svar, men med läraren/ handledaren som stöd. Tyvärr menade han och även de övriga intervjuade att de som var nybörjare i undervisningssammanhang inte fick den pedagogiska och ämnesmässiga uppbackning som är nödvändig för att kunna växa in i ett sådant förhållningssätt.

Under ett av mina mer informella samtal diskuterade jag med civilingenjör Christina Björkman som är doktorand vid BTH, Institutionen för Arbetsvetenskap och Medieteknik – forskningsenheten för Teknovetenskapliga studier.[34] Björkman har lång erfarenhet av undervisning i datavetenskap, bl a programmering. Hennes erfarenheter är att det många gånger är svårt att diskutera pedagogik vad gäller programmering. De i huvudsak manliga kollegorna är mer intresserade av tekniken, t ex vilken kompilator som ska användas, än pedagogiken. Ett av skälen, menade hon, kan möjligen vara att det inte ställs några krav på pedagogisk kunskap för att bli lärare inom de datavetenskapliga ämnena utan enbart på ämnesmässig. Att det blivit så, tror Björkman, har att göra med att det är svårt att hitta lärare över huvud taget. Då prioriteras de ämnesmässiga kunskaperna framför de pedagogiska. Björkman tycker själv att den pedagogiska förmågan är "viktigare" än de ämnesmässiga kunskaperna. *"Dessa måste naturligtvis finnas, men de behöver inte vara 'fullständiga'"*. I ett annat sammanhang diskuterade vi hur mycket man behöver kunna (ämnesmässigt) för att klara av att undervisa. Björkman menade då att läraren bör ligga minst ett par nivåer över den nivå hon eller han ska undervisa i, d v s om kursen läraren ska undervisa i är en grundkurs så bör hon/ han ha kunskaper motsvarande åtminstone ett par fortsättningskurser i samma ämne. Ligger man bara någon nivå över har man förmodligen inte tillräckligt stort perspektiv för att förklara på ett bra sätt, tror Björkman.

Vad gäller att komma in i lärarrollen menar Christina Björkman att de mer erfarna lärarnas inställning är betydelsefull. Det de förmedlar till de som kommer nya, i form av vad som är viktigt, hur man kan gå tillväga o s v, har oerhört stor betydelse. Björkman tror det hade varit bra med någon form av mentorskap, men säger att det i stort sett inte existerar något sådant i de sammanhang där hon verkar. Det hon finner mest anmärkningsvärt är att A-nivå kurserna ofta anses som enklare att undervisa på än C- och D-nivå kurser. Hon menar att det borde vara tvärtom eftersom studenterna på högre nivå är mer studievana och självgående. Grupperna är dessutom i regel mer homogena (och mindre).

3.4 Sammanfattning

I detta kapitel har jag redogjort för de resultat det empiriska materialet bidragit med. Jag har även beskrivit på vilka sätt jag gått tillväga vid insamlandet av materialet. Resultaten från intervjuerna har jag presenterat i de sammanlagt fjorton underavsnitten. Med hjälp av de teman dessa bildar har jag försökt återge de synpunkter som framkom i intervjuerna och fört ett resonemang kring dem. Min ambition har varit att beskriva de olika aspekterna på ett sätt

[34] Björkman är verksam vid Uppsala universitet och har även haft uppdrag vid Mälardalens högskola. Förutom undervisningsverksamhet har Björkman även deltagit i utvecklingen av det sen hösten 2000 nystartade civilingenjörsprogrammet System i Teknik och Samhälle ("humanistingenjörsprogrammet") i Uppsala.

som tydliggör interaktionen mellan studenter och lärare. De mest centrala aspekterna av innehållet i intervjuerna beskrivs nedan.

- **Utbildningsbakgrund och förhållande till matematikämnet**

Utbildningsbakgrunden hos de intervjuade studenterna varierade stort. Det gick dock inte att urskilja några direkta samband mellan den formella utbildningsbakgrunden och eventuella svårigheter med programmeringskursen, inte ens vad beträffar matematikkunskaperna. Det verkade snarare som att alla hade lika stora svårigheter oavsett hur mycket matematik de hade läst.

- **Innebörder av programmeringskunskap - ur student- respektive lärarperspektiv.**

Hur innebörden av programmeringskunskap definieras har ett tydligt samband med hur studentens erfarenheter av programmering eller likartade verksamheter ser ut. Det är också skillnad på hur studenter och lärare beskriver programmering som aktivitet. Det är få av studenterna som framhåller programmerandets problemlösande karaktär, medan så gott som alla lärare gör det. Många av de intervjuade studenterna uttalade att de hade höga förväntningar inför utbildningen. Betydelsen av att lära sig programmera varierar däremot liksom uppfattningar om hur mycket eller vad man bör ha lärt sig. Några uttalade mycket höga förväntningar, medan andra inte tillskrev programmeringskursen större betydelse än andra kurser. Den största skillnaden märks mellan de båda programmen. IT-studenterna visade generellt sett en större oro än MT-studenterna för att inte ha lärt sig "tillräckligt". Orsakerna till detta återfinns sannolikt i på vilka sätt programmeringskursens betydelse betonats av respektive kursledning. Vid IT-programmet underströks den inledande programmeringskursens stora betydelse för den fortsatta utbildningen. Vid MT-programmet framhöll programledningen att kunskaperna i programmering (och även andra ämnen) måste betraktas i ett längre perspektiv eftersom läroprocessen ser så olika ut för olika individer. De kursansvariga för MT-programmets programmeringskurs försökte bidra till detta synsätt genom att lyfta fram att avsikten med programmeringsprojekten var att studenten(erna) skulle kunna utgå från sin egen nivå.

- **Svårigheter (såväl som "mindre" svårigheter) i förvärvandet av programmerings-kunskap och försök att förklara dessa.**

De svårigheter som artikulerades i intervjuerna handlade om allt från problem med att förstå själva uppgiften (laborationerna) till att översätta teori till praktik och inte minst om bristande helhetsförståelse. Många uttryckte svårigheter med att upptäcka mönstren i programmerandet, t ex vad gäller att förstå hur man "börjar" och varför man gör på det ena eller andra sättet. Vid den uppföljande intervjun var det fler studenter som formulerade mer specifika problem, t ex med att använda API:et. Däremot tycktes de flesta ha fått grepp om grundläggande syntax. Påfallande många uttalade att de fortfarande kände stor osäkerhet inför programmeringen och flera gav uttryck för farhågor om att de inte skulle klara av att lära sig programmera under utbildningen.

De svårigheter som uttalades av studenterna bekräftades i de flesta fall av lärarna. Lärarna instämde också i den kritik mot bristande introduktion som studenterna framförde. Framförallt insåg man betydelsen av att introduktionen inte gjordes med för stora steg och att många tillfällen till övning gavs. Orsakerna till att de ambitioner som funnits kring introduktionen inte infriats såg man i de flesta fall som organisatoriska. Bristande pedagogisk erfarenhet

påtalades som möjlig orsak av en del studenter, vilket lärarna delvis bekräftade genom att efterlysa bättre pedagogisk uppbackning.

- **Olika förhållningssätt i förvärvandet av programmeringskunskap.**

IT-studenterna verkar inte helt förvånande inhämta mer information från skriftliga källor än MT-studenterna. IT-studenterna arbetade också mer på egen hand, men mer p g a att omständigheterna tvingade dem till det än för att de egentligen önskade det. Många såväl studenter som lärare upplevde skillnader i hur kvinnor respektive män tog till sig undervisningen. Kvinnorna ansågs både av sig själva och av männen ha större behov av att förstå helheten. De tycktes också vara mer frågvisa än sina manliga studiekamrater. Samtidigt konstaterade flera av lärarna att de uppfattade att fler kvinnor än män gav upp programmeringen.

- **Tidigare erfarenheter av datorer och pr ogrammering.**

Variationen var stor vad beträffar tidigare erfarenheter. Minsta gemensamma nämnaren utgjordes av erfarenhet av att arbeta i Windowsmiljö och med ordbehandling, skicka e-post och surfa på Internet. Bland de manliga studenterna och lärarna fanns en relativt stor grupp vars datorerfarenhet präglades av "lekfullhet". Många hade erfarenheter av datorspel både hemma vid den egna datorn och i nätverksmiljö. Det var också betydligt fler manliga än kvinnliga studenter som hade tidigare erfarenhet av programmering. Dessa erfarenheter tycks dock inte alltid varit till hjälp för förvärvandet av de nya kunskaperna. I avsnitt 5.2 för jag en fördjupad diskussion om tidigare erfarenheters betydelser för studenterna vid de olika programmen under rubriken *Skapandet av kollektiv kunskap*.

- **Bedömningsprocessen.**

Examinationsformerna vid de två programmen var av två olika slag. Tentamen och genomförda laborationer utgjorde bedömningsunderlag för IT-studenterna. För MT-studenterna utgjorde ett genomfört projekt med tillhörande kodgenomgång examination. Meningarna kring att använda tentamen som examination gick brett isär. En del hade svårt att se andra alternativ medan andra såg det som en hopplös form för kunskapsutvärdering. MT-studenterna var överlag positiva till att arbeta med projekt och ansåg också att själva principen med att presentera dessa i en kodgenomgång var bra. Den kritik som uttalades rörde snarast detaljer kring utförandet.

Från lärarhåll uttrycktes framförallt svårigheter med att bedöma studenterna när förkunskaperna var så olika, men också med tanke på att olika studenter behöver olika lång tid för att lära sig. Något som flera av lärarna menade att inte nuvarande uppläggning av utbildningar tar någon större hänsyn till.

- **Lärarrollen och olika pedagogiska förhållningssätt.**

Lärarna vid **MT-programmet** innehade hela kursansvaret för programmeringskursen. Detta innebar ett stort arbete, men medförde också tillfredsställelse enligt dem själva. De har uppskattat att få ha fria händer i arbetet och upplevt ett stort förtroende från programansvarige. Detta förtroende har betydligt underlättat prövandet av nya idéer i programmeringskursen. Sannolikt har också programledningens syn på lärandet som en långsiktig process påverkat de ansvariga för programmeringskursen. De har dessutom hela

tiden haft klart för sig att de ska ansvara även för MT-programmets uppföljande programmeringskurs och att de därigenom får möjlighet att följa upp resultaten av den egna undervisningen.

Vid **IT-programmet** har en lärare haft ansvar för föreläsningar. Övriga har varit labbhandledare med ansvar för laborationernas utformning och rättning. De uttalade stor frustration över att kursansvarig inte var direkt inblandad i programmeringskursen utan enbart stod som formellt ansvarig. De menade att uppläggningen på kursen hade kunnat bli betydligt bättre om det funnits någon med ett helhetsansvar för kursens utformning. Avsaknaden av en pedagogisk linje uttrycktes som problematisk. Bristen på pedagogiska diskussioner vid utformningen av programmeringskurser också vid andra högskolor, bekräftades av en utomstående lärare med mångårig erfarenhet av undervisning i programmering.

Gemensamt för de intervjuade lärarna var att flera av dem uttalade fördelarna med att kunna bolla idéer med varandra. De flesta efterlyste också någon form av mentorskap i lärarrollen.

4. Analys

Arbetet med projektet har inneburit en ständig dialog mellan de fältstudier jag genomfört och den litteratur jag läst. Valet av litteratur var dock inte givet på förhand. Många gånger har de frågor som fältarbetet genererat styrt mitt sökande efter texter som kunde tänkas ge någon form av svar. De följande avsnitten är ett försök att återspegla denna fortgående dialog.

4.1 Synen på kunskap

Frågor och funderingar kring kunskapsbegreppet har utgjort viktiga förutsättningar för detta arbete. Redan vid valet av ämne brottades jag med spörsmål av epistemologisk karaktär. Det har därför varit väsentligt att försöka artikulera både mina egna utgångspunkter och de olika föreställningar och förhållningssätt kring skapandet av kunskap som jag uppfattat bland studenter respektive lärare såväl som i den omgivande utbildningsmiljön.

Kunskapsprocesser

Ett viktigt avstamp för mitt arbete fick jag genom *The New Production of Knowledge - The Dynamics of Science and Research in Contemporary Societies.* Michael Gibbons m fl diskuterar där kring den utveckling som innebär att skapandet av kunskap sker i en värld där vetenskap, teknik och näringsliv förs allt närmare varandra. Författarna menar att för att förstå denna process måste man öppna den "svarta låda" inom vilken kunskap inom de vetenskapliga och teknologiska områdena skapas. Det gäller att synliggöra den blandning av kodifierad (vilket jag tolkar som "mätbar") och tyst kunskap som kännetecknar alla former av vetande. Inte minst är detta viktigt för att kunna genomföra någon form av kvalitetskontroll inom det utbildningssystem som utger sig för att stödja livslångt lärande. Något som också innebär att sammansättningen av studenter blir alltmer differentierad, såväl åldersmässigt som bakgrundsmässigt. Kunskapens karaktär i en värld där lärande utgör en kontinuerlig process blir med nödvändighet mer provisorisk än tidigare. Detta understöds av den accelererande betoningen på forskning, vilken - enligt författarna - paradoxalt nog riskerar att underminera grundutbildningarna.

Utbildningarna vid BTH kännetecknas i hög grad av föränderlighet. Genom att högskolan har ambitionen att profilera sig inom IT, ett område vars betydelser och innehåll är under ständig omvandling, utsätter den sig för stora krav på att "hänga med". Forskningen är därför oerhört betydelsefull. Frågan är bara vad detta innebär för grundutbildningarna och framförallt för den inledande fasen av dessa. Finns det tendenser att kurserna på A och B-nivå betraktas som "enklare" och därför överlåts åt de lärare som har minst pedagogisk och ämnesmässig erfarenhet? Finns det därtill tendenser att de engagerade lärarna lämnas utan uppbackning av sina mer erfarna kollegor? Vad innebär det i såfall för studenternas del? I vilken utsträckning ger skolan studenterna verktyg att öppna "den svarta lådan"? I det empiriska materialet uppfattar jag alltför många signaler som tyder på att det är nödvändigt att ställa dessa frågor. När alltfler uttalar att de inte tycker sig ha fått en bra grund att utgå från, utan tvärtom känner stor osäkerhet inför den fortsatta utbildningen, finns det all anledning att bli uppmärksam.

Skapandet av kunskap förvandlas, enligt vad Gibbons m fl (1994) hävdar, i allt större omfattning till en socialt distribuerad process. Det innebär bl a att varken utbildningssystemet i sig eller de olika disciplinerna inom det kan bibehålla sin status som autonoma verksamheter. Tvärtom blir det allt viktigare att interagera med olika parter med ömsesidig respekt. BTH utgör här ett tydligt exempel genom sin medvetna strävan att skapa former för *"att aktivt delta i utvecklingen av regionens näringsliv och samhälle"* [35], innefattande bl a den offentliga verksamheten. Dessa samarbeten är viktiga för att ge studenterna verklighets-baserad träning, men också för att befästa BTH:s betydelse för den regionala utvecklingen.

Samtidigt anar jag åtminstone två faror i den ovan beskrivna processen. Den ena utgörs av att vissa utbildningar ges helt andra möjligheter än andra att delta i denna process. - Hur ska t ex IT-programmet, som är en delvis distansbaserad utbildning, agera för att inte hamna vid sidan av i denna utveckling? Den andra faran utgörs av externa intressenters krav på resultat. Om kraven på att studenternas projektarbeten ska resultera i för näringslivet användbara produkter ställs alltför höga, vilka konsekvenser får det för lärandet? Finns det inte en risk att de som redan ”kan”, lär sig mer, medan de som ”inte kan” får fortsätta leva med sin okunskap? Möjligheten att lära sig i sin egen takt ges kanske inget utrymme om kraven på, för näringslivet, användbara resultat ställs alltför höga. De som inte lyckades få kläm på programmeringen i de första kurserna kanske aldrig får chansen att träna upp sina kunskaper, eftersom kraven på resultat medför att det i praktiken inte finns tid till det.

Tvärvetenskapliga ansatser[36]

Under arbetets gång har jag funnit det alltmer befogat att använda konvexa glasögon i mitt sökande efter material. För att förstå variationen av svårigheter som uttalades i intervjuerna måste jag behålla ett brett perspektiv. Ett uteslutande av en eller flera aspekter hade inneburit att jag tappat något i den meningsskapande processen.

> *"The narrower the focus, the more information is hidden from view"*[37]

Att skapa mening är en process, inte ett slutmål. Ett sådant förhållningssätt skapar förutsättningar för en tvärvetenskaplig ansats. I *The New Production of Knowledge* uppmärksammas transdisciplinaritet (eller transvetenskap) som ett möjligt förhållningssätt. Författarna förklarar begreppet transdisciplinaritet genom att beskriva vad som karakteriserar det. De centrala dragen i deras beskrivning utgörs av att hänsyn tas till sammanhanget eller sammanhangen i vilka kunskapen produceras, att det rör sig om en dynamisk process där frågor och svar genereras i ett spiralliknande förhållande och att nya metoder och praktiker utvecklas. Denna framställning gav mig stöd för min uppfattning. När det gäller att söka svar på frågor kring förvärvandet och förmedlandet av programmeringskunskap, är ett resonemang kring de epistemologiska utgångspunkterna en framkomlig väg. För att få hjälp att hitta dessa utgångspunkter har jag sökt bistånd i den breda forskningsflora som genusforskningen representerar.

[35] BTH:s presentationssida Näringsliv och samhälle: http://www.hk-r.se/naringsliv/ (nerladdad 2001-05).
[36] Tvärvetenskap är egentligen ett av tre begrepp för att beskriva olika sätt att skapa samarbete mellan discipliner. De två andra är multivetenskap respektive transvetenskap. Skillnaderna i betydelse av de olika begreppen diskuteras i Lena Trojers manuskript från föredraget *Gender Research - an interdisciplinary challenge.*
[37] Kaschak, s 24.

