Digitaliseringspotential av programmeringskurser inom den högre utbildningen

Fabian Lorig
Fakulteten för Teknik och samhälle, Internet of Things and People Research Center, Malmö Universitet

Abstract (Svenska)
För studenter inom IT-relaterade ämnen, är programmering en grundläggande kunskap som bör utvecklas tidigt i utbildningen. Det innebär inte bara lärandet av syntax och semantik av ett visst programmeringsspråk, utan också att utveckla förmågan att kunna kombinera olika enskilda instruktioner till en algoritm eller ett dataprogram som kan lösa ett visst problem. Hittills har många programmeringskurser inom högre utbildning genomförts som campuskurser och skiftet till distansundervisning har medfört behovet att identifiera nya lämpliga undervisningsformer.

Syftet med denna artikel är att analysera befintliga online-lärandemiljöer för att identifiera innovativa verktyg och didaktiska angreppssätt som kan användas i distansundervisning för programmering inom högre utbildning. I artikeln presenteras en diskussion om deras lämplighet för digitalisering av olika moment i programmeringskurser, ur både studentens och lärarens perspektiv, men också hur den konstruktiva länkningen med aktuella kursmål kan uppnås eller styrkas. Digitaliseringspotentialen visas genom exemplet kursen DA343A (”Objektorienterad programutveckling, trådar och datakommunikation”) på Malmö Universitet, som riktar sig till studenter på kandidatprogrammet i datavetenskap med inriktning systemutveckling och högskoleingenjörsutbildningen i datateknik.

Abstract (English)
For students of IT-related subjects, programming is a fundamental skill that should be acquired at an early stage of their education. This includes not only the learning of syntax and semantics of specific programming languages but also the developing of capabilities to combine different computer instructions to an algorithm or a computer program that can solve a particular problem. So far, many programming courses were held as on-campus courses and the shift to online courses requires the identification of new adequate forms of education.

The aim of this article is to analyze existing online learning platforms and to identify tools and didactical methods that can be used for online programming courses in higher education. In the article, a discussion of their suitability is presented for digitalizing different sessions of programing courses from both the students’ and teachers’ perspective but also of how the constructive alignment with existing learning outcomes can be achieved and strengthened. As a case study, the potential for digitalization is analyzed for course DA343A (“Object-oriented programming, threads, and communication”) at Malmö university, which is aimed at students in the bachelor’s programs in data science with focus on system development and computer engineering.

1 Introduktion

För studenter inom olika IT-relaterade ämnen är programmering en grundläggande kunskap som bör utvecklas tidigt i utbildningen. Det innebär inte bara lärandet av syntax och semantik av ett visst programmeringsspråk, utan också att utveckla förmågan att kunna kombinera olika

enskilda instruktioner till en algoritm eller ett dataprogram. I dagsläget genomförs många programmeringskurser inom högre utbildning fortfarande som campuskurser eftersom den personliga handledningen under praktiska övningar (till exempel laborationer) är ett givande moment i kursen. Det är här studenterna arbetar med olika uppgifter för att skapa och stärka rutiner, självständighet och samarbetsförmåga, genom att använda olika tekniska koncept och teorier från föreläsningarna.

På grund av den nuvarande coronaviruspandemin, men också som del av den generella utvecklingen av lärandet, finns det en trend som går mot digital undervisning och examination inom högre utbildning (Hrastinski, 2018). För studenterna innebär utbildning på distans flera fördelar som till exempel ökad flexibilitet angående familj eller jobb, möjligheten för individuell anpassning av lärtempot och att kunna upprepa lektioner för att stärka förståelsen. Samtidigt betyder distansundervisning också att det är lättare att undvika kontakt (*social eller fysisk distansering*), vilket är eftersträvansvärt med hänsyn till att minska smittspridningen.

Utmaningen är att många av de nuvarande undervisningsformerna i programmeringskurser inte är lämpliga för distansundervisning, eftersom de är beroende av studenternas fysiska närvaro. Det gäller främst de praktiska kursmomenten, som laborationer eller grupprojekt, där erfarna handledare ger återkoppling och råd medan studenterna löser olika praktiska programmeringsuppgifter, vilket bidrar till ett effektivt och framgångsrikt lärande. Att överföra sådana kursmoment direkt till online-plattformar kan påverka lärandet och nåendet av lärandemålen som definieras i kursplanen på ett negativt sätt. Det måste också beaktas att online-undervisning inte är någon universalmedicin mot alla pedagogiska problem och att det måste utvärderas och planeras noggrant om och hur tekniken kan användas för att främja och stödja lärandet (Elmgren & Henriksson, 2019; Laurillard, 2002).

