Förenklade verktyg i högre utbildning:
erfarenheter från programmering och elektronik

Anders J Johansson och Sara Willhammar, *EIT, LTH*

***Abstract*—Att introducera programmering till studenter utan programmeringsbakgrund är en utmaning, eftersom det innebär att man behöver lära sig både ett nytt språk och ett nytt sätt att strukturera sitt tänkande. Det finns idag en uppsjö av programmeringsspråk att välja på, där valet kan ske på olika grunder, dessa kan t.ex. vara pedagogiska, där språket stödjer den metod man vill lära ut, eller praktiska där man väljer ett språk som är relevant i omvärlden. I en projekt-kurs för industri-design som syftar till att ge en förståelse för möjligheterna i programmerbara system så har vi använt ett mycket förenklat språk. Detta är ursprungligen framtaget för skolbarn, och det isolerar och tydliggör de logiska delarna av programmering samtidigt som det till stor del gömmer den komplexa syntax som många moderna språk har. Språket är fortfarande tillräckligt kraftfullt för att användas i de produkt-prototyper som kursen leder fram till. Detta har fallit väl ut för en studentgrupp som i stort sett saknar bakgrund i programmering. I samma kurs så använder vi också en workshop för att introducera elektronik, också denna baserat på teknik ursprungligen framtaget för barn. Vi har sett samma fördelar här att det förenklade handhavandet stödjer pedagogiken, trots att det sker på bekostnad av relevansen. Vi har sedan tagit med oss erfarenheterna från detta till ytterligare en kurs där studenterna i Industriell Ekonomi introduceras till elektronikprojekt. Här kan de redan programmera, men vi använder i stället förenklade byggsätt för elektroniken för att kunna illustrera de för kursen relevanta delarna samtidigt som vi gömmer en del av komplexiteten inom elektronikkonstruktion.**

***Index Terms*—Förenklade verktyg, Lies-to-children, Berättelse, Narrativium, story-telling**

I. Introduktion

ATT introducera komplicerade begrepp för studenter är en klassisk pedagogisk utmaning. I vissa av våra kurser är målet inte att göra studenterna till experter inom en viss domän, utan att ge dem en förståelse för den och att använda den och dess verktyg i annan verksamhet. Ett exempel är en kurs som introducerar elektronik och programmering för industridesignstudenter. Målet är här inte att de ska bli programmerare eller elektronik-designers, utan att de ska få en förståelse för hur de kan använda dessa teknologier inom sin verksamhet som produktdesigners. Ett annat exempel är en projektkurs i elektronik för studenter i Industriell ekonomi där målet för dem snarare är att få en förståelse för industriellt utvecklingsarbete inom elektronikprojekt; projekt som de exempelvis skulle kunna jobba inom i deras framtida karriärer. Vi hamnar här i behovet att abstrahera bort mycket av komplexiteten, utan att förlora funktionaliteten och de möjligheter denna ger. Det här blir än mer relevant om vi vill arbeta med inbyggda system, dvs. programmerbara mikrokontrollers som är direkt kopplade till olika sensorer och displayer, och som kontrollerar olika maskiner och processer i realtid. Detta är ett steg mer komplicerat än att studera algoritmer som löser matematiska problem i en PC.

Detta kan liknas vid konceptet ”Lies-to-children”, där vi förenklar verkligheten för att göra den gripbar för barn i en undervisningssituation. Detta begrepp populariserades av Pratchett et.al (1999), och till detta kan man också koppla idén om *Narrativium*, vilket kan definieras som ”behovet av förklaringar som läsaren förstår och fängslas av: förenkling och berättelse.” (Sawyer 2000) (vår översättning). Dvs. vi lägger gärna upp presentationen av en teori eller teknologi som en berättelse, och när vi gör det så förenklar vi sådant som vi inte ser som centralt till det vi vill förmedla i den aktuella situationen, och det är detta som är ”lögnen” i begreppet ”Lies-to-children”. Detta kopplar även till ett föregående arbete (Johansson 2019) om berättelsens vikt i en föreläsning för att göra den intressant, och pekar på betydelsen i att lägga upp kunskap i form av en berättelse, eftersom det kan förenkla förståelse och inlärning.

Flera exempel kan tas från fysiken, där alla de modeller vi lär ut är förenklade, och där vi ofta har bättre modeller att tillgå som vi ändå väljer att inte använda. Ett klassiskt exempel är att vi lär ut Newtons mekanik, även om vi vet att Einstein gjorde viktiga korrigeringar av modellerna för massa, energi och acceleration. Att förenkla elektronik och programmering i pedagogiska sammanhang har en lång historia (Rony et.al. 1977), har från början mötts av en viss skepsis. Konceptet kan kritiseras för att det kan ge studenterna en förenklad världsbild, och det är därför viktigt att de är medvetna om att vi använder förenklade modeller. Det är också viktigt att man inte lär ut saker som senare behöver läras om, utan bara utelämnar eller gömmer komplexiteten.