Genusforskning inom det teknovetenskapliga området

Genusforskning i Sverige och internationellt domineras av två, delvis överlappande, inriktningar. Den ena inriktningen koncentrerar sig på studier kring kön respektive genus, köns- och maktrelationer. Inom den andra inriktningen är det de vetenskapliga kunskaps-processerna som studeras. Frågor kring bl a teori- och metodinnehåll problematiseras i ett försök att klarlägga hur och på vilka villkor kunskap skapas inom olika discipliner. Genusforskare inom det teknikvetenskapliga fältet har t ex arbetat med att försöka belysa vilka föreställningar som dominerar vetenskapssynen inom de naturvetenskapliga och tekniska forskningsområdena. De har bl a identifierat ett antal exempel på hur positivismen fortfarande utgör en outtalad hållning inom många av de vetenskapliga praktikerna[38]. Klassificeringar, standardiseringar och formaliseringar utgör en viktig bas för vetenskapande. Genusforskare försöker utveckla nya sätt att närma sig kärnan i kunskapsproduktionen genom att ifrågasätta hela eller delar av denna bas.[39]

Elisabeth Gulbrandsen skriver i sin licentiatsavhandling: *The Reality of our Fictions. Notes towards accountability in (techno)science* om nödvändigheten av att skapa nya utgångspunkter och sammanhang för kunskapsproduktion inom forskarsamhället. Hon formulerar möjliga förhållningssätt för hur forskare undviker att hamna i en position där de intar perspektivet av ett "allvetande och neutralt subjekt"[40] som påstår sig återge objektiva fakta. Hon menar att forskare måste våga ifrågasätta basen för kunskapsproduktionen inom det egna forskningsfältet genom att ställa frågor kring villkoren för kunskapens tillblivelse. Ett viktigt steg i denna process är att försöka synliggöra vems och vilken verklighet det är som skildras. Ett annat att försöka identifiera de krafter som driver forskningen - både hos en själv och hos utomstående. Det Gulbrandsen förespråkar är definitivt inte relativism som flyter ut i ett oändligt "å ena sidan, å andra sidan". Det hon rekommenderar, med stöd av framförallt Sandra Harding(1987), är erkännandet av en subjektiv hållning. Argumenten är att den som undersöker/ forskar kring något alltid gör det med färgade glasögon. De frågor som forskaren ställer och de eventuella slutsatser hon/ han drar har alla filtrerats genom den egna erfarenheten. Forskningen blir därmed per definition alltid subjektiv. Detta ses dock inte som något problem, men däremot som något man måste synliggöra och finna ett konstruktivt förhållningssätt till. Ett sådant förhållningssätt kan vara att hela tiden reflektera över sin egen roll i det man undersöker; att våga se sig själv som en del av problemet. Genom att göra sig själv delaktig i problemet kan man också bli en del av lösningarna, istället för att utgå från att man är den som ska komma med lösningar på andras problem. Donna Haraway (1991) benämner detta förhållningssätt som situerad kunskap. Tilläggas bör att den situerade kunskapen formas gemensamt av alla de entiteter som ingår i ett givet sammanhang. De enskilda studenter respektive lärare som jag intervjuat har alla sin situering, sina utgångspunkter, liksom andra inblandade aktörer. Situeringen är heller inte någonting statiskt utan något som ständigt formas och omformas, där det övergripande sammanhanget har en stor roll i hur detta sker. Det Haraway framhåller som väsentligt är att för att förstå dessa övergripande sammanhang måste man formulera sin egen utgångspunkt.

"The only way to find a larger vision is to be somewhere in particular."[41]

[38] Harding, Haraway, Gulbrandsen m fl.
[39] Mörtberg Christina: Teknikvetenskap och genusforskning eller Creating Better Worlds Through Imaginations. Utgör bilaga 5 i Trojer et al: *Genusforskningens relevans* - slutrapport från integreringsarbete i åtta svenska forskningsråd.
[40] Perspektivet benämns av forskare som Donna Haraway som "the God-trick" eller "the view from nowhere".
[41] Haraway, 1991, s 196.

Vilka är mina glasögon?

Hur tydliggör jag då min egen situering i det arbete som har resulterat i denna analys? En del av den har jag redan förklarat; jag är själv student, inskriven vid institutionen för programvaruteknik och datavetenskap och jag bär på delvis negativa erfarenheter vad gäller att lära mig programmera i högskolemiljö. Jag är dessutom kvinna, småbarnsförälder, äldre än den genomsnittlige studenten och med erfarenheter av yrkesliv och flera olika skolformer, inom olika discipliner med mig i bagaget. Dessa gamla och nya erfarenheter och min upplevelse av dem bär jag naturligtvis med mig när jag försöker tolka verkligheten - de utgör min situering. Denna situering har i sin tur avgörande betydelse för vilka ställningstaganden jag gör eller avstår från att göra. Mina drivkrafter för att sätta igång detta examensarbete utgår från ett ställningstagande. Jag ser det nämligen som ett betydande problem att det finns studenter som inte tillägnar sig relevanta kunskaper i programmering, trots att de lägger ner stora ansträngningar för att genomföra grundkurserna. Med relevanta kunskaper menar jag att vi som studenter inte enbart ska passera som godkända vid betygssättningen. Var och en ska även ha tillägnat sig en känsla av att kunna använda sig av det hon/han har lärt sig. Åtminstone bör alla ha fått förutsättningar att bygga vidare på sitt lärande inom utbildningssystemet eller i yrkeslivet. Om inte ens de förutsättningarna finns, om ett antal studenter upplever att de saknar en stabil "grund" att utgå från - då är detta ett allvarligt problem som måste åtgärdas.

I detta problem är jag en del. Jag har deltagit och deltar i den utbildningsmiljö som examinerar dessa frustrerade studenter. Jag kan välja att se problemet eller att bortse från det. Jag har valt att försöka se det. Samtidigt - genom att formulera en definition som den ovan riskerar jag att skapa dikotomisering, d v s tudelning. Antingen är det si *eller* så. Inom genusforskningen generellt ses dikotomisering som en belägenhet som bör undvikas, eftersom den bygger på föreställningar om att objektiv kunskap är möjlig. Genusforskningen betonar istället betydelsen av att öppet redovisa utgångspunkterna för den forskning man bedriver och insikter om att dessa utgångspunkter är avgörande för vilken kunskap som i slutänden produceras. Då det gäller att dra slutsatser, komma fram till resultat - d v s begrepp som har med skapandet av kunskap att göra - så handlar det hela tiden om att klargöra ur vems/ vilkas perspektiv detta görs. För att det över huvud taget ska bli möjligt att *välja* utgångspunkter måste först existerande hierarkier av vad som är viktigt/ sant raseras.

Perspektivet jag utgår från är min egen och andras upplevelse av att inte förstå. Det jag ser som en möjlig orsak till denna upplevelse är att utbildningsansvariga tenderar att förmedla bilden av att "förståelse" i programmering är liktydigt med att "bli godkänd på laborationer och tentamina". Eftersom behovet av att kunna "mäta" studenternas kunskaper ges så stort utrymme i undervisningen blir det litet eller inget utrymme över för den kunskap som inte går att mäta. Att jag kan lösa ett aldrig så litet programmeringsproblem, med en grundmurad - men inte nödvändigtvis artikulerad - förståelse för vad jag gör, betraktas som mindre viktigt än att jag gjort samma laboration som alla andra (utan att nödvändigtvis ha förstått den).

Konflikten med viktig/ synlig respektive oviktig/ osynlig kunskap härrör från en av positivismens mest centrala dikotomier - den mellan förnuft och känsla. Inom genusforskningen försöker man synliggöra hur den och andra dualistiska principer genomsyrar stora delar av i synnerhet det naturvetenskapliga och teknologiska forskningskomplexet och man försöker finna nya vägar i sökandet av kunskap. En sådan väg är att söka förhandling - att låta olika röster berätta för att därigenom ge utrymme för en förhandlingssituation. Orsaken är synbart enkel - handlingar utförs inte i ett vacuum. Det jag

gör eller inte gör påverkar alltid någon/ något, ofta på bekostnad av den/ det andra. Om programmeringskunskap ”förmedlas” med utgångspunkten att finna det ultimata sättet att lära sig på (d v s som om detta vore möjligt enligt principen om att kunskap är objektiv) så kommer alltid några att stängas ute. Om inte grundvalarna för varför man väljer att ”förmedla” och *mäta* kunskap på vissa sätt i ett givet sammanhang tydliggörs blir det omöjligt att förhandla. Förhandlingar kan bara genomföras om parterna upplever sig ha lika villkor - annars handlar det om att skapa hierarkier.[42]

Konserverande strukturer

En fråga som utbildningsansvariga bör ställa sig är i vilken utsträckning enskilda universitet och högskolor (inklusive BTH) diskuterar den övergripande kunskapssyn utifrån vilken utbildningarna formas. Finns det utrymme för olika synsätt kring vad kunskap är och hur den kan förmedlas? Vilken bild har man av de studenter som söker sig till högskolan? På vilket sätt resonerar man kring de studenter som får problem med studierna?

Minna Salminen - Karlsson skildrar i sin doktorsavhandling *Bringing Women into Computer Engineering* två utbildningar som initierats i syfte att rekrytera fler kvinnliga studenter till de datatekniska ingenjörsutbildningarna. Förutom att redogöra för den process som föregick lanseringen av de respektive utbildningarna ger hon också en uttömmande beskrivning och tolkning av de sammanhang inom vilka de nya utbildningarna skapats. Hon tecknar en bild av de datatekniska ingenjörsutbildningarna som visar på hur dessa präglas av omfattande prestationskrav. Prestationskraven kännetecknas av ett högt studietempo med litet eller inget utrymme för reflektion. Kurser tenteras av under vissa perioder, vilket kan innebära 2-3 tentamina inom loppet av en vecka. Dessutom eftersträvas en hög nivå på proven eftersom det anses som ett lika stort bekymmer om alltför många klarar av en tentamen som om alltför få gör det. Att bibehålla bilden av att det är "svårt" att tillägna sig kunskaper inom dessa utbildningar ligger högt på prioriteringsordningen. Många av lärarna utgår också indirekt från att studenterna har vissa förkunskaper som dock inte finns officiellt uttalade (t ex i form av antagningskrav)[43]. En annan aspekt Salminen-Karlsson visar på (och som bekräftades i mitt samtal med civilingenjör Christina Björkman) är den utbredda bristen på pedagogisk skolning bland dem som undervisar vid de datavetenskapliga utbildningarna. De flesta lärarna rekryteras bland f d studenter och anses tillräckligt kvalificerade för läraruppdraget om de har relevanta ämneskunskaper. Dessa lärare har lyckats genomföra sina egna studier tack vare en god anpassningsförmåga. Med avsaknad av andra erfarenheter av undervisning än sin egen faller de flesta tillbaka på denna och vidareförmedlar därmed de idéer om vad som är viktigt och hur detta bör förmedlas som de själva har upplevt. På så sätt bidrar de till att reproducera rådande värderingar. Det faktum att i synnerhet nybörjargrupperna är oerhört stora i kombination med att studenterna läser flera kurser parallellt och har fasta tentamensperioder, gör det dessutom svårt att i praktiken genomföra förändringar även för dem som har ambitionen.

[42] Hela detta stycke utgör min tolkning av genusforskningens centrala kritik såsom den beskrivs av forskare inom området, varav jag framförallt låtit mig inspireras av Alison, Gulbrandsen, Haraway, Harding, Kaschak, Mörtberg och Trojer.

[43] Christina Mörtberg beskriver också detta fenomen i sitt bidrag till utvärderingsrapporten *Datateknisk ingång för kvinnor*. Studenterna vid det utbildningsprogram Mörtberg m fl utvärderat vittnar om hur många kurser "börjar någonstans i mitten", d v s lärarna tycks utgå från att studenterna redan kan vissa saker.

Salminen - Karlsson gör en historisk tillbakablick som beskriver hur i synnerhet civilingenjörsutbildningarna länge omfattade individer som förutom att utbildas till duktiga civilingenjörer också utbildades till ledare. I utbildningen ingick lika mycket att skapa en social identitet som en yrkesidentitet - att arbeta som civilingenjör innebar att befinna sig högt upp i hierarkin på såväl den egna arbetsplatsen som i samhället i stort. Civilingenjörs-utbildningarna var, enligt denna beskrivning, förknippade med hög status och makt. Den näst intill totala dominansen av män bland såväl studenter som lärare, i förening med teknikutvecklingens starka kopplingar till den militära industrin bidrog också till skapandet av en kultur som präglades av hård konkurrens och strategiskt förnuftstänkande. Den höga arbetsbelastningen i kombination med ständig tidspress skulle dessutom sålla agnarna från vetet. Endast de som var starka och duktiga nog att klara detta ansågs mogna för de ledande positioner som väntade.

Dagens civilingenjörsutbildningar kännetecknas enligt avhandlingen fortfarande av ett högt tempo och en hård arbetsbelastning. De datatekniska ingenjörsutbildningarna verkar inom samma miljö som civilingenjörsutbildningarna och präglas därför av samma kultur. En av de föreställningar som omhuldas är att det ska vara lika *svårt* för alla. Genom att behandla alla studenter lika så anses de få samma förutsättningar att klara utbildningen. Likhetstanken innebär att alla studenter ska göra samma typ av uppgifter och ges lika mycket eller lite stöd i undervisningen. Salminen - Karlsson lyfter dock fram det som borde vara uppenbart, men inte är det, nämligen att för att alla ska ges samma förutsättningar krävs det att de har det från början. Att alla uppfyller antagningskraven innebär *inte* att de har samma förutsättningar. För att upptäcka detta måste man se längre än till vilken formell utbildningsbakgrund studenterna har. De kulturella och sociala faktorerna har en långt mer avgörande betydelse än vilka betyg studenterna förvärvat. Ett exempel utgörs av det faktum att männen i betydligt större utsträckning än kvinnorna har stor erfarenhet av datorer (och programmering). Något som påpekas även i andra undersökningar[44]. Dessutom har de redan då de inleder sin utbildning, till skillnad från kvinnorna, ofta ett nätverk av likasinnade att dela och diskutera sina intressen med. Till bilden hör också den kulturella kontext som generellt förväntar sig ett större socialt ansvar av den kvinnliga delen av befolkningen. Detta visar sig genom könsrelaterade attitydskillnader till hur kunskap (kan) förvärvas. Dessa skilda attityder tar sig uttryck i form av att betydligt fler manliga än kvinnliga studenter inte ser någon konflikt mellan att ägna timmar framför datorn bara för att det är "kul" och att klara av studierna. De ser ingen konflikt mellan den tid det tar att "leka" och kravet på ansvar. De kvinnliga studenterna däremot ställer generellt betydligt större krav på att kunskapen ska gå att använda, d v s det måste finnas ett nyttobetonat syfte med den. De strävar efter att effektivisera studierna för att också hinna med sina sociala åtaganden. Mot denna bakgrund blir det tydligt att kvinnorna alltför ofta är de som drar det kortaste strået. Tidspressen och prestationskraven stämmer å ena sidan väl överens med kravet på nytta och effektivitet. Å andra sidan kommer samma faktorer i obönhörlig konflikt med kravet på socialt ansvar. *"Dygnet har bara 24 timmar"*, för att citera en kvinnlig student.

Psykologen Ellyn Kaschak (1992) beskriver hur föreställningar om vad som är "korrekt" manligt respektive kvinnligt beteende sitter djupt rotade inom den västerländska kulturen. Den manliga normen utgörs enligt Kaschak av strävan efter oberoende ("En bra karl reder sig själv"). Den kvinnliga normen präglas av relationsskapande och omtanke. Kvinnlig identitet

[44] Se Björkman respektive Wistedt.

skapas utifrån de relationer kvinnor har. Samtidigt utgör den manliga normen också den mänskliga, d v s den definierar sanningar som anses giltiga för hela mänskligheten[45].

> *"[...] whoever gets to draw this line, to create the boundary, is the owner of the context and, as such, holds the power to define reality, to say what matters and what does not".*[46]

Dessa föreställningar menar Kaschak, medför att både män och kvinnor i betydligt högre utsträckning accepterar att män ägnar sig åt sina intressen än att kvinnor gör det. En kvinna däremot tillåts ägna sig åt sina intressen bara så länge de inte går ut över de närmaste relationerna (t ex man och barn). Kaschak menar att om man behandlar alla "lika" - utifrån föreställningen att alla har samma förutsättningar - negligerar man det faktum att vår (västerländska) kultur i grunden bygger på ett hierarkiskt tänkesätt där värderandet av vad som är sant/ falskt, bäst/ sämst, viktigt/ oviktigt utgör grundpelare. I själva verket innebär "likhetstanken" att man gynnar dem som redan är en del av den dominerande kulturen. De som inte är det måste antingen anpassa sig till den eller bli betraktade som misslyckade. Att inte lyckas blir därmed något som härrörs till individen, inte strukturen eller sammanhanget. "Det är jag som är så korkad att jag inte kan lära mig".

I vilken utsträckning har man inom de respektive institutionerna vid BTH artikulerat de värderingar utifrån vilka man skapat den egna strukturen? Varifrån kommer uppfattningar om att det går att sätta likhetstecken mellan kvalitet och "svårt" som uttrycks av såväl vissa studenter som lärare? Utgör högt studietempo, krav på mätbara resultat och enhetliga undervisningsformer i själva verket delar av en utbildningskultur som - åtminstone inom delar av BTH - anses så självklar att ingen kommer på tanken att den kan ifrågasättas?

Synen på teknik ur genusperspektiv.