Syftet med den här artikeln är att identifiera och presentera lämpliga metoder som kan användas för digitalisering av olika moment inom Java-programmeringskurser inom högre utbildning, men också att diskutera hur den konstruktiva länkningen mellan dessa metoder och kursens befintliga lärandemål kan stärkas. Det innebär moderna didaktiska angreppssätt för online-föreläsningar om programmeringsspråk, men också digitala verktyg och anpassade uppgifter för praktiska övningar som är väsentliga för att fördjupa teorin och för att få rutin i programmering. Slutligen presenteras och diskuteras digitala examinationsformer som kan ersätta en skriftlig salstentamen och som ger studenterna bästa möjlighet att visa sina förvärvade kunskaper.

Digitaliseringspotentialen visas genom exemplet kursen DA343A (”Objektorienterad programutveckling, trådar och datakommunikation”) på Malmö Universitet, som är en fördjupningskurs i programmeringsspråket *Java*. I dagsläget är kursen en del av utbildningen till högskoleingenjörer med fokus på datateknik och mobil IT och till systemutvecklare. Den ligger under den andra terminen av kandidatutbildningen och kräver grundläggande förkunskap inom programmering med Java som förmedlas i en kurs under första terminen. Till och med vårterminen 2020 erbjöds DA343A som campuskurs på Malmö Universitet. Kursen består av klassiska föreläsningar i hörsalar, praktiska laborationer med handledning i datorsalar och gruppuppgifter i projektform som ofta bearbetas genom att studenterna träffas på universitetet.

2 Digitala föreläsningar om programmeringsspråk

Ett programmeringsspråk används inom datorsystem för att definiera vad datorn ska göra genom att skriva dataprogram. Ett program består av en serie instruktioner som datorn betar av. Det kan jämföras med ett recept som man använder i köket när man till exempel ska baka en kaka. Syftet med både programmet och receptet är att instruktionerna beskrivs så noggrant som möjligt så att den som ska följa dem kan göra så utan vidare hjälp och utan att behöva ställa ytterligare frågor till författaren. Precis som i den reala världen med många främmande språk, finns det också flera olika programmeringsspråk med egen syntax och semantik, som olika datorsystem kan förstå.

Den här artikeln fokuserar på programmeringsspråket Java, ett mångsidigt objektorienterat programmeringsspråk från 1995. Språket används i många större projekt men är också lämpligt som språk för nybörjare. Enligt TIOBE popularitetsstudie[1] från maj 2020 är Java, trots sin ålder, fortfarande på andra plats, efter att förra året legat på första plats. Enligt andra studier som baserats på github.com, ett webbhotell för mjukvaru-utvecklingsprojekt som ägs av Microsoft, använder 14 % av alla användare Java som programmeringsspråk – vilket motsvarar tredje plats efter JavaScript och Python (Fredrikson, 2018). Studierna visade också en fortsatt ökning av antalet Javautvecklare i industrin under de senaste åren, vilket är en indikator för relevansen på arbetsmarknaden (DAXX, 2020).

Salanci (2015) beskriver en process med fem steg om hur programmeringskunskap skapas (Bild 1). Först ska ett problem (*problembaserat lärande*) eller ett case (*casemetodik*), som ger en ram eller kontext till lärandemålet, presenteras för studenten. Det är viktigt att motivera studenterna genom att till exempel belysa relevansen av det som ska läras in med hjälp av ”vardagsproblem” från det verkliga livet och att lära känna ”best practices” för att lösa olika sorters problem (Ansarian & Teoh, 2018; Elmgren & Henriksson, 2019). Studenterna ska sedan ha möjlighet att samla egna erfarenheter men också få en förklaring som innehåller underliggande regler som hjälper dem att generalisera den förvärvade kunskapen. Slutligen måste kunskapen förstärkas och fördjupas genom praktiska övningar.