I Walsh & Currie (2015) så skiljer de på att lära ut *myter* och *karikatyrer*. *Karikatyrer* är då man förvränger, i deras fall, historiska skeenden för att underlätta förståelsen av filosofiska resonemang, alternativt av andra pedagogiska skäl. De blir dock till *myter* om man förvränger historien så pass mycket att det hindrar framtida kunskap om samma historiska skeenden. Artikeln argumenterar för att *karikatyrer* är produktiva att använda i undervisning, och ibland nödvändiga, men att *myter*, som hindrar framtida inlärning, inte ska användas. Vi kan här dra en parallell till undervisning av teknikvetenskap. Här har vi inte samma fokus på de historiska skeendena, men vi studerar system uppbyggda av komponenter, som i sin tur är komplexa

Anders J Johansson (e-mail anders.j.johansson@eit.lth.se) är Universitetslektor vid institutionen för Elektro- och Informationsteknik på LTH.

Sara Willhammar (e-mail sara.willhammar @eit.lth.se) är Postdoktor vid institutionen för Elektro- och Informationsteknik på LTH.

system. Vi väljer sällan att diskutera och modellera systemen ner till sina minsta beståndsdelar, utan vi väljer en förenklad, men tillräckligt komplicerad, modell av de ingående delarna. Hur dessa modeller är uppbyggda varierar, men analogt med föregående resonemang kan vi säga att dessa får vara förenklade, men inte så förenklade eller omskrivna att de hindrar framtida detaljkunskap om de ingående systemen.

Parallellt med teknikutvecklingen i industrin och inom högre utbildning så har det skett en utveckling att göra programmering och elektronik mer tillgängligt för en bredare allmänhet. Detta både i system framtagna för barn i grundskolan och även för hobbybruk. Vi kan analysera dessa förenklade system och verktyg och se om de går att klassificera som *karikatyrer*, och då vara användbara, eller som *myter*, och då riskera att göra mer skada än nytta. En *myt* blir i detta sammanhang något som döljer för mycket av komplexiteten, eller av dess kompromisser. En *myt* skulle också kunna vara något som ger en felaktig uppfattning om den underliggande funktionen av komponenten.

Det skal också nämnas att uppdelningen mellan karikatyrer och myter inte alltid är självklar, och kommer bero på vem som gör den. Men begreppen i sig ser vi som användbara i analys av pedagogiska framställningar och verktyg, och de, tillsammans med begreppet ”Lies-to-children” och idén om narrativet, är ett ramverk utifrån vilket vi kan diskutera och analysera olika pedagogiska framställningar.

II. Förenklade Verktyg

Det vi har arbetat med är att använda teknologier och verktyg utvecklade för dels mycket yngre studenter, dels för hobbybruk, i vår undervisning. Dessa är valda så att de fortfarande är kompletta, dvs. de abstraherar inte bort några möjligheter inom sitt respektive område, samtidigt som de gömmer en stor del av komplexiteten av de ingående systemen. I de kurser som vi presenterar här har vi arbetat med authentic learning (Herrington 2006) i form av projektbaserad undervisning, där valet av förenklade verktyg gör att vi kan uppfylla kriteriet att vara verklighetsbaserade och relevanta trots de begränsade förkunskaperna hos studenterna.

Kurs 1: MakeCode för Indu stridesignstudenter

I kursen ETIA06 så är målet att industridesignstudenterna ska lära sig om vilka möjligheter och begränsningar som det ger att inkludera programmerbara elektroniska delar i en design. Detta gör de i projektform genom att designa ett föremål och bygga en funktionell prototyp av detta. Som central del av det elektroniska systemet använder de Playground Express, vilket är ett kretskort med lysdioder, givare och en mikrodator, framtaget för undervisnings- och hobbybruk. Playground Express påminner i många delar om MicroBit, vilken togs fram av BBC i Storbritannien och som används både där och i Sverige för att introducera programmering i grundskolan, och bägge programmeras i språket MakeCode.

Att lära sig programmering från grunden är en utmaning, eftersom det innebär att man behöver lära sig både ett nytt språk och ett nytt sätt att strukturera sitt tänkande. Det finns idag en uppsjö av programmeringsspråk att välja på, där valet kan ske på olika grunder, som pedagogiska eller med målet att vara aktuell och relevant. MakeCode är ett språk som är utvecklat för barn och som bygger på pusselbits-programmering (Devine et.al. 2019). Det påminner till vissa delar om Scratch, vilket även det är framtaget för undervisningsändamål. En skillnad är att MakeCode är designat för att användas i inbyggda system, dvs. system som typiskt inte använder ett tangentbord, mus och skärm för att interagera med omvärlden.