Hur ser den övergripande synen på teknik ut som präglar utbildningarna vid BTH och andra högskolor och universitet? Vilka föreställningar finns kring hur vi formas av och formar den teknik som framställs? Att tekniken skulle vara autonom ifrågasätts av allt fler[47], samtidigt som det är tydligt att frågan i flera fall upplevs som komplex. Den grundläggande problematiken tycks i åtskilliga fall beröra förhållandet mellan teknik och kön och frågor kring vad som är essentiellt eller inte i skapandet av oss som könsvarelser. I denna diskussion utkristalliseras två primära förhållningssätt. Det ena utgörs av föreställningen att kvinnor och män har olika sätt att se på teknik, t ex vad gäller nyttobetonade aspekter, så som diskuteras ovan. Enligt denna uppfattning fascineras män generellt av tekniken för dess egen skull medan kvinnor anammar den teknik som de tycker sig ha användning för. Att det är så förklaras med kulturella och sociala orsaker, men det ses också som något som inte utan vidare låter sig förändras. Det andra förhållningssättet karakteriseras av inställningen att vårt förhållande till teknik förvisso är socialt konstruerat, men att det därmed också är något som

[45] Inom genusforskningen betonas att den manliga norm som här refereras till är den västerländska, vita och heterosexuella manliga normen.
[46] Kaschak, s 31.
[47] De artiklar som publicerats i *Från symaskin till cyborg,* utgör exempel på olika aspekter av detta ifrågasättande.

ständigt förändras (eller omförhandlas)[48]. De två perspektiven utesluter inte varandra, men kan definitivt få olika konsekvenser för vårt handlande.

Det första perspektivet är lätt att finna indicier för i den vardag som omger oss. Det är t ex betydligt lättare att finna exempel på pojkar och män som gillar att "mecka" - med mopeder, bilar, TV-apparater, datorer mm - än vad det är att hitta motsvarande exempel bland flickor och kvinnor. Samtidigt är det många gånger svårt att avgöra vad som är myt och vad som är verklighet i dessa exempel. Karin Mårdsjö och Pär Carlshamre visar i *Retoriken kring tekniken* hur vårt samhälle är fyllt av olika typer av retorik som används för att cementera såväl som omskapa synen på teknik. Ett exempel är hur marknadsförares syn på vad som är kvinnligt respektive manligt återspeglar såväl faktiska erfarenheter som mytbildningar. Hos de flesta marknadsförare finns en väl utvecklad medvetenhet om att olika segment bland de presumtiva kunderna bär på olika föreställningar. Det gäller därför att balansera mellan konservativa föreställningar om själva tekniken som manlig och mer "radikala" som ifrågasätter rådande schabloner - t ex att det bara är killar som tillåter sig att betrakta tekniken som ett medel för *både* nytta och nöje.

Detta perspektiv - att kvinnor och män har olika syn på teknik - framhålls ofta som ett erkännande av de olika villkor som formar oss som män respektive kvinnor. Kvinnor får lov att använda tekniken som de vill utan att detta ses som sämre än det "manliga" sättet. Samtidigt medför ett synsätt som visserligen värderar ett "kvinnligt" förhållningssätt som lika mycket värt som ett "manligt" en uppenbar risk att rådande könsroller och föreställningar kring teknik cementeras. För de kvinnor som är intresserade av teknik på ett sätt som kännetecknas av den manliga normen - och t ex tycker det är oerhört fascinerande att pilla i datorns innandöme - eller för de män som bygger upp och vårdar relationer genom att chatta och surfa på Internet, men aldrig skulle komma på tanken att skruva av hårddiskens skyddshölje - typiska kännetecken för den kvinnliga normen - för dessa individer blir tillvaron en ständig påminnelse om att de "inte är som andra". Som väl är finns det otaliga exempel på människor som är på väg att bryta sig ur rådande normer om "hur man bör vara" och tillåter sig att älska datorer och människor på samma gång. Det är dock viktigt att inse att för att dessa människor inte ska tvingas leva med att utgöra undantag som bekräftar regeln (om att kvinnor och män har olika syn på teknik) så måste det till ett annat perspektiv. Detta andra perspektiv ser såväl kön som teknik som ofärdiga entiteter som skapas och omskapas genom förhandling. Med förhandlingen som utgångspunkt blir tillvaron betydligt mindre stereotyp och skapandet av förhållningssätt gentemot teknik såväl som gentemot oss själva blir öppet för prövningar och anpassningar till rådande situation och sammanhang. För att göra dessa förhandlingar möjliga krävs dock en medvetenhet om att världen inte är en utan många och att dessa världar kan betraktas ur flera olika perspektiv. Det är samtidigt viktigt att komma ihåg att vissa av dessa perspektiv innebär skapandet av förtryckande strukturer medan andra medför möjlig frigörelse. Att skapa jämlika förutsättningar i utbildningssammanhang blir därför en fråga om hur man skapar utrymme för mångfald.

4.2 Formandet av kunskap

Synen på kunskap i ett större perspektiv, d v s epistemologi, är avgörande för hur kunskap i ett mindre perspektiv formas. Villkoren för formandet av kunskap bestäms av den

[48] Det senare förhållningssättet har i svensk forskning förts fram av bl a Christina Mörtberg - t ex i hennes doktorsavhandling *"Det beror på att man är kvinna…": gränsvandrerskor formas och formar informations-teknologin.*

övergripande synen på hur detta bör ske. I detta avsnitt försöker jag beskriva olika föreställningar och förhållningssätt kring skapandet av (programmerings)kunskap.

Fokusering på teoretisk kunskap

Alison Adam(1995) undervisar och forskar inom området artificiell intelligens (AI) vid University of Manchester, Institute of Science and Technology. I sin forskning riktar hon kritik mot symbolisk AI[49], den inriktning av AI som representeras av synsättet att hjärnan kan jämföras med en dator i sitt sätt att arbeta. Förenklat innebär det att hjärnan anses fatta beslut utifrån vissa givna fakta. Adam hävdar att inom symbolisk AI finns alltför många exempel på att kunskapsbaserade system, s k expertsystem, skapas utifrån en positivistisk kunskapssyn, d v s man tror sig återge någon form av objektiv kunskap i expertsystemen. Denna inställning diskuteras även i boken *Gendered by Design*, men med fokus inställt på hur administrativa system tenderar att utvecklas utan att tillräcklig hänsyn tas till vems och vilken kunskap som borde relateras till. Genom att fokusera enbart på den synbara, materiella kunskapen går man miste om den tysta kunskap som är kopplad till yrkeserfarenhet. Något som sitter i kroppen och som har avgörande betydelse för hur ett arbete faktiskt utförs. Denna tysta kunskap[50] är dock i regel inte verbaliserad, eftersom det är omöjligt att ge den en, formellt sett, exakt beskrivning - något som bidrar till att den osynliggörs än mer.

Adam menar att den positivistiska kunskapssyn som förmedlar illusionen av att förvärvandet av kunskap primärt handlar om att ta till sig formellt avgränsade (objektiva) fakta, helt fokuserar på teoretisk kunskap i form av påståendekunskap (knowing what). Påståendekunskapen är relativt lätt att avgränsa och definiera med en sådan utgångspunkt, men det man förlorar är de kunskaper som har med färdighet och förtrogenhet (ofta tyst kunskap) att göra. I *Det praktiska intellektet - Datoranvändning och yrkeskunnande* formulerar författaren Bo Göranzon vad som kännetecknar de olika kunskapsbegreppen[51].

- **Färdighetskunskap** beskrivs som den praktiska kunskap vi tillägnar oss inom ett område genom att utöva det - t ex ju mer vi kör bil desto större färdighet får vi i att köra bil.
- **Förtrogenhetskunskap** är den kunskap vi förvärvar genom att ta del av andras erfarenheter från området eller av det Göranzon benämner traditionens exempel. Det är ur dessa samlade erfarenheter - färdigheter och förtrogenhet - som vi bygger vår kompetens. *"Samspelet med andra inom en yrkesgrupp blir här av avgörande betydelse."* [52]
- **Påståendekunskap** slutligen, är den teoretiska kunskap som i regel går att läsa sig till, d v s teorier, metoder, föreskrifter och allmänna principer som med tiden formulerats inom ett kunskapsområde.

Göranzon påpekar att när kunskap diskuteras tenderar den sistnämnda, teoretiska kunskapen, att överbetonas på den praktiska kunskapens bekostnad, medan förtrogenhetskunskapen ofta glöms bort helt. Samtidigt är de tre kunskapsformerna intimt förknippade med varandra och skapas genom en fortlöpande dialog. När de inte tillåts göra det därför att färdighets- och/eller

[49] Enligt Adam domineras AI som forskningsområde av två huvudinriktningar. Den ena är symbolisk AI som beskrivs ovan och den andra är associerande AI (jmfr engelskans connectionism). Associerande AI försöker, med en kortfattad förklaring, beskriva de mentala processer som ligger på perceptions- eller förnimmelsenivå.
[50] Allan Janik (1991) skiljer på denna typ av tyst kunskap och den som utgörs av kunskap som är förhållandevis lätt att artikulera, men som den eller de som omfattas av den medvetet eller omedvetet avstår från att uttrycka i ord. Janik påpekar hur den senare varianten av tyst kunskap ibland används som maktmedel.
[51] Formuleringar som i avgörande stycken formats av filosofen Ludwig Wittgensteins skrivningar i ämnet, av Göranzons notapparat att döma.
[52] Göranzon, s 138.

förtrogenhetskunskapen tonas ner eller rent av osynliggörs, töms även påståendekunskapen på sitt innehåll.

Praktisk kunskap

> *"Det är nog rutin och intuition och vana. För det är nog som en hantverkare som ska snida en docka. […] han gör det liksom […] lite på känn och så kommer det fram…"*[53]

Programmerandets karaktär av hantverk, d v s som en i första hand praktisk kunskap, lyfts ofta fram av både dem som utövar programmering och av dem som forskar kring hur programmering lärs in/ utövas.[54] Innebörden av praktisk kunskap diskuteras av Allan Janik - bl a i *Kunskapsbegreppet i praktisk filosofi* respektive i *Cordelias tystnad - om reflektionens kunskapsteori.* Janik påpekar att praktisk visdom i stor utsträckning handlar om en personlig kunskap, där det gäller att komma underfund med vad som är rätt och fel för *mig*, ungefär som en skicklig hantverkare som funnit sin egen stil. För att nå dithän krävs dock att man både kommer underfund med vilka regler som kännetecknar ett kunnande - och att man lär sig tillämpa dem. För att förstå de grundläggande reglerna krävs att någon *visar* exempel på vilka dessa är och hur de kan tillämpas. Parallellt med detta måste jag sedan själv får öva - med möjlighet till kontinuerlig feedback i form av konstruktiv kritik från någon med djup erfarenhet av området. Överfört på programmering skulle detta innebära att som novis måste man få hjälp att se de mönster som kännetecknar lösningen av ett problem och hur denna lösning omvandlas till programkod. Visserligen kan lösningar ofta se ut på många olika sätt, men för att få en chans att förstå det måste jag få många olika exempel. *"Poängen är att man lär sig veta vad man ska se."*[55]

Janik relaterar till den femstegsmodell för hur vi förbättrar vårt yrkeskunnande, som utvecklats av den amerikanske filosofen Hubert Dreyfus[56]. Modellen beskriver fem utvecklingsstadier varav de två första utgörs av kunskapsutvecklingen hos noviser respektive nybörjare.[57]

Det första stadiet (novis) kännetecknas av att nybörjaren lär sig behärska ett fåtal regler. Som nybörjare är man oftast nyfiken, men det finns också en begränsning i hur mycket ny information man kan ta emot. Av den anledningen anses det ofta bäst att koncentrera sig på förståelsen av vedertagna fakta och regler för hur dessa fakta kan bearbetas. Sociologen Bo Eneroth beskriver detta stadium i följande ordalag, betraktat ur den lärandes perspektiv:

> *[…] det är i situationer vi inte kan så mycket om, inte har mött så ofta i praktiken, är ovana vid, som vi känner det största behovet av att begränsa oss till de viktigaste faktorerna, analysera dem rationellt och helst vara förberedda med inlärda rutiner som vi automatiskt kan tillgripa. Vi känner ett nästan desperat sug*

[53] En programmerare med mångårig erfarenhet beskriver programmering som aktivitet. Ur Selling Sjöberg & Tjernström s 5.
[54] Göranzon, Papert & Turkle, Selling Sjöberg & Tjernström m fl.
[55] Janik (1991), s 137.
[56] I *Mind over Machine. The Power of Human Intuition and Expertise in the Era of the Computer.*
[57] De tre övriga utvecklingsstegen rubriceras 3) Kompetent, 4) Skicklig, 5) Expert.

> *efter metoder, tekniker och något slags "teoretisk" modell som beskriver situationen för oss.*[58]

I det andra stadiet, som benämns nybörjarstadiet, betonas vikten av praktisk erfarenhet i verkliga situationer. Den avancerade nybörjaren kan nu börja upptäcka delar som inte är exakt definierade, men som har betydelse för hur en uppgift kan lösas. Hon eller han börjar skönja likheter mellan aktuellt problem och problem som hon/ han stött på tidigare. Vikten av praktisk erfarenhet betonas, även om den teoretiska grunden ses som självklar. *"Det är praktiken som kan ge de teoretiska kunskaperna liv".* [59]

Janik gör dessutom en viktig distinktion vad gäller regler som koncept. Regler kan vara såväl regulativa som konstituerande. Regulativa regler är normativa, d v s de bestämmer vilka handlingar som är tillåtna och vilka som inte är det. Trafikregler utgör ett exempel på regulativa regler. Konstituerande regler är *"typfall eller exempel som spelar en mönster-bildande roll när det gäller att bestämma den 'regelmässighet' eller ordning som skall prägla en viss aktivitet."*[60] Programmering, liksom de flesta andra aktiviteter, kännetecknas av båda dessa regeltyper. Kodstandard kan sägas utgöra exempel på regulativa regler medan själva problemlösningen och hur denna utförs för att resultera i ett fungerande program är exempel på konstituerande regler. Genom olika exempel på hur man kan lösa likartade problem får novisen möjlighet att öva sig i hur man gör. De konstituerande reglerna är med Janiks ord *"handlingsexempel som är till för att upprepas"*[61]. För att ta till sig de normativa reglerna, påpekar Janik vidare, måste man först lära sig behärska tekniken, eller hur man gör. Översatt till programmeringskunskap skulle man kunna säga att för att förstå syntaxen och varför den är uppbyggd som den är, måste man först förstå hur man löser problemet och omvandlar lösningen till kod. Skrivandet av pseudokod kan sägas utgöra ett försök till att överbrygga steget från problemlösning till korrekt syntax i programmering. Janik menar att för att utveckla ett (yrkes)kunnande måste man lära sig "hur" innan man lär sig "vad". I formell undervisning betonas, enligt Janik, den påståendekunskap som utgörs av "vad". *"Betecknande för detta slags kunskap är förmågan att återge det vi kan i skriftliga prov."* [62]

Föreställningar kring programmering

För att bättre förstå vilka processer som försiggår vid formandet av programmeringskunskap, tog jag hjälp av olika undersökningar kring programmering som aktivitet. En av dessa texter var Shirley Booths doktorsavhandling *Learning to program - a phenomenographic perspective*. Den centrala frågeställningen hos Booth är: Vad innebär det och vad krävs för att lära sig programmera? Booth försöker i sin studie att klargöra ett urval nybörjarstudenters förståelse av programspråket SML (Standard Meta-Language), ett funktionellt språk. Hon sätter fokus dels på ramverket i form av föreställningar om vad det innebär att programmera, att lära sig programmera och föreställningar om vad ett programspråk egentligen är. Dels undersöker hon studenternas förståelse av ett antal begrepp relaterade till programmering: funktioner, rekursion och korrekthet (jmfr. felhantering). Därutöver har Booth studerat hur några av studenterna närmat sig ett antal givna (programmerings)problem.

[58] Eneroth, s 68.
[59] Precision och Improvisation, s 13.
[60] Janik (1996), s 42.
[61] Ibid, s 43.
[62] Ibid, s 46.

Vad gäller *synen på programmering* formulerar Booth tre huvudkoncept utifrån sin studie.[63] Dessa visar att programmering betraktas som en aktivitet som är antingen:
1) **datororienterad**
2) **problemorienterad**
3) **produktorienterad** .

Vad gäller *synen på programmeringsspråk* (i allmänhet) finner Both fyra olika synsätt.[64] Programmeringsspråk som:
1) ett **användbart program** , där språket betraktas som ett fristående objekt som möjliggör att skriva program på ett visst sätt, eller passa för en speciell tillämpning.
2) **kod**, d v s instruktioner, kommandon och annan syntax som kännetecknar ett programspråks uppbyggnad.
3) **medel för kommunikation** - t ex mellan programmeraren och datorn eller mellan datorn och programmets slutgiltiga användare.
4) ett **sätt att uttrycka sig** genom att formulera lösningen på ett problem eller genom att uttrycka en idé på ett sätt som kan användas/ påverkas av datorn.

Slutligen formulerar Booth också fyra koncept om vad det innebär att lära sig programmera:[65]
1) **lära sig ett programmeringsspråk** . Där fokus ligger på att lära sig kännetecken och detaljer i syntaxen för ett eller flera språk.
2) Lära sig att **skriva program i ett specifikt programspråk** .
3) Lära sig att **lösa problem** i formen av datorprogram, d v s att lära sig producera program som löser problemet.
4) För att **bli en del av det/ de sammanhang där datorprogram** skapas, med fokus på att lära sig lösa problem och skriva program i samarbete med eller för andra.