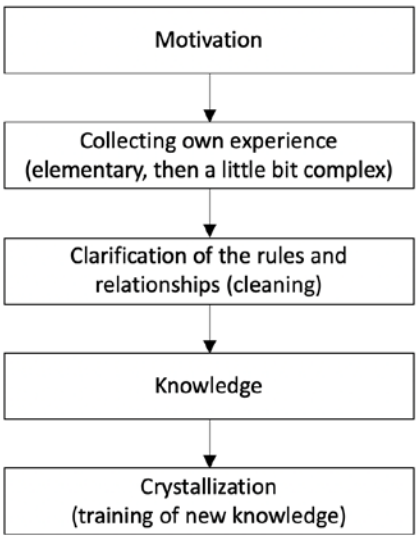

Bild 1: Steg för inlärandet av programmering och andra färdigheter (Salanci, 2015).

Idén att undervisa programmering med hjälp av digitala verktyg är inte ny. För privatpersoner finns det ett stort urval av online-resurser och -kurser för att lära sig många olika

[1] https://www.tiobe.com/tiobe-index/

programmeringsspråk. Dessa kurser är ofta riktade mot den privata utbildningen som sker efter jobbet, på fritiden och som kvalifikation för nya arbetsuppgifter. Inom den högre utbildningen finns däremot inga liknande övergripande plattformar och många lärare känner behovet att producera nya videor för den egna kursen – en tidskrävande uppgift som många skyr. Så varför inte kombinera, vidga och använda det som redan finns?

Ett exempel på online-resurser, som också kan användas inom föreläsningar, är YouTubekanaler (till exempel TheNewBoston[2] eller Derek Banas[3]). Dessa kanaler tillhandahåller handledningsvideor som förklarar grundläggande men också mer avancerade koncept inom programmering. Fördelen med videor är att de kan pausas, hastigheten kan anpassas och de kan tittas på flera gånger ifall en repetition behövs. Den positiva effekten av att använda videor som föreläsningsmedium eller att ladda upp inspelningar av föreläsningar undersöktes bland andra av Cascaval et al. (2008). Det visade sig att studenterna uppskattade möjligheten att ha tillgång till videor av föreläsningen men också att studenterna med lägre betyg tittade på videon oftare. Samtidigt visade det sig att studenterna upplevde inspelningen under föreläsningen som störande eller distraherande. Studier visar också att tiden som läraren måste investera i online-kurser är högre per student jämfört med konventionella kurser (Cavanaugh, 2005). Det beror bland annat på den högre tidsåtgången för att spela in, efterbearbeta och ladda upp föreläsningar. Men tidsåtgången kan minskas genom att använda sig av resurser som redan finns och kombinera dem med eget material.

Existensen av online-plattformar och tillgången till handledningsvideor betyder dock inte att lärarens roll blir överflödig – utan tvärtom. Genom att kunna använda sig av existerande och omfattande videor får läraren möjlighet att fokusera kursmomenten och studentkontakten på de mer praktiska aspekterna av programutvecklingen och på att transferera det som lärs ut. Det kan till exempel innebära användandet av programvaran inom olika verksamhets- eller affärsprocesser, som är en del av det praktiska arbetet för många studenter efter avslutad universitetsutbildning.

3 Digitala verktyg för praktiska övningar

Praktiska övningar är centrala för att studenterna ska kunna tillägna sig tekniska rutiner men också för att främja självständighet och samarbetsförmåga (Elmgren & Henriksson, 2019). På så sätt ger laborationer nya perspektiv på teorin genom att koppla samman teori och praktik. Praktiska online-övningar för programmering är sällsynta inom högre utbildning. Istället följs varje föreläsning av en laboration, där studenterna arbetar med praktiska övningar och uppgifter. Målet med praktiska övningar är att studenterna ska lära sig att lösa problem på ett tekniskt sätt. Eftersom övningarna ofta är komplexa och oförutsägbara, krävs det stor flexibilitet av lärarna och handledarna som ofta är studenter som absolverade kursen året innan (Elmgren & Henriksson, 2019).