MakeCode har flera pedagogiska fördelar, som att det inte går att göra rena syntaxfel då pusselbitarna bara blir aktiva om de placeras i koden på ett sätt som går att exekvera. Detta leder till att många enkla fel försvinner, som felstavade kommandon, glömda parenteser och missade semikolon. Samtidigt så är språket komplett, dvs. det kan, i princip, implementera alla kända algoritmer. Det enda som direkt saknas är ”äkta” funktioner, dvs. funktioner som ger ett värde tillbaka. Det saknas också mycket av de ”magiska” kommandon som ofta behövs i andra språk, och länkning och kompilering sker genom en knapptryckning som samtidigt också laddar ner en fil på datorn som är färdig att exekveras på kretskortet. Nackdelen med MakeCode är att större program snabbt blir ohanterliga och svåröverskådliga. Det är också svårt att hantera kommentarer i koden. Rent praktiskt så bör man byta till ett annat språk så snart hela programmet inte syns på en, eller möjligen två, skärmar samtidigt.

Att introducera programmering via konceptet SARA (sekvens-alternativ-repetition-abstraktion), och analog programmering, och sedan gå vidare och visa respektive begrepp i MakeCode är en rak process. Att det sedan är enkelt att ta steget vidare till Playground Express gör att man direkt kan se att programmet fungerar även utan en PC eller laptop, och detta utan att få några fel pga. felaktiga kopplingar, då många sensorer och lysdioder finns tillgängliga direkt på kortet. Det är dock också möjligt att koppla till externa delar, som ytterligare strömbrytare, lysdioder och motorer. Detta ger att man kan introducera komplexiteten steg för steg, och även visa på hur man inom ingenjörspraktiken delar upp system i små delar och testar dem var för sig systematiskt.

I ljuset av föregående så ser vi MakeCode som en *karikatyr* av programmering, dvs. något som döljer en del av komplexiteten av underliggande system, som länkning och kompilering, utan att hindra framtida förståelse av dessa begrepp.

Kurs 2: Elektronik för I3

I denna kurs så ska studenterna använda sina programmeringskunskaper från två tidigare kurser för att i projektform bygga en prototyp som är baserad på en inbyggd mikrokontroller. Till denna så ska det koppla sensorer och ut-enheter, som displayer och lysdioder. Vi har här valt att basera projekten på delar framtagna för hobby och prototypbygge. Dessa bygger på att delarna kopplas ihop med en standardiserad kabel, STEMMA/Qwiic, vilken distribuerar både matningsspänning och en databuss. Den senare följer I2C-standarden, vilket gör att det finns en större mängd komponenter i form av sensorer och liknande tillgängliga. Två företag, AdaFruit och SparkFun, tar i sin tur bägge fram små kretskort som förser dessa komponenter

med nödvändiga kringkomponeneter, och med kontakter så att de enkelt går att koppla ihop. Detta ger även här en abstraktion av det arbete som en elektroingenjör skulle ha gjort och tar bort felkällor. En stor del av komplexiteten abstraheras också bort då studenterna inte behöver gå in och läsa datablad, studera nödvändig kretskortslayout, kringkomponenter eller sätta upp enheterna för rätt sorts kommunikation, utan detta löses genom ett färdigt kretskort och färdiga mjukvarubibliotek. I denna kurs så programmeras projekten i MicroPython, vilket är en dialekt av Python som är anpassad för inbyggda system.

Även här så ser vi STEMMA/Qwiic och användandet av färdiga moduler och bibliotek som en *karikatyr*, då den förenklar byggandet av prototyper utan ingående kunskap om elektronikkonstruktion, men utan att göra detta till en *myt*, då det inte gömmer komplexiteten bakom någon ytterligare förklaringsmodell.

III. Erfarenheter

MakeCode har nu använts i flera år i kursen för industridesign. Språket fungerar bra som ett verktyg för att introducera programmering för dessa studenter, och det är samtidigt tillräckligt kraftfullt för att fylla de behov deras projekt har. Vissa studenter har redan kunskap om programmering, och dessa har i vissa fall valt att istället använda andra språk. Vissa större program och mer komplexa designer tänjer på gränserna för vad MakeCode är ett lämpligt verktyg för, men visar då samtidigt för studenterna varför vi i praktiken använder andra språk, och att det finns en anledning till den komplexitet de har. En poäng här är att det i verktyget enkelt går att växla mellan MakeCode och JavaScript, vilket enkelt visar på att pusselbitarna bara är ett annat sätt att representera samma logiska struktur eller program som en textfil. Att MakeCode är ett väl spritt verktyg leder också till att studenterna kan hitta hjälp och lösningar på internet, vilket också ger en inblick i hur praktiskt ingenjörsarbete faktiskt utförs.