Booth påpekar att de olika ramverk för förhållningssätt som hon identifierar i sin undersökning bottnar i den enskilde studentens erfarenheter av (eller brist på) aktiviteter som programmering och problemlösning. Ramverken påverkar, på olika sätt och i varierande grad och kombinationer, hur den enskilde studenten tar till sig olika aspekter av programmeringskunskap, genom att fokus på vad som är väsentligt skiftar. Slutsatserna man bör dra är, enligt Booth, att:

> *Implications for teaching lie in the nature of the conceptions identified. Stress is placed on the framework constituents in that teaching should encourage well-founded conceptions through example. Teaching should not merely try to bring about expert behavoiur in students by offering expert views on the content, but give students the range challenges which enable them to come to the expert understanding via experience. Teachers should above all be aware of the range of conceptions held by their students, and that* ***poor conceptions are not necessarily caught in lab exercises and examinations*** *(min markering.)*[66]

När jag analyserade resultaten från genomförda intervjuer blev det uppenbart för mig att studenterna vid de båda programmen mycket sällan såg programmering som en i första hand problemorienterad aktivitet, utan snarare som antingen datororienterad eller produkt-orienterad. Innebörden av att lära sig programmera koncentrerade sig därför i stor

[63] Booth, s 302.
[64] Ibid., s 303.
[65] Ibid., s 304.
[66] Booth, abstract.

utsträckning till att lära sig ett programmeringsspråk och/ eller lära sig skriva program i ett specifikt programspråk. För många framstod det också som centralt att bli en del av det/ de sammanhang där datorprogram skapas genom att få anställning inom området, dock inte nödvändigtvis som programmerare. De intervjuade lärarna däremot betonade den problemorienterade sidan av programmering, men uttryckte också att de uppfattade att just den sidan orsakade problem hos många studenter. Vad gäller synen på programmeringsspråk förekom alla fyra synsätt som Booth definierar. Dock var det enbart bland studenter och lärare med lång erfarenhet och upplevelser av programmering som en lekfull aktivitet, som programmering och programmeringsspråk sågs som ett sätt att uttrycka sig. Booths slutsats att bristande förståelse (för programmeringsämnet) inte nödvändigtvis upptäcks genom laborationer och tentamina har även påtalats av andra[67]. I studentintervjuerna framkom det också med all tydlighet att ett godkännande i programmeringskursen inte nödvändigtvis innebar förståelse för ämnet.

Vid läsningen av Christina Björkmans rapporter om kvinnliga studenters upplevelser av datavetenskap i de sammanhang där Björkman verkar[68], fick jag ytterligare bekräftelse för att de svårigheter i att förvärva kunskap i programmering som jag upplevt - hos mig själv och andra vid BTH - existerade även vid andra högskolor. Christina Björkmans iakttagelser kan sammanfattas i rubriken till rapporten från hennes föredrag *Varför väljer kvinnor inte datavetenskap? - och hur kan vi stötta dem som gör det?* [69] Björkman visar i sin rapport på några viktiga faktorer:

- Att datavetenskapen som disciplin både i utbildnings- och arbetssammanhang betraktas som könlös samtidigt som den i praktiken domineras av män, vilket Björkman ser som problematiskt.
- Att bilder och föreställningar kring vad som är datavetenskap/ teknik, vem som är datavetare/ datatekniker och vad det innebär att arbeta som datavetare/ datatekniker måste problematiseras mer. Varifrån kommer t ex föreställningar om att arbete med datorer innebär "långa, ensamma nätter framför datorn"?
- Att den identitet studenten skapar vid förvärvandet av en utbildning och formandet av en framtida yrkesroll kompliceras för de kvinnliga studenterna i en miljö där kommentarer i stil med att det behövs fler "duktiga datakillar" är legio. (Även om kommentarerna snarare ger uttryck för bristande medvetenhet än om att deras uttalare inte önskar se duktiga datatjejer.)

Björkman har arbetat med olika projekt som haft för avsikt att stödja de kvinnor som valt att läsa datavetenskap (för att i förlängningen förhoppningsvis också rekrytera fler kvinnliga studenter). Det mest signifikanta är projektet som beskrivs i rapporten *Projekt Q+ . Med och för kvinnliga studenter i datavetenskap*. Björkman har dels genom enkäter, dels genom intervjuer försökt besvara frågor kring:[70]

- hur studenterna har uppfattat utbildningen (generellt och inom de olika ämnena)
- varför de valt den
- vilka förkunskaper de haft med sig (med särskild vikt lagd vid datorvana respektive kunskaper i programmering)
- synpunkter på könsfördelningen
- vilka programmeringskunskaper studenterna anser sig ha förvärvat efter första årskursen.

[67] Bl a i Christina Mörtbergs bidrag till *Datateknisk ingång för kvinnor, en utvärdering*.
[68] Christina Björkman har arbetat som lärare på Datavetenskapliga programmet vid Uppsala Universitet och undervisat bl a i programmering. Hon har också arbetat som studievägledare för programmet. Se även fotnot 34.
[69] Föredraget hölls vid konferensen Kvinnor och Matematik, som arrangerades i Uppsala den 16-18 april 1999.
[70] Björkman (2000), s 14-15.

Enkäterna och intervjuerna har ingått som en del i projektet och utgjort ett viktigt underlag vid utformande av aktiviteter som syftat till stödja de kvinnor som studerat vid programmet.

Den bild som växer fram, vad gäller projektets resultat och som har direkta beröringspunkter med mitt arbete, visar på några betydelsefulla faktorer: Kvinnorna vid utbildningen är i minoritet. De har betydligt mindre erfarenhet av datorer och programmering då de påbörjar utbildningen än sina manliga studiekamrater. Det går att visa ett samband mellan de förkunskaper studenterna har i programmering och deras studieresultat i programmeringskurserna (såväl som deras egen upplevelse av hur väl de lyckats). I diskussionsavsnittet (kapitel fem) för jag ett resonemang kring detta under rubriken *Skapandet av kollektiv kunskap.*

Förhållningssätt kring programmering som aktivitet

Sherry Turkle och Seymour Papert propagerar i sin artikel: *Epistemological Pluralism - Styles and voices within the Computer Culture*, för en större öppenhet gentemot olika inlärningsstilar i programmeringsundervisningen. De har tillsammans intervjuat studenter på såväl högskole- som grundskolenivå och därigenom fått många olika förhållningssätt beskrivna för sig. Problemet är, enligt författarna, att vissa inlärningsstilar har högre status än andra:

> *Our data points to discrimination in the computer culture that is determined not by rules that keep people out, but by ways of thinking that make them reluctant to join in.*[71]

Turkle och Papert menar att det förhållningssätt som premieras kännetecknas av ett, i formell bemärkelse, mycket strukturerat närmande där problemen bryts ner i separata delar för att möjliggöra designen av modulära lösningar, vilka sedan kan passa in i den avsedda helheten. Författarna påstår inte att detta förhållningssätt är felaktigt på något sätt, tvärtom påpekar de att vissa studenter lär sig programmera mycket bra genom att använda sig av denna hållning. Det finns dock en hel del studenter som har andra sätt att närma sig programmeringen varav ett utgörs av en stil som författarna benämner "bricolage"[72]. Denna stil kännetecknas, enligt författarna, av att studenten tillägnar sig kunskap genom att arrangera och omarrangera, genom att förhandla och omförhandla med utgångspunkt från det givna materialet. Jag vill kalla det ett improviserande arbetssätt.

> *[...] their work at the computer is marked by a desire to play with the elements of the program, to move them around almost as they were material elements - the words in a sentence, the notes in a musical composition, the elements of a collage.*[73]

Författarna menar att detta sätt att lära sig, om det alls tillåts, inte betraktas som lika välutvecklat och därmed sämre än det formellt strukturerade. Det närmar sig inte det stadium av abstraktion som enligt västerländsk pedagogik (författarna hänvisar här till Piaget) är det

[71] Papert & Turkle s 132.
[72] Uttrycket kommer från antropologen Lévi-Strauss som använde begreppet för att kontrastera den västerländska analytiska vetenskapen mot den "konkreta vetenskap" som han menade tillämpas i s k primitiva samhällen.
[73] Papert & Turkle, s 136.

högsta stadiet i intellektuell utveckling. Papert och Turkle beskriver hur denna tendens att värdera kunskap och de tillvägagångssätt som används för att tillägna sig kunskap, skapar en känsla av mindervärde hos de studenter som inte använder sig av det "påbjudna" sättet. En känsla som i värsta fall kan få dem att tappa lusten för ämnet programmering. Vad som är intressant är att författarna ser en tydlig koppling mellan de(t) sätt att närma sig problem som de benämner bricolage och den programmeringskultur som synliggörs i hackerkulturen[74].

I Astrid Selling Sjöbergs och Mattias Tjernströms examensarbete *"Lära sig leva i den grå zonen"* framkommer en del av de skillnader som kännetecknar en erfaren programmerare i förhållande till en oerfaren (e g nybörjare). De oerfarna programmerarna brottas bl a med svårigheter att förstå helheten. Hon/ han har också problem med att läsa och förstå den egenhändigt framtagna kodtexten. Många har dessutom svårigheter med att över huvud taget skapa kodtext (min anm.). För nybörjare innebär detta, enligt Selling Sjöberg och Tjernström, att de har svårt att se vart de är på väg och hur problemen hopar sig. Den erfarne programmeraren däremot har lärt sig att upptäcka brister i programkoden och kan därigenom åtgärda dem innan de utvecklas till problem. De har ett annat förhållningssätt till programmeringen än nybörjarna. Ett förhållningssätt som kännetecknas av ett flöde i arbetet. De har tillräcklig erfarenhet för att inte fastna i små enkla stavfel, odeklarerade variabler mm och de behöver inte hela tiden tänka på vad de gör och varför. Programmeringen sitter i kroppen. De erfarna programmerarna som Selling Sjöberg och Tjernström intervjuat har flerårig vana av yrkesmässig programmering inom högskolan och de har även undervisat i programmering. De har dessutom många års erfarenhet av "hemmaprogrammering" eller "hackande" som de själva kallar det[75]. Det intressanta är att de beskriver hemma-programmerandet som mer lekfullt än yrkesprogrammerandet. De beskriver också behovet av snabb feedback i hemmaprogrammerandet för att upprätthålla motivationen. Feedbacken skapar de själva genom att t ex göra testklasser eller skapa enkla användargränssnitt. Det viktiga är att få se någon form av resultat som bekräftar att koden fungerar och utför det den ska. Samtidigt är det tydligt hur deras dubbla erfarenheter av yrkesprogrammering och eget "hackande" befruktar varandra. De kunskaper de "lekt" sig till skapar grunden för att de ska förstå och kunna ta till sig de mer formella metoder och arbetssätt som kännetecknar yrkesprogrammerandet. Metoderna blir i sin tur en del av den omedvetna hållning som kännetecknar hemmaprogrammerandet. På så sätt kan de upprätthålla ett kreativt förhållnings-sätt i båda situationerna.

Selling Sjöbergs och Tjernströms arbete handlar om att identifiera de olika processer som noviser och nybörjare respektive erfarna programmerare genomgår i skapandet av programkod. Utifrån denna identifikation diskuterar de hur utformningen av ett utvecklingsverktyg skulle kunna se ut som ger stöd framförallt för nybörjaren, men som även kan fungera för den mer erfarne programmeraren. De problem de har koncentrerat sig på rör läsande och tolkande av kodtextens struktur (t ex nästlade styrstrukturer i form av olika slags loopar), svårigheten att leta fel på rätt ställe, och problem att förstå systemet i sin helhet. För den sista uppgiften har de försökt föra samman t ex användargränssnitt eller klassdiagram med kodtexten. Deras förslag visar på att det finns möjligheter att utveckla verktyg som fungerar betydligt bättre i utbildningssammanhang än de som används idag.

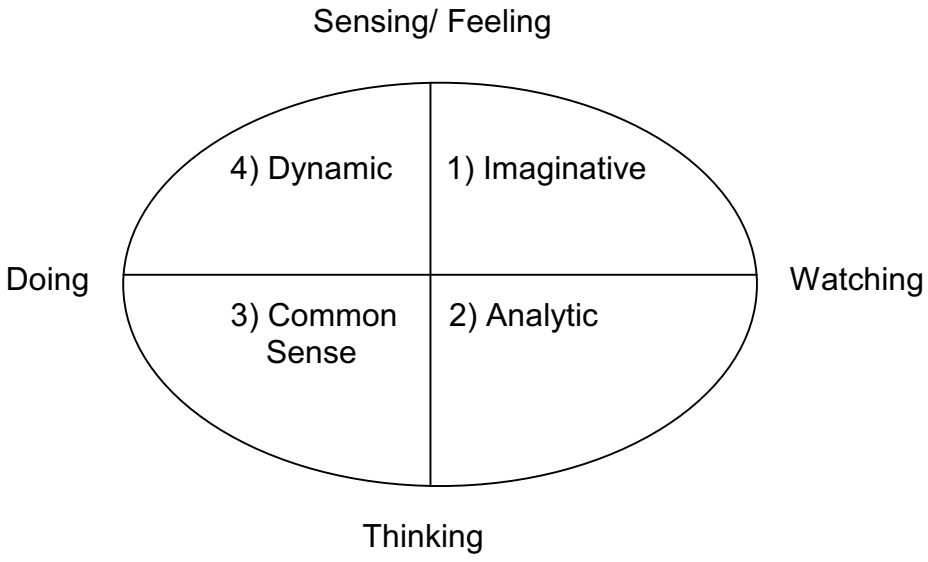
[74] Se t ex Jörgen Nissens artikel i *Från symaskin till cyborg* för en beskrivning av hackerkulturens historia och kännetecken.
[75] De intervjuade programmerarna började laborera med programkod redan i 10-års åldern.

Olika inlärningsstilar

I de intervjuer jag genomförde framkom att olika studenter har olika sätt att närma sig programmeringen. Detta bekräftas också av de undersökningar jag refererar till. Vikten av att erkänna och ge utrymme för olika förhållningssätt i programmeringen framhålls också.

Pedagogen Bernice McCarthy (1990) har försökt utveckla en modell som ger utrymme för olika förhållningssätt i studenters och lärares kunskapsprocess - "the 4MAT System". Hon identifierar fyra huvudsakliga inlärningsstilar, men påpekar med emfas att ingen använder en inlärningsstil helt uteslutande av en annan. I korthet utgörs de olika stilarna av:[76]

1) **Imaginative learners**. En stil som kännetecknas av att information uppfattas konkret och reflekteras genom den egna erfarenheten. All kunskap integreras med det egna skapandet av en identitet och med strävan att förstå den omgivande världen.
2) **Analytic learners.** Inom denna grupp återfinns individer som uppfattar information abstrakt, men bearbetar den på ett reflekterande sätt. De skapar teorier genom att integrera observationer med det de vet. De lär sig genom att tänka genom idéer och har stort behov av detaljkunskap för att förstå ämnet. De värderar linjärt tänkande, är ofta mycket noggranna och de hyser stor tilltro till experter.
3) **Common Sense learners** uppfattar information abstrakt, men bearbetar den aktivt. De lyckas integrera teori och praktik genom att pröva sig fram. De är ofta pragmatiker - om någonting fungerar, så använd det. De värderar strategiskt tänkande och tycker om att experimentera och ta reda på hur saker fungerar.
4) **Dynamic learners** anses uppfatta information konkret och bearbeta den aktivt. De integrerar erfarenheter och tillämpningar och lär sig ofta genom "trial and error". De är ofta entusiastiska inför nya saker och i regel mycket flexibla. De kan ofta dra riktiga slutsatser trots att de saknar en logisk underbyggnad.



Figur1. Inlärningsstilar. Efter McCarthy, s 32.

[76] McCarthy, s 32.

Som framgår av figur 1 omges inlärningsstilarna i sin tur av olika förhållningssätt som identifieras som känsla (Sensing/ Feeling) kontra förnuft (Thinking) respektive handlande (Doing) kontra iakttagande (Watching). Dessa förhållningssätt är inte absoluta utan ofta befinner man sig mellan flera av dem. Om man tänker sig två skalor som korsar varandra varav en lodrät och en vågrät, så utgör känsla respektive förnuft polerna på den lodräta axeln och handlande respektive iakttagande utgör polerna på den vågräta. De flesta av oss känner oss mer bekväma när vi får lov att befinna oss på vissa delar av skalan än andra. Var man känner sig mest bekväm varierar från individ till individ beroende på vilka vi är, vilket sammanhang vi befinner oss i, hur vi ser på oss själva, vad vi finner viktigt och vad andra frågar efter och förväntar sig av oss (eller vad vi tror att de förväntar sig, min anm.) McCarthy avser alltså inte på något sätt ställa de olika förhållningssätten mot varandra. Tvärtom vill hon genom att identifiera dem propagera för att de erkänns som likvärdiga. Hon menar också att eftersom respektive förhållningssätt inrymmer såväl styrkor som svagheter behöver vi egentligen lära oss att integrera dem, eftersom de så uppenbart kompletterar varandra. Tyvärr, menar McCarthy att dagens utbildningsväsende alltför ofta tenderar att få slagsida gentemot den analytiska inlärningsstilen(2), vilket komplicerar lärandet för eller i värsta fall helt utestänger dem som finner sin inlärningsstil inom någon av de andra tre kategorierna. Därför menar McCarthy att det är oerhört väsentligt att lyfta fram de olika sätt att närma sig kunskap som kan identifieras för att därigenom skapa mer jämlika förutsättningar för lärande, men också för att skapa förståelse för den alltmer komplexa och interaktiva värld vi lever i. I synnerhet är det väsentligt att de som har till uppgift att "förmedla" kunskap genomgår denna process. För att de ska kunna göra det och för att de ska kunna skapa förändring, krävs att också de ges möjlighet till feedback.