För privatpersoner och hobbyutvecklare finns det ett urval av kostnadsfria online-kurser med praktiska övningar (*learning-by-doing*). Utöver det finns det också professionella avgiftsbelagda kurser som även kan användas för den egna yrkesmässiga utbildningen. Vissa av dem utfärdar även ett certifikat som kan användas som merit (till exempel codecademy.com eller udemy.com). Dessa kurser satsar ofta på interaktiva lärandemiljöer med hänsyn till den didaktiska processen som Salanci (2015) beskriver (Bild 2). Här kombineras videor och skriftlig information med praktiska övningar samt återkoppling, till en

[2] https://www.youtube.com/user/thenewboston
[3] https://www.youtube.com/user/derekbanas

omfattande lärprocess. Det är inte minst återkopplingen som gör dessa kurser nyttiga och som även är en central beståndsdel i undervisningen (Stigmar, 2009), men detta saknas ofta i videoföreläsningar eller -övningar. Det handlar inte bara om återkoppling genom bedömning, utan huvudsakligen att ge konkreta och tydliga förbättringsförslag som ger studenterna möjlighet till positiv utveckling. Uppmuntring genom positiv återkoppling bidrar också till ett motiverat lärande.

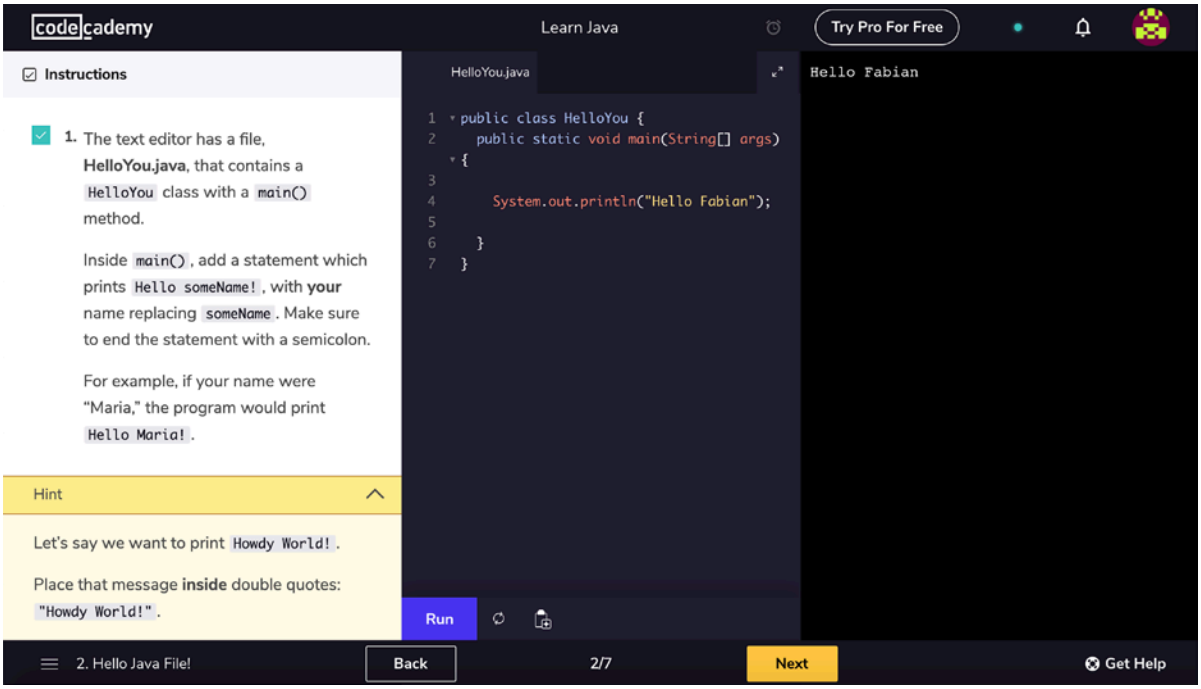

Bild 2: Lärandemiljö med tre delar: (vänster) instruktioner och tips, (mitten) användarens utvecklingsmiljö och (höger) resultaten när programmet utförs (codecademy.com).

Ofta består online-övningar av flera steg med mindre uppgifter som vid slutet formar ett sammanhängande program. I början finns det ofta en kort introduktion som beskriver det huvudsakliga syftet av programmet som ska utvecklas. Det ger studenten möjlighet att välja lektion efter det konkreta behov eller projekt hen har. Varje steg består av en kort teoretisk förklaring, till exempel av en ny operator eller anvisning, som sedan ska implementeras. Det finns möjlighet att testa koden och att få detaljerad återkoppling med konkreta förbättringsförslag ifall den innehåller några fel.