Kursen för studenterna i Industriell Ekonomi har precis genomförts för första gången med det här upplägget. Studenterna kunde använda sig av sina tidigare programmeringserfarenheter för att snabbt lära sig att använda ett, för nästan alla, obekant programmeringsspråk. Trots att väldigt få hade någon erfarenhet av elektronik sedan innan så lyckades samtliga grupper att ta sina produkter från idé till prototyp. Projekten kunde under kursens gång modifieras med hjälp av en handledare, som agerade som representant från en fiktiv produktutvecklingsledning, för att hitta en lämplig nivå och svårighetsgrad. Baserat på erfarenheten från den här första kursomgången så finns det utrymme att öka komplexiteten rörande elektroniken ytterligare. Även i den här kursen så kunde studenterna hitta hjälp på internet och jobba på ett sätt som liknar praktiskt ingenjörsarbete.

IV. Slutsatser

Verktyg framtagna för barn och för hobbybruk har ofta fördelen att de förenklar handhavandet av komplex teknik. I vissa fall i vår undervisning på LTH så är detta en tillgång, då vi kan använda dessa förenklingar för att abstrahera bort komplexitet som ligger utanför kursmålen för en specifik kurs, samtidigt som det ger studenterna möjlighet att arbeta med modern teknik i relevanta system. Att marknaden för grundskola och hobby är större än för ren högskoleundervisning ger också att dessa verktyg uppdateras, utvecklas och är tillgängliga till ett rimligt pris.

Vi har använt dessa med goda resultat i två kurser, där de möjliggör användandet av avancerad teknik även då detta i sig inte är målet för kursen, men där det leder till ökad relevans i uppgifterna man arbetar med. De fungerar här bra som karikatyrer av den underliggande komplexiteten, enligt det ramverk vi introducerat ovan.

Därmed inte sagt att de specifika verktygen passar på alla våra kurser. Tvärtom så är det tydligt att man når en gräns där man behöver introducera den underliggande komplexiteten för att förklara begränsningar eller lösa vissa specifika problem. Då detta sker i en kurs som använt de förenklade verktygen, som också introducerats som detta, blir effekten att det tydligt visar för studenterna varför den större komplexiteten finns och behövs. Detta pekar också på att de inte är myter, vilket skulle ha hindrat eller försvårat detta steg. I andra kurser, som de för studenter som redan har studerat grunderna, så finns det antagligen begränsade vinster att använda dessa enkla verktyg, då man snabbt kommer att få behovet att ta steget vidare.

Ramverket med att analysera förenklade strukturer som *karikatyrer* eller *myter* ser vi som användbart i utvecklingen av nya kurser och kursmoment. Kopplingen till berättelsen och narrativet är något som förtjänar ytterligare studier framåt.

Författarnas tack

Tack till alla studenter som hart deltagit i dessa kurser och som med sin entusiasm och återkoppling har bidragit till utvecklingen av dem, och av de idéer som presenterats ovan.

Referenser

Devine, J., Finney, J., de Halleux, P., Moskal, M., Ball, T. & Hodges, S. (2019). *MakeCode and CODAL: Intiutive and efficient embedded systems programming for education.* Journal of Systrems architecture 90, 2019, pp. 468-483.

Herrington, J. (2006) *Authentic E-learning in higher education: Design principles for authentic lerarning environments and tasks.* ELEARN 2006, Honolulu, Hawaii, USA

Johansson, A.J. (2019) *The Lecture as Storytelling*, 7:e utvecklingskonferensen för Sverigegs Ingenjörsutbildningara, Luleå, 2019

Pratchett, T., Stewart, I. & Cohen, J. *The Science of Discworld*, 1999, Ebury Press.

Rony, P.R. & Larsen, D.G. (1977) *Teaching Microcomputer Interfacing to Non-Electical Engineers*, Computer, vol. 10, no. 1, pp. 53-57, Jan 1977.

Sawyer, A. (2000). *Narrativium and Lies-to-Children: "Palatable Instruction " in "The Science of Discworld".* HJEAS, 2000 spring., vil. 6, No. 1, pp 155-178.

Walsh, K. & Currie, A. (2015). *Caricatures, Myths and Lies.* Metaphilosophy, Vol. 46, Issue 3, pp. 414-435., July 2015, Whiley.