Slutligen betonar McCarthy vikten av att se hur mångfacetterad läroprocessen är och att förståelse av och förändring av de villkor som skapar olika lärprocesser kräver insikt inte bara om olika inlärningsstilar utan också om "förmedling" av kunskap, undervisningsplanens uppläggning, och hur utvärdering av kunskap görs.

4.3 Vägledare i formandet av kunskap

Vad innebär det att vägleda studenter i ett ämne som programmering? Hur skapar man goda förutsättningar för ett lärande klimat? Vad beror det på att det ibland tycks vara så svårt att leva upp till de förväntningar och föresatser som studenter respektive lärare uttalar? Dessa och andra frågor har jag ställt mig under arbetets gång. Nedan följer ett resonemang kring några av dessa frågeställningar. Till att börja med vill jag för säkerhets skull klargöra att då jag använt begrepp som förmedlare(lärare) och förmedling(vägledning), har jag inte haft för avsikt att lägga någon värdering i dessa begrepp.[77]

[77] Ordet förmedla översätter det inom engelskan betydligt öppnare begreppet ”intermediate”, som också innebär mellanliggande, mellanhand o s v. Begreppet används bl a av Ehrlich & Kate i deras artikel *The Invisible World of Intermediaries.* Ett möjligt ord för att ersätta subjektet ”förmedlare” skulle kunna vara ”intramediär”, vilket anger att den som verkar som vägledare gör det inte som ”mellanhand” utan ”inom” (intra = inom) det sammanhang som kunskapen formas i.

Förmedlarrollen

> *"Arbetet med att förmedla ett synsätt har inte det ensidiga fokus på instruktioner och beskrivningar som man först kan tro. Det är återkopplingen av frågor, där konstruktören testar sitt omdöme mot bilden och utvecklingen av dessa frågeställningar som är den andra delen i förmedlandets dialog. Denna återkoppling av frågor har samma funktion som när skådespelaren visar upp en del av pjäsen under repetitionsarbetet"* [78]

Konstruktören i citatet ovan kan lika gärna vara programmeraren som vill diskutera olika lösningsförslag. Oavsett om det handlar om en mer erfaren programmerare eller en helt oerfaren finns behovet av att diskutera med någon annan, få respons, kritik, förslag på förändringar. För novisen handlar det ofta om att helt enkelt få hjälp att *"tyda och tolka de mönster som kodtexten uppvisar"*[79]. Behovet att få hjälp utifrån - genom böcker, Internet, lärare och andra studenter är mer markant hos nybörjaren än hos den som har lite mer erfarenhet av programmering, vilket Selling Sjöberg och Tjernström påpekar. Jag skulle dessutom vilja tillägga att hos novisen, d v s för den som befinner sig i det första stadiet i den av Dreyfus definierade utvecklingsmodellen[80], är behovet av mänsklig vägledning extra stort. För novisen kan även kurslitteraturen och hjälpsidorna på Internet te sig nog så kryptiska. De intervjuade studenterna vittnar om detta när de uttalar svårigheter med att översätta det de läser i kurslitteraturen till praktisk handling eller med att tolka API:et. Med en lärares vägledning kan dock denna information bli både intressant och framförallt användbar.

> *"Making sense of information is a realtime process, and is often both collaborative and emergent."* [81]

För att klara av att tolka den information som finns att tillgå behöver novisen få hjälp av någon som har en välutvecklad ämnesmässig förtrogenhet. En sådan person har förvärvat både praktisk erfarenhet, genom att själv ägna sig åt programmering, och är väl förtrogen med praxis i form av kodstandard och annat. Idealt har personen i fråga också goda teoretiska kunskaper och kan därmed knyta ihop de tre kunskapsformerna: färdighetskunskap, förtrogenhetskunskap och påståendekunskap. Samtidigt är det viktigt att inse att förmedlarrollen innebär ett kontinuerligt lärande. För att lära krävs feedback. Även lärare behöver således återkoppling från andra. Återkopplingen kan ske på flera sätt. Primärt sker den dock i mötet med studenter respektive med andra lärare.

En av de intervjuade lärarna vill se sig själv som en mentor, någon som kan utgöra ett stöd i studenternas egen läroprocess. Han påpekar dock att det inledningsvis kan vara ganska svårt, eftersom många studenter tenderar att betrakta honom som en expert i bemärkelsen att de förväntar sig att han ska kunna "allt". En annan lärare påpekar att hon upplever att hon lärt sig oerhört mycket av att försöka förklara och visa för andra. Mentorskap som ett möjligt förhållningssätt i förmedlarrollen, framhålls även av erfarna systemutvecklare. Eftersom de egna erfarenheterna hos den som agerar mentor lyfts in i ett sammanhang, upplever många som intar rollen att den skapar energi.[82] Samtidigt är det viktigt att som mentor analysera sitt

[78] Precision och Improvisation, s 74.
[79] Selling Sjöberg och Tjernström, s 31.
[80] Se avsnittet om Praktisk kunskap.
[81] Ehrlich & Kate, s 164.
[82] Precision och Improvisation s 73 ff.

sätt att agera gentemot adepten, vari ingår att identifiera sin egen inlärningsstil för att tala med McCarthy.

Att göra förmedlarrollen synlig

Att förmedlarrollen kan te sig problematisk i och med att det inte alltid är uppenbart vad den innebär diskuteras bl a av de två systemvetarna Kate Ehrlich och Debra Cash (1999). De har studerat förmedlarrollen vad gäller yrken som har det gemensamt att de förmedlar information och tjänster som också kan nås via datoriserade nätverk. Deras studier pekar på att den erfarenhet och expertis som de mänskliga förmedlarna kan ge uttryck för på ett helt annat sätt än de datoriserade (t ex genom sin unika förmåga att lyssna, föra en för sammanhanget anpassad dialog och överhuvud ta hänsyn till faktorer som bestäms av den aktuella situationen) tenderar att bli helt osynliggjord i den värld vi lever. Såväl de kunder som anlitar dessa förmedlare av information och tjänster, som de organisationer de verkar inom, och de arbetsgivare som anställer dem tenderar att inte se vidden av deras kunskap. Olika aspekter av enskilda yrkens synlighet diskuteras också av Susan Leigh Star och Anselm Strauss i deras artikel *Layers of Silence, Arenas of Voice*. En av de aspekter de diskuterar är "uteslutande av bakomliggande arbete" (disembedding background work[83]). Innebörden av denna aspekt är att i en synlig yrkesroll (som t ex lärarens) ingår väldigt mycket arbete som för en yttre betraktare inte är uppenbart på annat sätt än upplevelsen av att en lärare är "bra eller mindre bra", ungefär som att en nyhetsreporter eller skådespelare är det. Det Star och Strauss pekar på är att denna upplevelse, av hur väl en yrkesutövare utför sitt yrke, i hög grad bestäms av det arbete och den erfarenhet som ligger bakom. Att en lärares arbete inte enbart består av att möta en grupp studenter under en lektion eller av att rätta skrivningar, är säkert uppenbart för de flesta. Däremot är det kanske inte lika uppenbart på vilka sätt lärare arbetar med att forma strategier för *hur* hon/han/de ska förmedla ett kunnande liksom för *vad* som egentligen är viktigt att förmedla.

Den dialog som möjliggörs i mötet med en mänsklig förmedlare innebär i utbildningssammanhang att studenterna med stöd av lärarna ges tillfälle att stegvis ta till sig såväl de konstituerande (hur) som de regulativa (vad) regler som kännetecknar ämnet. En lärare i programmering kan med tiden träna upp sin förmåga att lyssna sig till vilka svårigheter studenterna brottas med generellt såväl som enskilt. I intervjuarbetet var det uppenbart att de lärare som hade lite längre erfarenhet av undervisning, var mer medvetna om vikten av mångfald i undervisningssituationen. Betydelsen av att kunna presentera samma sak på många olika sätt var m a o självklar för dem. Likaså framhöll de vikten av att i undervisningssituationen ge utrymme för reflektion genom en fortgående dialog mellan studenter såväl som mellan studenter och lärare. De lärare som inte hade tidigare undervisningserfarenhet var i de flesta fall på de klara med vilken typ av svårigheter studenterna brottades med. Det de tycktes sakna var förmågan att anpassa undervisningen till den aktuella situationen. De hade m a o inga strategier för att rekapitulera sin undervisning från en annan vinkel. Detta ger en antydan om att den komplexitet förmedlandet av (programmerings)kunskap innebär, inte är särskilt synlig. I så fall skulle knappast oerfarna lärare anställas utan möjlighet till uppbackning från mer erfarna lärare. De skulle heller inte ges en tidsram som saknar utrymme för omplanering, något som självklart tar längre tid för den oerfarne än för den erfarne.[84]

[83] Star & Strauss, s 15.

[84] Bristen på pedagogisk uppbackning respektive den snäva tidsramen är problem som påtalats både av författare till andra rapporter som jag refererar till i detta arbete och av de lärare jag intervjuat. Programmeringslärarna vid MT-programmet tycks dock inte upplevt just dessa faktorer som särskilt problematiska. De har visserligen

Jag vill betona att jag inte betraktar inlärningsprocessen som enbart avhängig de lärare som medverkar i den. Processen sker lika mycket i mötet med andra studenter och inte minst i de sätt som studenten använder för att bearbeta kunskaperna. Lärarens/ lärarnas förmedlarroll är dock central i den bemärkelsen att den utgör (eller borde utgöra) en garanti för att alla studenter ges samma grundläggande möjligheter att bygga upp en förståelse för ämnet. Läraren utgör också en garant för att det finns någon som ser som sitt ansvar att försöka fånga upp de studenter som ser ut att vara på väg att tappa greppet. Förutsättningen för att detta ska fungera är att dessa delar av lärarrollen synliggörs.

Projektarbete och eget lärande

I en omfattande utvärdering som Inger Wistedt gjort på uppdrag av Högskoleverket: *Recruiting Female Students to Higher Education in Mathematics, Physics and Technology*, studeras fem tekniska utbildningar som introducerades vid olika högskolor och universitet under 1995. Målsättningen med utvärderingen var att göra en integrerad studie av de fem, inbördes väldigt olika, utbildningarna där själva initiativet till att utforma nya utbildningar var det man ville analysera. Gemensamt för de olika programmen var deras strävan att rekrytera nya grupper av studenter, i synnerhet kvinnor. Dessutom sammanhölls utbildningarna av en strävan att anpassa sina undervisningsmetoder till dessa nya grupper. Två föreställningar sammanfattar enligt Wistedt de pedagogiska idéer som uttrycktes i intervjuer med lärare och projektledare. Den ena är föreställningen om att lära sig själv ("the self-directed learner"[85]). Den andra handlar om föreställningar kring den sociala karaktären av lärandet, d v s om hur vi som deltagare i en utbildning formar och formas av själva utbildningsmiljön och vilka krav som ställs på dem (oss) som deltar. De delar av utvärderingen som jag finner intressanta för mitt arbete är framförallt de som berör aspekter av studenternas upplevelser kring undervisningen.

En av dessa aspekter är studenternas rädsla för att de undervisningsmetoder som används (e.g projektarbete) ska bidra till att kunskaperna blir alldeles för grunda. Studenterna menar att de projektarbeten som genomförs riskerar att bli väldigt resultatinriktade, eftersom tiden för deras genomförande är starkt begränsad. Studenterna läser dessutom i regel andra kurser parallellt med projektkurserna. Detta innebär att det i praktiken inte blir någon tid över för att fundera djupare kring olika frågeställningar eller söka alternativa lösningar. Wistedt bekräftar denna bild och visar även på hur detta försvårar skapandet av en förståelse för den del av problematiken som utgörs av att uppgifter ofta är motsägelsefulla och att det därför är viktigt att träna upp förmågan till mångfaldigt seende. Att identifiera och formulera olika förhållningssätt till ett givet problem är en väsentlig del av den kreativa förmågan.

Grupparbete som arbetsform är också något som diskuteras och utvärderingen visar att föreställningen om att denna arbetsform skulle passa kvinnor särskilt väl inte är entydig. Flera av de kvinnliga studenterna uttrycker svårigheter med att "tänka själv" i gruppsammanhang och upplever att det ofta blir så att de som anses kunna mest får ta på sig rollen av att förklara för de andra. Detta motverkar det egentliga syftet med grupparbete - nämligen att alla gemensamt diskuterar sig fram till lösningar och argumenterar för och emot dem. För det första, som Wistedt accentuerar, så är syftet inte att hitta en allenarådande lösning. För det

uttryckt önskemål om möjligheter till ämnesmässigt mentorskap och påtalat svårigheter med att studera och undervisa parallellt. Däremot förefaller de relativt nöjda med ramarna i övrigt.

[85] Wistedt, s 44.

andra är gruppdiskussionen avsedd att bidra till vars och ens självförståelse av berörda problem. De intervjuer som presenteras visar dock på att dessa syften med grupparbetena alltför sällan uppfylls. Wistedt diskuterar kring denna problematik och konstaterar att problemorienterat grupparbete som det här handlar om, lika mycket som andra pedagogiker eller undervisningsformer, skapas utifrån vissa givna föreställningar kring utformning och genomförande, men att dessa föreställningar inte artikuleras. Det blir därför lika svårt för de studenter som helt saknar erfarenhet från eftergymnasiala studier, som för dem som har sådana erfarenheter, men av mer traditionell (formell) undervisning. Studenterna uttrycker detta i termer av att det är svårt att förstå vad som förväntas av en. De studenter som har minst svårigheter med detta är de med starkt självförtroende, som törs förlita sig på att de lär sig även om de inte exakt kan förklara hur eller vad de lär sig. Studenter med sämre självförtroende hamnar lätt i en situation där de känner osäkerhet både över vad som förväntas av dem och vad de faktiskt lär sig. Risken att så sker förstärks i de fall där utbildningarna befinner sig i ett sammanhang där omgivande utbildningar utmärks av förmedlandet av påståendekunskap. En kvantitativ jämförelse, där "vad" blir viktigare än "hur", ligger då nära till hands eftersom den i regel är lättare att göra (och framförallt mer synbar vid en ytlig betraktelse) än en kvalitativ.

Studenterna menar att lärare som prövar nya arbetsformer måste ta sitt ansvar för dem i den aktuella situationen (och inte enbart i den efterföljande utvärderingen, min anmärkning.) Detta inkluderar en aktiv handledning som innefattar vägledning i de arbetsformer som prövas och konkreta råd för hur man utvecklar ett kreativt problemlösande. Wistedt konkluderar med att såväl studenter som lärare behöver handledning för att utveckla alternativa arbetsformer och att den process det innebär tar tid och kräver ett stort engagemang från inblandade parter.

De två utbildningar jag har studerat omfattas i hög utsträckning av föreställningar kring den egna läroprocessen. För IT-programmet betonas betydelsen av denna i relation till det faktum att de inledande studierna genomförs på distans. Det innebär att det är mycket upp till den enskilde studenten att skapa den motivation som behövs. Det är också den enskilde studentens sak att avgöra när och hur studierna utförs. Så länge laborationer lämnas in i tid och tentamina genomförs lägger sig ingen i på vilket sätt studenterna arbetar för att slutföra kurserna. För MT-programmets del understryks också vikten av att ta ansvar för sin egen läroprocess. Betoningen ligger dock inte så mycket på att studera när och hur man vill utan snarare på att den enskilde studenten måste ta ansvar för vad hon/ han vill och behöver lära sig. Häri ingår att ta hänsyn till de egna förutsättningarna. Samtidigt utgör projektarbeten en viktig del av studierna vid MT-programmet. Studenterna måste därför ta ansvar för såväl den egna läroprocessen som den kollektiva.

I intervjuerna framkom att samtliga aspekter av läroprocessen upplevdes som problematiska av många studenter. Vid IT-programmet uttryckte flera studenter tvivel över det lämpliga i att lära sig framförallt programmering på distans. Många saknade just möjligheten att diskutera med andra - direkt när problemen uppstod och med fysiskt närvarande människor. Vid MT-programmet uttalade flera studenter frustration över att så mycket av ansvaret las på dem själva. De önskade ett mer handfast stöd i form av (bättre) introduktioner för dem som så önskade. Dessa önskemål gällde inte bara programmeringskursen utan även flera av de andra kurserna. Projektarbetena betraktades också de som problematiska av en del studenter. Ett av skälen var att de tog mycket tid i anspråk (samtidigt som studenterna hade parallella kurser att ta hänsyn till) Ett annat skäl var gruppdeltagarnas varierande ambitionsnivå, som gjorde det svårt att skapa tillfredsställande samarbetsformer för alla i gruppen. Samtidigt bör i sammanhanget påpekas att grupperna/ konstellationerna var självvalda, vilket också innebar

möjligheter till omkonstellationer[86]. Denna möjlighet togs också tillvara av en del studenter medan andra inte tycktes se den som ett alternativ. MT-programmets projektarbeten innebar således inte några låsta eller påtvingade samarbetsformer. Inte desto mindre pekade de intervjuer jag genomförde på likartade problem som de Wistedt identifierar i sin undersökning.

4.4 Sammanfattning

I det empiriska materialet framkom att såväl studenter som lärare bär på många goda idéer om hur programmeringsundervisningen skulle kunna förändras till det bättre. I analysen resonerar jag kring svårigheter såväl som möjligheter genom att formulera tre olika teman. Dessa är:
1) Synen på kunskap
2) Formandet av kunskap
3) Vägledare i formandet av kunskap

Genom att organisera analysen på detta sätt har jag försökt lyfta fram de centrala frågeställningarna i materialet.

I avsnittet om **synen på kunskap** resonerar jag kring vikten av att synliggöra de epistemologiska utgångspunkterna för den kunskap som skapas. Genom att forskaren formulerar sin egen situering blir det tydligare vems och/ eller vilken verklighet som beskrivs. Detta förhållningssätt är särskilt framträdande inom de delar av genusforskningen som inriktar sig på att studera de vetenskapliga kunskapsprocesserna. Denna inriktning problematiserar frågor kring teori- och metodinnehåll i ett försök att öppna den "svarta låda" inom vilken vissa typer av vetande produceras. Avsnittet innehåller också en kontextuell beskrivning av de värderingar som i utbildningssammanhang premierar prestationstänkande. Slutligen tecknar jag en bild av två olika föreställningar kring genus och teknik, d v s kring hur vi som könsvarelser förhåller oss i skapandet respektive användandet av teknik.

Formandet av kunskap utgör en beskrivning av olika föreställningar och förhållningssätt kring skapandet av (programmerings)kunskap. Olika aspekter av kunskap diskuteras såsom färdighetskunskap, förtrogenhetskunskap och påståendekunskap. Den sistnämnda, teoretiska aspekten av kunskap, identifieras som den dominerande i utbildningssammanhang. Samtidigt visar jag hur programmeringskunskap i stor utsträckning handlar om praktiskt kunnande. Detta praktiska kunnande verkar dock vara det som ges minst utrymme i undervisningen. Olika koncept kring vad programmering som aktivitet kan innebära, liksom vad det innebär att lära sig programmera och vad som karaktäriserar programmeringsspråk diskuteras också. Vikten av att erkänna och ge utrymme för olika förhållningssätt till programmering avslutar detta avsnitt, bl a med en skildring av olika inlärningsstilar.

Lärarnas roll problematiseras under rubriken **Vägledare i formandet av kunskap** . Den betydelse som mänsklig vägledning har, för i synnerhet noviser i programmering, lyfts fram. Samtidigt för jag ett resonemang kring hur läraren eller handledaren kan förhålla sig, t ex genom att se sig själv som mentor. Jag diskuterar också hur lärarrollen tenderar att osynliggöras av utbildningsansvariga, genom att bara det som karakteriserar den på ett ytligt plan blir uppmärksammat. Till sist betonar jag det angelägna i att inse att även (nyblivna) lärare behöver handledning. I synnerhet i de fall där ambitionen finns att förändra undervisningen genom att pröva nya former för den.

[86] Detta påtalades särskilt av programledningen.

5. Diskussion

Nedan följer en diskussion kring några av de, enligt min uppfattning, mest centrala frågeställningarna i detta arbete. Min ambition har varit att upprätthålla ett brett perspektiv. Det empiriska materialet väckte en sådan mångfald med frågor som jag varken kunnat eller velat släppa taget om. Samtidigt inser jag vikten av att försöka synliggöra de olika trådar som kan spåras i intervjuerna. Med hjälp av andras forskning har jag formulerat ett antal teman som kan användas för att återberätta såväl som skapa nya berättelser. Dessa tematiska berättelser har i sin tur skapat utgångspunkter för nedanstående diskussion.

5.1 Att öppna den svarta lådan

Flera av de frågor jag ställer i arbetet handlar om på vilka sätt högskolestudenter ges möjligheter att öppna den "svarta låda" som omger den kunskap som skapas inom de vetenskapliga och teknologiska områdena. I vilken utsträckning avkodifieras t ex programmeringskunskap så att den blir tillgänglig för alla som vill lära sig? Det jag försöker komma åt genom att ställa en sådan fråga är om undervisningen i praktiken utformas som om studenterna fortfarande rekryterades från en begränsad "grupp", från främst de tekniska och naturvetenskapliga gymnasieutbildningarna. Finns det därtill underförstådda, men ej uttalade föreställningar om att studenterna bär med sig en omfattande datorvana i bagaget? Har högskolan lyckats anpassa sig till den alltmer differentierade sammansättningen av studenter som Gibbons m fl talar om[87]?

Vilken kunskapssyn förmedlar de utbildningar där kunskapsutvärderingen i hög grad bygger på att kontrollera att studenterna lärt sig exakt samma saker och i praktiken också på nästan samma sätt eftersom inget utrymme finns för alltför individuella inlärningsstilar och tempon? Varför finns det en sådan utbredd rädsla för "fusk", både bland studenter och lärare? Rädslan för att någon ska kopiera ens kod bygger på denna fruktan. Visst förekommer fusk i bemärkelsen att studenter plagierar arbeten från andra och detta är naturligtvis tragiskt, inte minst för studenterna själva. Problemet är när denna rädsla tar sig sådana proportioner att man inte kan skilja på denna uppenbara typ av fusk, som är lätt att genomskåda, och den inlärningsprocess som innebär att man lär sig genom att diskutera, titta på, och härma andra. Att lära sig av andras exempel tenderar att bli sanktionerat bara i de fall som exemplen inte skapats av andra studenter. Parallellt med att studenterna uppmuntras till samarbete sänder högskolan många gånger motsatta signaler genom att kräva att laborationer utförs individuellt. På så sätt hjälper högskolan till att förstärka rädslan för att det som egentligen borde ses som normalt samarbete, ska betraktas som "fusk". Kraven på att studenter individuellt ska bevisa vad de "kan", genom att lämna in likadana laborationer, verkar bygga på en kombination av föreställningar om att det ska vara "lika för alla" och farhågor om att inte kunna kontrollera att studenterna kan det de synes kunna. Frågan är bara vilken kvalitet det är som mäts när flera av de studenter som blivit godkända i den grundläggande programmeringskursen upplever att de i alla fall inte förstår! Kanske det är så att examinationsformer som bygger på att alla ska göra exakt samma laborationer under exakt samma tid och exakt samma tentamen stimulerar den teoretiska kunskapen, men på smärtsam bekostnad av färdighets- och förtrogenhetskunskaper. Kanske detta sätt att lära som bygger på principen att det ska vara "lika för alla" i själva verket uppmuntrar till "fusk", i bemärkelsen skriva av, utan att förstå. Om det saknas utrymme

[87] Se avsnitt 4.1 under rubriken Kunskapsprocesser.

för individuella inlärningsstilar och individuellt tempo eller om CSN stryper studiemedlen ifall man inte blir godkänd, är det kanske inte så konstigt att det blir viktigare att "klara laborationen" än att förstå den. För de lärare som har som sin uppgift att "förmedla" programmeringskunskap blir dessutom kraven på "likhet" en snara som medför att inget reellt utrymme ges för att pröva nya former i undervisningen. Risken att de då inte hinner med det de "ska" får ju den oerhörda konsekvensen att studenterna inte får med sig de förkunskaper de förväntas ha för att klara nästa kurs.

Vid de två utbildningar jag har tittat på märks ovanstående tendenser i olika omfattning. IT-programmet har kravet att laborationer ska redovisas individuellt och rädslan för "fusk" är mer uttalad än vid MT-programmet. Framförallt är denna rädsla mer eller mindre gemensam för de studenter och lärare från IT-programmet som jag intervjuat. Lärarna vid MT-programmet accepterar naturligtvis inte heller fusk i bemärkelsen plagiering, men de - och i synnerhet programledningen - har på ett medvetet sätt försökt arbeta bort den attityd som sammanblandar denna typ av uppenbart fusk med de former av samarbete som borde anses som ett naturligt inslag i läroprocessen. För att åstadkomma detta har de uppmuntrat studenterna att hjälpa varandra och ta tillvara den kunskap som finns samlad i gruppen. Studenterna har också fått välja i vilka konstellationer de vill samarbeta. Framförallt har de uppmanats att utgå från sin egen läroprocess. En viktig del i detta är de examinationsformer som bygger på att skapa något utifrån den egna kunskapsnivån och att sedan förklara det man gjort, hur och varför. Inte desto mindre ger en del av studenterna uttryck för farhågor om att andra "fuskar" eller slipper "lindrigt" undan. Ett fall av direkt plagiering upptäcktes också i programmeringskursen. Plagieringen var gjord direkt från kursboken. De ansvariga lärarna godkände naturligtvis inte de studenter som låg bakom plagieringen, men de gav dem chansen att utföra ett nytt projekt. Allt detta menar jag visar på att det är en lång och inte helt okomplicerad process att "lära bort" de föreställningar som många av oss, både studenter och lärare, bär på. Föreställningar som i grunden handlar om att utbildning går ut på att "prestera" inte på att "förstå".

Jag påstår inte att man ska slopa all form av kvalitetskontroll. Istället menar jag att det är nödvändigt att höja kvaliteten på denna "kontroll". Jag tror jag det är nödvändigt att kontinuerligt och tillsammans, utvärdera de enskilda studenternas (och lärarnas) utveckling såväl som utformningen på kurserna. Att utvärdera enskilda studenters utveckling innebär dock att utgå från varje enskilt fall. Precis som man vid ett utvecklingssamtal på en arbetsplats utgår från individens förutsättningar och ambitionsnivå (i förhållande till det uppdrag företaget eller organisationen i fråga har) borde man kunna göra det i studiesammanhang, på högskolenivå.

För att öppna den "svarta låda" som omger bl a programmeringskunskap, krävs det insikter om att kunskap inte är något som skapas genom en rätlinjig process. Det krävs också insikter om att vi som enskilda deltagare i en läroprocess bär med oss olika erfarenheter och förkunskaper. Att skapa en undervisning som är "lika för alla" förfelar därmed sitt syfte - att skapa jämlika förutsättningar. De enda som ges några egentliga förutsättningar är de som klarar av att anpassa sig till de uttalade och outtalade krav som finns och/ eller som har sådana förkunskaper och erfarenheter att de klarar sig ändå. För alla andra blir utbildandet en ständig kamp mot olika paradoxer. Om dessa insikter tas tillvara kan möjligheter skapas för en utbildning som utgår från den individuella läroprocessen, men med den kollektiva som omgivande "membran". Det innebär en process där stort utrymme för mångfald ges och där vars och ens *förståelse för* istället för *prestation i* ämnet är det "resultat" som eftersträvas.

5.2 Hur ger man utrymme för mångfald i skapandet av kunskap?

För att skapa utrymme för mångfald krävs vilja till mångfald. Det måste finnas en kunskapssyn som i grunden utgår från att alla som vill kan lära sig om de bara ges förutsättningar till det. Det är därför viktigt att inse att olika studenter har olika erfarenheter med sig, men också olika mål för vad de hoppas få ut av utbildningen. Samtidigt är det angeläget att komma ihåg att kunskap är något som ständigt formas och omformas utifrån det eller de sammanhang som den skapas i. En aspekt av det påståendet är att motivationen att lära sig varierar i förhållande till vad man kan i övrigt.

Att lära sig javaprogrammering ses kanske inte som något självklart lustfyllt för den som inte har konfronterats med situationer där det uppenbart vore bra att kunna det. - Hur många har gjort det under den första terminen av utbildningen? Naturligtvis söker man sig inte till en utbildning om man inte upplever att den tilltalar de intressen man har (just nu eller sedan flera år tillbaka). Av intervjuerna framgår dock att många har en ganska diffus bild av vad utbildningen ska leda till för deras del. Denna bild blir i regel klarare med tiden. Utvecklingen av den sker i förhållande till hur studenten upplevt de olika inslagen i utbildningen. En av utmaningarna för lärare är därför att väcka det presumtiva intresset hos studenterna genom att visa på möjliga tillämpningsområden för de olika delmomenten.

Skapandet av kollektiv kunskap

Vi själva, studenter och lärare, är ytterst delaktiga i den process som skapandet av kunskap innebär. Det är därför viktigt att ständigt reflektera över hur denna process ser ut.

I samband med att studenterna vid MT-programmet redovisade sina projekt slogs jag av att det fanns en tydlig tendens att de manliga studenterna arbetade för sig och de kvinnliga för sig. Endast i ett par grupper arbetade män och kvinnor tillsammans. I de grupper som bestod av enbart män hade många skapat väldig likartade program. Svårighetsgraden liksom finessen varierade, men i princip alla hade gjort spel som gick ut på att samla poäng genom att träffa ett eller flera föremål. Med tanke på att de kursansvariga vid programmeringskursen är oerhört spelintresserade och med tanke på att en hel del av de manliga studenterna (men få av de kvinnliga) ägnar eller har ägnat sig åt datorspel på sin fritid är det kanske inte så konstigt. Det verkar som att männen inspirerat varandra i hög utsträckning och en del av de män som haft svårigheter med programmeringen har uttalat uppskattning över att det alltid har funnits någon att fråga. Bland de kvinnliga studenterna och i de blandade grupperna var variationen större vad gällde typen av program man skapat. Svårighetsgrad och finess varierade också här, men framförallt uppfattade jag att betydligt fler i den här gruppen hade uttalade problem med programmeringen. Alla blev heller inte godkända i samband med examinationen. En fråga jag ställde mig (och fortfarande ställer mig) var om det kan vara så att de manliga studenterna skapar en kollektiv kunskap som inte riktigt kommer de kvinnliga studenterna till del? Jag vill för säkerhets skull inflika att det fanns grupper med enbart kvinnliga studenter såväl som blandade grupper som godkändes utan tvekan. Vad jag försöker komma åt är om det finns en tendens att de manliga studenterna lär av varandra och de kvinnliga av varandra. Vad innebär det i såfall mot bakgrund av männens (som kollektiv betraktade) betydligt större erfarenheter av datorer och programmering?

Känslan av att en kollektiv kunskap skapades bland de manliga studenterna vid MT-programmet förstärktes av att situationen vid IT-programmet var så annorlunda. Här gjorde alla exakt samma laborationer och alla arbetade i princip på egen hand (uppgifterna skulle

redovisas individuellt). Vid IT-programmet var det också tydligt att de flesta hade mer eller mindre stora problem, *oavsett* tidigare kunskaper. Visserligen skapades en del lokala studiegrupper och de som hade kamrater att fråga utanför utbildningen använde sig naturligtvis av denna möjlighet. Den kollektiva kunskap som byggdes upp var dock varken lika uttalad som hos studenterna vid MT-programmet eller lika begränsad som den tycktes vara där - till framförallt de manliga studenterna. IT-programmet framstod i det perspektivet som mer "jämlikt", samtidigt som studenterna där naturligtvis hade det svårare på många sätt eftersom de var så mycket mer utlämnade till sig själva. Idealet vore givetvis om det gick att skapa en atmosfär som byggde upp den kollektiva kunskapen på ett sätt så att den kunde komma *alla* till del. Kanske är det bara en fråga om tid vad gäller MT-programmet för att de, ytterst subtila - men likväl existerande, könsmässiga barriärerna ska raseras. Ingen av de manliga studenterna stänger naturligtvis medvetet ute någon av de kvinnliga. Det handlar mer om djupt rotade kulturella mönster kring kvinnlighet och manlighet som måste brytas. För att bryta dessa på ett tidigt stadium krävs dock en medvetenhet kring att de existerar och ett målmedvetet arbete för att förändra dem.

För IT-programmets del ligger frågan på en annan nivå. Där handlar det snarare om att skapa strategier för hur man bygger upp och tillvaratar kollektiv kunskap över huvud taget. De studenter som deltagit vid de föreläsnings - och lektionstillfällen som varit i Ronneby respektive Höganäs tycks i viss mån skapat embryot till detta. De diskuterar mycket när de träffas och av de som deltagit vid dessa träffar har alla uttryckt den stora betydelsen av att träffas på detta sätt - både vad gäller att bli allmänt "peppad" som att få direkt hjälp med laborationer. Utmaningen för IT-programmet ligger i att bygga vidare på detta embryo och skapa förutsättningar för kollektivt lärande även bland dem som aldrig har möjlighet att resa till högskolan. Förmodligen krävs nya strategier vad gäller skapandet av lokala studiegrupper såväl som mer välutvecklade virtuella gemenskaper. Sannolikt krävs också en annan attityd till samarbete i form av att öppet visa och diskutera kod med varandra. Rädslan för att någon ska kopiera koden i kombination med en allmän osäkerhet hos många studenter har inneburit att ytterst få skickar in några längre kodsnuttar till det virtuella klassrum som finns. Problemet är att det är ytterst svårt att diskutera kring något som man inte kan se.

Långsamhetens betydelse

I intervjuerna framkom att många studenter vid båda programmen upplevde att undervisningen gick fram med alldeles för stora steg. De efterlyste fler små exempel och övningar med anknytning till de teoretiska genomgångarna. Flera av lärarna visade sig också vara medvetna om att introduktionen till javakursen inte varit vad den borde. Av intervjuerna framgick att en del av problemen hade gått att undvika om lärarna anställts på ett tidigare stadium. Samtidigt är frågan i vilken utsträckning lärarna verkligen "ser" den tid det tar att lära sig programmering för dem som ingenting kan. Ett par av dem påtalade just hur lätt det är att bli blind för de svårigheter det kan innebära och den tid det tar för novisen att lära sig programmera.

Vilken betydelse får "likhetstanken" för den tidsmässiga inramningen av studierna? Om lärare uppfattar det som ett outtalat krav att det ska vara "lika för alla", vilket utrymme ges de då att skapa en differentierad undervisning? Finns det kanske en risk att lärare med den vällovliga ambitionen att ta hänsyn till "alla", d v s både de som kan lite programmering sen tidigare och de som inte kan någonting, tvingas lägga sig på en nivå där ingen blir riktigt nöjd? Studenter med tidigare kunskaper i programmering blir understimulerade vilket får dem att klaga över att det är för "lätt". De som saknar kunskaper känner sig ännu dummare eftersom de måste

brottas både med känslan av att inte förstå och känslan av att de borde förstå, eftersom andra säger att det är lätt.

De kursansvarigas ambition med de projekt som MT-studenterna genomförde var att var och en skulle kunna välja ett projekt på "sin nivå". Genom att ha en snabb introduktion lämnade lärarna utrymme för en mer differentierad undervisning resten av terminen. I viss mån har denna uppläggning fyllt sitt syfte. De flesta studenterna var väldigt nöjda med att få arbeta i projektform. Flera upplevde att det de lärt sig i programmering var det de lärde sig under projektet. Samtidigt var många frustrerade - de upplevde att de hade behövt en bättre grund för att förstå vad de gjorde. Några lyckades heller inte genomföra sitt projekt. Andra genomförde det, men upplevde att resultatet av projektet i praktiken hade gått före förståelsen av det. Dessa studenter hade förmodligen varit mindre frustrerade om introduktionen gjorts i ett långsammare tempo.

Frågan hur kursansvariga kan hantera det faktum att studenterna har så olika förkunskaper kvarstår dock. Hur bär man sig åt för att motivera alla studenter? Tanken med individuellt anpassade projekt är förmodligen en bra utgångspunkt. De kursansvariga har dock ett ansvar för att noviserna får allt bistånd de behöver för att skapa en grundläggande förståelse för programmeringen. Det tar tid att skapa en grundläggande förståelse för något som man aldrig hållit på med tidigare, något som utgör en ny begreppsvärld - och det måste få ta tid. Därtill måste det få ta olika lång tid för olika människor. Med tanke på att studietiden är begränsad blir därför högskolans viktigaste uppgift att se till att alla studenter lyckas tillägna sig en "grund" att utgå från. Det fortsatta lärandet - hur mycket man lär sig och hur duktig man blir är en högst individuell process. Har man en stabil "grund" att utgå från kan man fortsätta utvecklas i denna process. Kanske är begreppet "grund" dessutom missvisande. Egentligen handlar det kanske om att tillägna sig "lusten att lära". Det är däremot ytterst tveksamt om en alltför stor fixering vid att alla ska utföra likadana laborationer och svara på samma frågor i en tentamen utgör de bästa förutsättningarna för att bygga upp denna "lust". I synnerhet med tanke på att programmering i så stor utsträckning handlar om praktik. Det innebär att var och en, studenter såväl som lärare, måste hitta sitt eget förhållningssätt, sin egen förståelse. Att prestera och förstå är inte nödvändigtvis samma sak.

5.3 De hälften synliga - hur formas och formar "förmedlarna" av kunskap?

Bland de lärare och handledare jag har intervjuat finns både de som lärt sig programmera under sin högskoletid och de som kunde programmera innan (dock inte nödvändigtvis i java). För en del är programmeringen lika mycket en hobby som ett studieämne. För andra är programmering i första hand ett studieämne i vilket man bör tillägna sig kunskaper för den fortsatta utbildningens skull och i förlängningen för det framtida yrket. I arbetet med denna rapport har jag uppfattat att förhållningssätten till de läroprocesser som omger programmeringskunskap ser något annorlunda ut beroende på vilken av "kategorierna" en lärare tillhör.

Hur är det då att komma ny som lärare i programmering? Vilket stöd får de som är nya? Vilka signaler får de från högskolan vad gäller övergripande kunskapssyn? Vad anses från högskolans sida som viktigt att förmedla?

De lärare och handledare som kommer in i undervisningen tack vare sitt stora intresse och i regel också långa erfarenhet av programmering, besitter oftast en oerhörd förtrogenhetskun-

skap vad gäller programmering som aktivitet. Ett problem menar jag kan vara att de inte alltid har verbaliserat denna kunskap. När de hamnar i en undervisningssituation förmedlar de därför sina kunskaper på ett sätt som de tror är riktigt (eller som högskolan förmedlar som riktigt), vilket alltför ofta är det formella sättet. De förmedlar den teoretiska kunskapen, men glömmer bort att tala om hur man gör. Detta i kombination med att de själva behärskar programmeringen så väl bidrar till att de ofta går alldeles för fort fram. Det faktum att dessa personer i regel tycks fungera bättre i den individuella handledningssituationen tyder också på detta, eftersom de då på ett helt annat sätt kan relatera till sina egna erfarenheter och ges en möjlighet att klä dessa i ord. De behöver inte känna sig pressade av att formulera förklaringar som ska fungera för en mångfald människor i en större grupp utan de kan försöka möta respektive individ på dennes nivå genom att föra en dialog. Det innebär också att de lättare kan frigöra sig från "krav" som lätt kan uppstå i en grupp på att saker och ting måste presenteras på ett visst sätt eller på att det inte får vara för "lätt".

De personer som kommer in i förmedlarrollen utan att ha programmering som hobby, men med tillräckliga kunskaper formellt sett, tenderar att lära ut programmering på det sätt som de själva lärt sig. Om de inte varit nöjda med det sättet försöker de naturligtvis genomföra förändringar, men de tycks ha svårt att göra dessa särskilt radikala. Förmodligen innebär avsaknaden av andra referensramar till programmering, än de dessa lärare förvärvat genom sin egen utbildning, att det är svårt att hitta nya infallsvinklar. När de dessutom varken ges något tidsmässigt utrymme eller någon handledning från mer erfarna lärare är förutsättningarna för att förändra undervisningen inte särskilt goda. Den största styrkan hos dessa lärare ligger kanske i att deras egna erfarenheter av att lära sig programmera ligger så pass nära i tiden att de bättre än många andra kan sätta sig in i de svårigheter nybörjare brottas med. Risken att de ska gå för fort fram är långt ifrån lika stor som hos de lärare som har programmering som hobby.

De mer erfarna lärarna i programmering som jag intervjuat, tycks ha hittat strategier för att anpassa undervisningen till de individer som ingår i respektive grupp. De arbetar också kontinuerligt för att utveckla undervisningen. Samtidigt är bristen på erfarna pedagoger i programmering så omfattande att det därför är oerhört angeläget att rekrytera nya. När bristen är så stor att man tvingas låta de studenter som egentligen skulle fungera som labbhandledare ta betydande delar av eller hela kursansvaret, är situationen alarmerande. Jag menar inte att dessa personer inte skulle ha förutsättningar att bli bra pedagoger - tvärtom. Däremot menar jag att det är ytterst tveksamt om de ges några förutsättningar att bli det. Frågan är vilka signaler högskolan i vissa fall sänder till dem som - i rollen som "förmedlare" av programmeringskunskap - är noviser. De som vittnar om bristande eller obefintlig uppbackning från mer erfarna kollegor eller att det tidsmässiga utrymmet för att utforma undervisningen varit minimalt kan inte känna sig särskilt lockade av att satsa på en karriär som lärare efter avslutad utbildning.

Högskolan lär förmodligen aldrig kunna konkurrera med näringslivet vad gäller den ekonomiska ersättningen till dem som anställs. Däremot skulle högskolan kunna attrahera potentiella pedagoger genom att skapa ett klimat som, i betydligt större utsträckning än idag, gav utrymme för ett kreativt och livslångt lärande.

5.4 Slutord

Vid BTH borde förutsättningarna för att skapa ett "lärande klimat" vara goda. Högskolan är förhållandevis liten och framförallt inte tyngd av traditioner om hur utbildningar bör vara. Samtidigt upplever jag att utbildningsansvariga vid BTH inte i tillräcklig utsträckning artikulerat och diskuterat den övergripande kunskapssyn utifrån vilken utbildningarna formas. Diskussioner förs på sina håll, men förefaller vara begränsade till vissa institutioner och utbildningar. I alltför många fall verkar kurser planeras i ett vakuumliknande tillstånd. Var förs samtal om hur utbildningarna kan bidra till skapandet av ett helhetsperspektiv t ex genom att kurser integreras med varandra? Var reflekteras de problem som många studenter uppenbarligen upplever, bl a med programmering? I vilken utsträckning har utbildningsansvariga vid BTH gjort upp med den kunskapssyn som innebär att prestation går före förståelse?

De två program jag tittat på för denna undersökning står inför utmaningar som har både likheter och olikheter. Vid IT-programmet finns en medveten strävan att skapa studiemöjligheter för nya grupper av studenter genom att göra det möjligt att läsa stora delar av, eller t om hela, utbildningen på distans. Den största utmaningen för IT-programmet är att skapa förutsättningar för att den kunskap som förmedlas inte alltför ensidigt inriktas på teoretiskt kunnande. I synnerhet är detta viktigt i ett ämne som programmering som förutsätter en kombination av problemlösningsförmåga och hantverksmässighet. En potentiell möjlighet är tillvaratagandet av den kollektiva kunskap som finns i gruppen. För att genomföra ett sådant tillvaratagande krävs dock andra former för, och sannolikt också attityder till, samarbete än vad som är fallet i dag. Bättre utnyttjande av gamla och nya former av virtuella gemenskaper, utvecklingsverktyg, självstudietest mm är några av de konkreta förslag som intervjuade studenter och lärare bidragit med. Lokala studiegrupper, som ges aktivt stöd från högskolans sida (Varför kan inte någon gång lärarna resa till studenterna? Måste det alltid vara studenterna som reser till lärarna?), är ett annat.

Vid MT-programmet har programledningen kommit långt vad gäller att artikulera den övergripande kunskapssynen och strategier för hur man kan skapa förutsättningar till "lärande" för alla studenter. En av dessa strategier är att uppmuntra till att var och en tar ansvar för den egna läroprocessen. Studenternas egna förutsättningar, erfarenheter och ambitioner bildar utgångspunkter för vars och ens läroprocess, som sker inom ramen för den kollektiva. För att motverka konkurrens- och prestationstänkande görs också försök med andra examinationsformer än de traditionella. Programledningen eftersträvar dessutom att skapa utrymme för kontinuerliga diskussioner och reflektioner kring läroprocessen(erna) inom de olika ämnena. Vid intervjuerna framkom att de problem som lärare och framförallt studenter vid MT-programmet brottas med vad gäller programmering delvis bär på stora likheter med de problem som IT-studenterna brottas med. Den bild MT-studenterna förmedlade av vilka svårigheterna är med (undervisningen i) programmering är dock betydligt mer differentierad än den bild IT-studenterna förmedlade. De största generella skillnaderna mellan programmen utgörs av olika attityder till att ha lärt sig "tillräckligt" och av den/ de betydelser programmeringen har för utbildningen i övrigt. MT-studenterna tycktes i allmänhet vara mer införstådda än IT-studenterna med att formandet av programmeringskunskap måste betraktas i ett långsiktigt perspektiv. De hade heller inte samma föreställningar om vad man "bör" ha lärt sig utan verkade ha en större benägenhet att se till vad de lärt sig i jämförelse med vad de kunde sedan tidigare. Samtidigt uttalade många både oro och frustration över att inte förstå och/ eller "hänga med i undervisningen". Detta tyder på att förändringar återstår att göra (t ex vad gäller tempot, men också strategier för att hantera den variation av

förkunskaper som varit ytterst märkbar bland MT-studenterna). Det tyder också på att förändring är något som tar tid och att en termin egentligen är alldeles för kort tid för att utvärdera resultaten; i synnerhet som förändringarna måste ske på flera plan. Såväl de inblandade lärarna som de deltagande studenterna måste komma till insikt om och kanske "göra upp" med en del av de föreställningar de bär med sig om vad det innebär att utbilda sig i allmänhet och lära sig programmera i synnerhet. Den största utmaningen för MT-programmet är kanske att inte tappa taget om enskilda studenter (eller lärare) i denna förändringsprocess. Det faktum att programansvarig anstränger sig för att i så stor utsträckning som möjligt vara tillgänglig för programmets olika deltagare är betydelsefullt och har tydligt bidragit till att en hel del svårigheter kunnat lösas på ett tidigt stadium. Samtidigt är det viktigt att vara medveten om hur svårt det är att bryta hierarkier. Det är inte alltid de mest demokratiska föreställningarna som vinner gehör i en miljö där ansvaret ligger hos deltagarna. Den som har problem med programmering blir heller inte nödvändigtvis hjälpt av att det är högt i tak för diskussioner. Snarare behöver den personen ganska handfast stöd med att skapa förståelse för programmerandets principer.

Att förändra programmeringskurserna (eller utbildning över huvud taget) är en process som jag menar framförallt har att göra med vilken den grundläggande synen på kunskap är inom det eller de sammanhang där utbildningarna formas. Att diskutera de enskilda detaljerna i innehållet är absolut inte oviktigt, men sätts inte detaljerna in i ett större epistemologiskt sammanhang så blir det som att försöka bota symptomen istället för att hitta orsakerna. Som utbildningsansvarig, lärare och student måste jag tro på möjligheten att alla kan lära sig. I denna process har vi som deltagare olika roller. Som student måste jag ta ansvar för mitt eget lärande, lägga ner den tid jag behöver, men också sätta upp realistiska mål för vad jag vill genomföra. Som lärare är det viktigt att fundera kring den egna inlärningsprocessen för att därigenom lättare kunna möta och ta tillvara alla de individuella stilar och ambitionsnivåer som studenterna bär med sig. Den som är utbildningsansvarig måste se till att förutsättningar skapas för att såväl lärare som studenter ges utrymme för ett livslångt lärande.

Referenser:

Adam, Alison: Artificial intelligence and women´s knowledge *i Women´s Studies International Forum*, vol 18, no 4, 1995, s. 407-415.

Björkman, Christina: *Projekt Q+. Med och för kvinnliga studenter i datavetenskap*. Uppsala Universitet. Enheten för utveckling och utvärdering. Arbetsrapport nr 6, januari 2000. ISSN 1403-3968.

Björkman, Christina: Varför väljer kvinnor inte datavetenskap och hur stöttar vi dem som gör det? Manuskript från föreläsning som hölls vid konferensen *Kvinnor och matematik 4*, Uppsala, april 1999.

Booth, Shirley: *Learning to program. A phenomenographic perspective*. Acta Universitatis Gothoburgensis, Göteborg, 1992. ISBN 91-7346-256-X

Carlshamre, Pär & Mårdsjö, Karin: *Retoriken kring tekniken*. Studentlitteratur, Lund, 2000. ISBN 91-44-01203-9.

Checkland, Peter: *Systems thinking, systems practice ; Soft systems methodology : a 30-year Retrospective,* Chichester, Wiley, 1998. ISBN 0-471-98606-2.

Datateknisk ingång för kvinnor. En utvärdering. Inger Wistedt (red). Luleå tekniska universitet. Teknisk rapport nr 2000:15. ISSN 1402 - 1536.

Ehrlich, Kate & Cash, Debra: The Invisible World of Intermediaries. A Cautionary Tale. In *Computer Supported Cooperative Work* nr 8, s 147-167. Kluwer Academic Publishers, 1999.

Eneroth, Bo: *Att handla på känn*. Om intuition i professionell verksamhet. Natur och kultur, Stockholm 1992. ISBN 91-27-02381-8.

Eriksson, Hans-Erik: *Programutveckling med java*. Studentlitteratur, Lund, 1997. ISBN 91-44-00219-X.

Från siffror till surfning: Könsperspektiv på informationsteknik. Lena Kallin & Lena Palmquist (red). Umeå Universitet. Institutionen för datavetenskap, Umeå 1999. ISBN 91-7191-598-2.

Från symaskin till cyborg. Elisabeth Sundin & Boel Berner (red). Nerenius & Santérus förlag, Stockholm, 1996. ISBN 91-88384-90-X.

Gendered by Design?: Information Technology and Office Systems. Eileen Green, Jenny Owen, Den Pain (ed.) Taylor & Francis, London, 1993. ISBN 0 748 400 915 alt 923.

Gibbons, Michael, m fl: *The New Production of Knowledge*: The Dynamics of Science and research in Contemporary Societies. SAGE Publications Ltd, London, 1994. ISBN 0-8039-7793-X, 0-80397794-8 (pbk)

Gulbrandsen, Elisabeth: *The reality of our fictions - notes towards accountability in (techno)science.* Luleå tekniska universitet 1995:20.

Göranzon, Bo: *Det praktiska intellektet. Datoranvändning och yrkeskunnande.* Carlsson Bokförlag, Stockholm, 1990, ISBN 91 7203 1123.

Haraway, Donna: Ett manifest för cyborger. Vetenskap, teknologi och socialistisk feminism under 1980-talet. Publicerad i *Samtidskultur - karaoke, karnevaler och kulturella koder*. Thomas Johansson, Ove Sernhede, Mats Trondman (red). Nya Doxa, Nora 1999. ISBN 91-578-0056-1.

Haraway, Donna: ”Situated Knowledges: The Science Question in feminism and the Privilege of Partial Perspective”, *Simians, Cyborgs and Women – The reinvention of nature*, Routledge New York, 1991.

Harding, Sandra: Is there a feminist method? Introduction of *Feminism and Methodology: Social Science Issues.* Bloomington: Indiana University Press, 1987.

Janik, Allan: *Cordelias tystnad*. Om reflektionens kunskapsteori. Carlsson Bokförlag, Malmö, 1991. ISBN 91 7798 499 4, s 106-141.

Janik Allan: *Kunskapsbegreppet i praktisk filosofi*. Bruno Östlings Bokförlag Symposium. Stockholm/ Stehag, 1996. ISBN 91-7139-319-6, s 13-54.

Kaschak, Ellyn: *Engendered lives*. A new psychology of women´s experience. Basic Books, New York, 1992. ISBN 0-465-01347-3.

Lewis, John & Loftus, William: *Java Software Solutions*. Foundations of Program Design. Addison-Wesley, 1998. ISBN 0-201-57164-1.

Merriam, Sharan B: *Fallstudien som forskningsmetod*. Studentlitteratur, Lund, 1994. ISBN 91-44-39071-8.

McCarthy, Bernice: Using the 4MAT System to bring learning styles to schools. Artikel publicerad i: *Educational leadership* / journal of the Association for supervision and curriculum development. 1990, October, 48(2), s 31-37.

Mörtberg, Christina: *Det beror på att man är kvinna….* Gränsvandrerskor formas och formar informationsteknologi. Luleå tekniska universitet, Institutionen för Arbetsvetenskap - avdelningen Genus och Teknik, Luleå, 1997.

Mörtberg, Christina: *Teknikvetenskap och genusforskning eller Creating Better Worlds Through Imaginations.* Bilaga 5 (TFR:s ansvarsområde) i Trojer et al: Genusforskningens relevans. Se separat referens.

Papert, Seymour & Turkle, Sherry: Epistemological pluralism: styles and voices within the computer culture i *Signs/ journal of women in culture and society*, Chicago, Autumn 1990.

Patel, Runa & Davidson, Bo: *Forskningsmetodikens grunder*: Att planera, genomföra och rapportera en undersökning. Stockholm, 1998. ISBN 91-44-30952-X.

Precision och improvisation: Om systemutvecklarens yrkeskunnande. Christer Hoberg, red. Combitech Software och Dialoger, Stockholm 1998. Ingår i serien Filosofi och ingenjörs-arbete.

Salminen-Karlsson, Minna: *Bringing Women into Computer Enigneering*. Curriculum Reform Processes at Two Institutes of Technology. PhD Thesis, Department of Education and Psycology, University of Linköping, 1999

Selling Sjöberg, Astrid & Tjernström, Mattias: *Lära sig leva i den grå zonen* . En studie om användande av programmeringsverktyg för utveckling och lärande. Kandidatarbete 20 poäng, MDA 1999.

Stanley, Liz & Wise, Sue: Method, methodology and epistemology in feminist research processes i *Feminist Praxis,* London: Routledge, 1994.

Star, Susan Leigh & Strauss, Anselm: Layers of Silence, Arenas of Voice: The Ecology of Visible and Invisible Work. In *Computer Supported Cooperative Work* nr 8, s 9-30. Kluwer Academic Publishers, 1999

Trojer, Lena, Gender research - an interdisciplinary challenge. Manuskript från föreläsning som hölls vid konferensen *Theory and practice of interdisciplinary work*, MISTRA och FRN, Stockholm, 1998.

Trojer et al: *Genusforskningens relevans*. Slutrapport från integreringsarbete i åtta svenska forskningsråd. Rapport från forskningsrådens expertgrupp för genusforskningens integrering under samverkansgruppen för tvärvetenskap, genusforskning och jämställdhet., Stockholm, 2000.

Wistedt, Inger: *Recruiting Female Students to Higher Education in Mathematics, Physics and Technology,* An evaluation of a Swedish Initiative. Högskoleverket, 1998.

Women in computing, edited by Rachel Lander & Alison Adam: Intellect Books, Exeter, 1997, ISBN 1-871516-58-7

Bilagor

Bilaga 1: Liten ordlista

Nedanstående ordlista är tänkt som ett stöd vid läsningen av rapporten. Den bör därför inte ses som någon fullständig beskrivning av grundläggande begrepp, varken i java eller något annat programspråk.

Abstrakt klass En klass som används i syfte att fungera som basklass (superklass) från vilken andra klasser kan ärva. Innehåller en klass en eller flera abstrakta metoder måste den defineras som abstrakt. En abstrakt klass kan inte användas för att skapa objekt.

Abstrakt metod I en metod som defineras som abstrakt ges ingen implementation, d v s regler för hur metoden ska tillämpas. Det enda som anges är metodens huvud eller signatur (namn, returtyp och parametrar). Den/ de klasser som ärver den abstrakta klass som innehåller en viss/ vissa abstrakta metoder måste definiera dessa metoder och hur de ska implementeras.

API Applications Programming Interface. API:er är de paket som innehåller mjukvara i form av standardiserade klasser och gränssnitt. Som programmerare skapar man också egna API:er. Exempel på ett standard API är java.awt (awt står för abstract windowing toolkit) som bl a innehåller klasser för fönster, knappar och grafik.

Applet En applet är ett program som kräver en webbläsare alternativt ett program i form av en s k appletviewer för att kunna exekveras/ köras.

Applikation Avser egentligen applikationsprogram som är ett fristående program.

Arv Inom objektorienterad programmering är användningen av arv grundläggande, eftersom just möjligheten att ärva egenskaper från en klass till en annan utgör grunden för den struktur som möjliggör återanvändning av kod, vilket är den objektorienterade programmeringens främsta styrka. Exempel på en egenskap som kan ärvas är t ex titel som är ett attribut som förekommer i alla typer av medier. Attributet titel kan därför definieras i bas- eller superklassen Media och sedan ärvas av underliggande subklasser som Bok, Videofilm, etc.

Attribut Används i programspråket java synonymt med variabel. Fungerar som värdehållare för t ex ett tal eller en text. Det är m a o en plats i datorns minne där man kan lagra information. Exempel på ett attribut är " String title" (anger att det rör sig om ett attribut av typen String som innehåller en text i form av en titel.)

Editor Ett enkelt program, t ex Anteckningar (Notepad) som i programmeringssammanhang används för att skriva in kod.

Exekvera Att exekvera koden innebär att man kör programmet.

Extends Är ett s k reserverat ord som används i samband med arv. För att markera arv importerar man en klass genom att skriva " public class

	NamnetPåKlassenSomÄrver (subklassen) *extends* NamnetPåKlassenManÄrverFrån (superklassen)"
Felhantering	Som programmerare måste man kunna hantera olika typer av fel - dels "förväntade" fel (som att en användare matar/ skriver in felaktiga värden), dels "oväntade" fel t ex nätverksfel, I/O-fel (in/utmatning) eller interna programmeringsfel.
GUI	Graphical User Interface. Grafiskt AnvändarGränssnitt
Inkapsling	Innebär i korthet att ett objekt betraktas som en "svart låda", d v s de objekt som kommunicerar med varandra bryr sig inte om den interna strukturen hos varandra utan bara om de tjänster som utförs mellan objekten. Detta innebär att objekten är relativt skyddade eftersom den information som används internt inom objektet inte går att komma åt utan vidare.
Instansiera	När objekt skapas från en klass kallas detta för instanisering. Det skapade objektet utgör en instans eller ett exempel av klassen.
JDK, jdk	Java Development Kit. Utvecklingsmiljö för java. Innehåller en javakompilator, en virtuell maskin och samtliga standardbibliotek (eg. paket) för java. Se API.
Klass	En klass representerar en viss objekttyp (t ex objekt av typen Bok) och kan sägas utgöra den mall utifrån vilken nya objekt skapas.
Kompilera	För att datorn ska kunna "läsa" javakoden (källkoden) måste den först översättas till bytekod i en javakompilator. Bytekoden måste sedan antingen tolkas av en javatolkare (i form av en virtuell maskin) eller översättas till maskinkod som förstås av den aktuella datorn.
Konstruktor	En konstruktor är en speciell metod som varje java-klass innehåller och som används för att skapa instanser av klassen.
Main-metoden	Main-metoden är den metod som gör att programmet går att starta.
Metod	En metod är en funktion som utför något t ex " setTitle" som ändrar titeln på ett objekt. Metoder beskriver m a o de tjänster som en klass tillhandahåller.
Paket	Paketbegreppet används inom java för att gruppera ihop klasser.
Parameter	En parameter är ett värde som skickas till en metod när det anropas.
Polymorfism	Kallas ibland för dynamisk bindning, vilket egentligen är tekniken för att uppnå polymorfism, vilket innebär att en och samma metod kan ha många olika implementationer eller tillämpningar i olika klasser. Vilken metod som anropas avgörs i själva exekveringsögonblicket (när koden körs) av vilken klass ett objekt tillhör . Den generella metoden "rita" som definieras i en basklass med namnet "Figur" ritar i subklassen "Cirkel" ut en cirkel, medan den i subklassen "Rektangel" ritar ut just en rektangel.
Private	En metod eller variabel som deklareras som privat kan enbart nås inom den klassdefinition i vilken den är definierad. Attribut är normalt privata eftersom de inte bör vara synliga utåt enligt principen om inkapsling.

Protected Variabler och metoder som deklareras som "skyddade" nås infrån den klass-definition inom vilken de är deklarerade och från klasser som ärver från denna. De är även tillgängliga för andra metoder inom samma paket som den klass de tillhör ligger i.

Public En publik klass, variabel eller metod nås såväl inifrån som utifrån den klass-definition inom vilken den är deklarerad. Den kan alltså nås även av klasser i andra paket.

Rekursion En rekursiv algoritm anropar sig själv för att lösa ett delproblem med vissa egenskaper, som t ex att varje delproblem är mindre än det ursprungliga problemet. Syftet med rekursion är att dela upp större problem så att dessa kan lösas genom att kombinera lösningarna på delproblemen.

Returtyp Anger av vilken typ ett returvärde ska vara. Ett returvärde kan t ex vara en siffra. Om man vet att metoden enbart ska hantera heltal väljs lämpligen en int (integer) som returtyp. Om metoden ska hantera text väljs String som returtyp. Om inget värde ska returneras anges returtypen till " void", som t ex i main-metoden. Omvänt kan man säga att ett returvärdes returtyp måste korrespondera med returtypen som anges i metodhuvudet.

Returvärde Det som en metod returnerar t ex ett namn, en titel, ett årtal etc.

Sträng Engelskans String, är egentligen en klass som finns fördefinierad i den del av API:et som medföljer JDK. I java är text- eller teckensträngar egentligen objekt av typen String.

Subklass En subklass ärver egenskaper från en klass som har mer generella egenskaper och därmed kan sägas ligga högre upp i arvshierarkin.

Superklass Superklass är en s k basklass från vilken andra klasser kan ärva. Klassen Media utgör t ex en superklass i förhållande till klasserna Bok, Film och CD.

Tilldelning När man vill ändra värdet på en variabel så kallas detta för tilldelning.

Tillgänglighet Se åtkomst.

Variabel Se attribut.

Virtuell maskin En virtuell javamaskin är ett program som är skrivet för en viss plattform (t ex Windows 98) och som tolkar bytekod och utför instruktionerna i denna. I och med att bytekodens format är standardiserat kan en fil med bytekod köras på samtliga plattformar som har en virtuell javamaskin. Det är alltså samma bytekod som körs, men olika plattformar använder olika virtuella maskiner.

Åtkomst För att kunna kombinera principen om inkapsling med den om arv krävs att man kan styra hur tillgängliga metoder och variabler (och även klasser) ska vara för varandra. Detta görs genom s k åtkomstmodifierare - private, protected, public och default.

Bilaga 2: Intervjufrågor till studenterna, första intervjuomgången

1. **Hur gammal är du?**

2. **Vad har du för studiebakgrund/ arbetslivserfarenhet?** Hur ser din gymnasiebakgrund ut? Övriga studier? Arbetslivserfarenhet? Matematikstudier (utöver grundläggande C-nivå)?

3. **Berätta om dina förväntningar inför utbildningen i sin helhet.**

4. **Beskriv dina förväntningar inför programmeringskursen.** Vad innebär det att programmera för dig? Varför vill du lära dig programmera? Vad vill du använda det till? Vad hoppas du ha lärt dig efter genomförd kurs? Vad upplever du som det officiella målet med kursen?

5. **Försök att beskriva dina förkunskaper vad gäller datoranvändning.** Är du t ex van att arbeta i dos-miljö, installerar du ofta program själv, sitter du ofta och arbetar vid datorn för att lära dig använda program/ hitta den mest ultimata användningen, spelar du mycket dataspel o s v?

6. **Berätta om dina förkunskaper i programmering.** Har du t ex gått någon kurs tidigare? Vilka olika programmeringsspråk känner du till? Har du någon erfarenhet av programmering i html, javascript, xml, applets…?

7. **Beskriv de laborationer du har genomfört hittills.** Försök förklara vad de har gått ut på och berätta hur du uppfattat dem.

8. **Hur har du arbetat med laborationerna?** Har du t ex försökt lösa dem ensam eller tillsammans med dina klasskamrater (eller andra)? Har du börjat med att läsa mycket kurslitteratur för att sedan programmera eller tvärtom?…

9. **Kan du på ett ungefär bedöma hur mycket tid du har använt för varje laboration** (inklusive tid för läsning och eventuella sidoövningar för att förstå problemet)**?**

10. **Hur upplever du programmeringskursen, så här långt, i sin helhet?** - Beskriv hur du uppfattar föreläsningar respektive laborationer - svårighetsgrad, lustfylldhet, tid för att arbeta med laborationerna mm. Hur uppfattar du att programmeringstekniska (och andra) begrepp förklaras? Upplever du att föreläsningar och laborationer "hänger ihop"? Om inte - kan du beskriva var glappen finns? Kan du ge exempel på vad du upplever som lätt/ svårt? Stämmer utformningen med vad du hade förväntat dig?

11. **Vad tycker du om kurslitteraturen?** Beskriv för- och nackdelar samt berätta om eventuella referenser (litteratur, www, e t c) utanför de rekommenderade som du tycker dig haft användning av.

12. **Hur uppfattar du de lärare som deltagit i programmeringsundervisningen?** Försök göra en generell beskrivning av hur du uppfattar kunskapsnivå, förmåga att förklara o s v hos lärarna som grupp. Vilken syn på programmering upplever du att de har? Ange det som du uppfattar som sämst respektive det du uppfattar som bäst i deras undervisning.

13. **Hur uppfattar du stämningen i klassen vad gäller programmeringen?** (Försök fundera över hur du uppfattar denna stämning i klassrumssituationen såväl som utanför.) Upplever du att de andra tycker det är enkelt eller svårt? Är det lätt att ställa frågor? Tycker du att lärare respektive dina klasskamrater använder sig av olika datatekniska respektive programmeringstekniska begrepp på ett som du upplever motiverat och konsekvent sätt? Upplever du att man förväntas kunna vissa saker (tycker du själv att man ska kunna vissa saker?) och vad är det i såfall man förväntas kunna?

14. **Tror du att det har någon betydelse vilket kön man tillhör för hur man uppfattar undervisningen?** Förklara.

15. **Om du skulle föreslå någon förändring av programmeringsundervisningen -** vad skulle det vara?

Bilaga 3: Intervjufrågor till lärarna

1. **Beskriv din roll i programmeringskursen.**

2. **Vem ansvarar för vad i programmeringskursen?**

3. **Upplever du att det är tydligt definierat vad du ska göra/ inte göra?**

4. **Hur kom du in i yrket/ lärarrollen?**

5. **Hur vill du beskriva målsättningen med programmeringskursen?** Hur är kursen upplagd och varför är den upplagd som den är? Vad tycker du man ska kunna/ ha lärt sig efter den första kursen?

6. **Vad är din syn på programmering?** Vad innebär det? Vad är viktigt, hur "bör" man lära sig det, d v s vad bör betonas i undervisningen? Hur förmedlar du din syn? Vad utmärker en "god" programmerare?

7. **Hur uppfattar du studenternas kunskapsnivå/ förmåga att ta till sig undervisningen?** Var uppfattar du att de har problem/ svårigheter respektive vad förfaller vara lätt? Varför tror du det är så? Är förkunskapskraven tillräckliga?

8. **Beskriv de olika förhållningssätt till programmeringskunskap som du sett bland studenter (och lärarkollegor).**

9. **Har du sett några förändringar över åren bland studenterna när det gäller att komma in i programmeringen och hur man klarar kurserna?**

10. **Upplever du att det finns någon generell skillnad mellan hur män respektive kvinnor tar till sig undervisningen?** Om du upplever någon sådan skillnad - hur skulle du i så fall vilja förklara den?

11. **Upplever du att det finns någon generell skillnad mellan hur studenter i olika åldrar och med olika bakgrund tar till sig undervisningen?** Förklara**.**

12. **Upplever du att du kan besvara de frågor du får av studenterna?** Hur hanterar du de eventuella fall då du inte kan besvara en fråga?

13. **Berätta om dina egna erfarenheter av att lära dig programmera.**

14. **Om du skulle förändra något i undervisningen - vad skulle det vara?**

Bilaga 4: Intervjufrågor till studenterna, andra intervjuomgången

1. **Hur upplever du din datorvana idag jämfört med när du började?**

2. **Vilken betydelse upplever du att dina förkunskaper alternativt brist på förkunskaper har haft för att du ska ta till dig undervisningen?**

3. **Anser du att man borde ha vissa förkunskapskrav vid intagningen? - Vilka?**

4. **Hur har dina förväntningar på programmeringskursen infriats? Beskriv vad du är mest/ minst nöjd med.**

5. **Kan du beskriva vad du har lärt dig under höstterminen vad gäller programmering?** Upplever du att du har sammanhangen klara (t ex hur man bygger upp ett program och varför man väljer olika lösningar)?

6. **Om du upplever att du har förlorat greppet, var och när hände det i såfall och vad tror du det beror på?**

7. **Vilka förväntningar har du upplevt vad gäller din egen prestation i programmeringskursen** (från dig själv, från klasskamrater, från lärare)? Är förväntningarna tydliga?

8. **Vad tycker du om examinationsformen?** Skulle du vilja ha andra redovisningsformer? Hur skulle de se ut?

9. **Hur skulle du vilja lära dig programmera?** Försök tänka på något som du tycker är riktigt roligt att göra, hur du gör det, stämning, omgivande villkor etc. Försök sedan tänka dig att du programmerar under likadana betingelser. - Vilken bild kan du då måla upp?

10. **Har dina förväntningar på utbildningen i sin helhet infriats så här långt, om inte - vad är det du saknar/ är missnöjd med? Om du är nöjd - kan du beskriva vad du framförallt är nöjd med?**

11. **Hur tycker du att schemaläggningen för höstterminen har fungerat i sin helhet?** Hur har det t ex fungerat att läsa flera ämnen parallellt över en längre period? Bidrar de olika delkurserna till varandra och i så fall på vilket sätt? Tid för att smälta?

Tilläggsfråga MT-programmet: Vilken är din upplevelse av att läsa i en grupp där förkunskaperna är så blandade?