Handledningen sker nästan helt automatiskt eftersom det önskade resultatet av uppgiften och vanliga fel är definierade. Att fullständigt automatisera granskningen av uppgifter är bara möjligt vid mindre uppgifter i grundläggande kurser, där syftet av granskningen är att kontrollera om syntaxen har använts på rätt sätt och att resultatet är som förväntat (Ala-Mutka, 2005). Senare i utbildningen blir uppgifterna större och det blir snarare mindre programmeringsprojekt som studenterna arbetar med. Då kan den automatiska granskningen fortfarande användas som komplement eller förarbete till den manuella granskningen och för att effektivisera den. Exempel på vad som kan granskas automatiskt är programmeringsstilen, programmeringsfel eller olika mått på kodkvaliteten som till exempel *lines of code* eller *McCabe-tal* för cyklomatisk komplexitet (Fenton & Bieman, 2014).

Här visar sig dock en nackdel vid automatisering, nämligen att bara bekanta och vanliga fel kan upptäckas och respektive återkoppling kan ges. Men även etablerade agila arbetsmetoder

för att upptäcka programfel såsom par-programmering, där en observatör granskar koden medan den skrivs, är svåra att använda i online-miljöer där studenten löser sina uppgifter ensamma och utan andra studenter som de kan diskutera sina problem med (Williams & Kessler, 2002).

Utveckling, underhåll och skötsel av online-lärandemiljöer är tidskrävande uppgifter men de kan löna sig. Efter att ha utvecklat en plattform är det ofta ingen större skillnad om den används av 30 eller 250 studenter, Jämfört med campuskurser, där ökningen av studentantal kräver bokningar av fler datorsalar som bara finns i begränsat kapacitet, tillåter online-miljöer en flexibel ökning och minskning av antal studenter. Utöver det, undervisar de flesta högskolor samma programmeringsspråk så att synergierna kan utnyttjas.

4 Digital examination av programmeringskunskap

Enligt Gerrevall (2009, s. 97) är ”att examinera och att sätta betyg […] några av lärarens absolut viktigaste uppgifter”. Syftet med examinationen är att ge studenten återkoppling om hens kunskaper i ett visst ämnesområde, samt att kontrollera om kursplanens krav uppfylls. Många lärandemål i programmeringskurser beskriver praktiska skickligheter såsom användningen av vissa digitala programmeringsverktyg och implementeringen av konkreta funktioner.

I dagsläget genomförs många examinationer av programmeringskurser fortfarande som salstentamen på papper och utan hjälp av datorer. Det beror ofta på att många utvecklingsmiljöer för dataprogram ger utvecklaren utökat stöd, genom att till exempel kontrollera den korrekta användningen av syntax och semantik, samt ger föreslag på hur koden kan förbättras. Utöver det finns det många webbsidor och online-forum som tillhandahåller lösningar för vanliga programmeringsproblem. Särskilt i grundläggande kurser, där elementära kunskaper ska läras in, är användningen av dessa hjälpmedel ofta inte önskvärd. Det kan jämföras med språkkurser för nybörjare där det är tillåtet att använda ett ordbehandlingsprogram (till exempel Microsoft Word) med autokorrigering för att skriva en diktamen eller uppsats.

En av två vanliga möjligheter att hantera problemet, är användning av en reducerad eller begränsad utvecklingsmiljö på ett speciellt datorsystem utan tillgång till internet och utan funktioner som kontrollerar programkodens syntax och semantik. Administration och uppdatering av dessa system är tids- och resurskrävande och tillhandahålls därför inte av många skolor, högskolor eller universitet.

Den andra möjligheten är att anpassa examinationsuppgifterna till den reala situationen som studenterna kommer att möta i arbetslivet, där alla dessa stödfunktioner som förenklar programutvecklingen finns och ska användas. Så kallad öppen bok-tentamen (*open book examinations*), där alla hjälpmedel är tillåtna, skiljer sig från traditionella examina för både studenterna och lärarna (Francis, 1982). Det beror också på att den här typen av examination utesluter några typer av uppgifter eller frågor som motsvarar, mestadels, de lägre nivåerna i Blooms taxonomi (Bild 3) (Bloom et al., 1956).

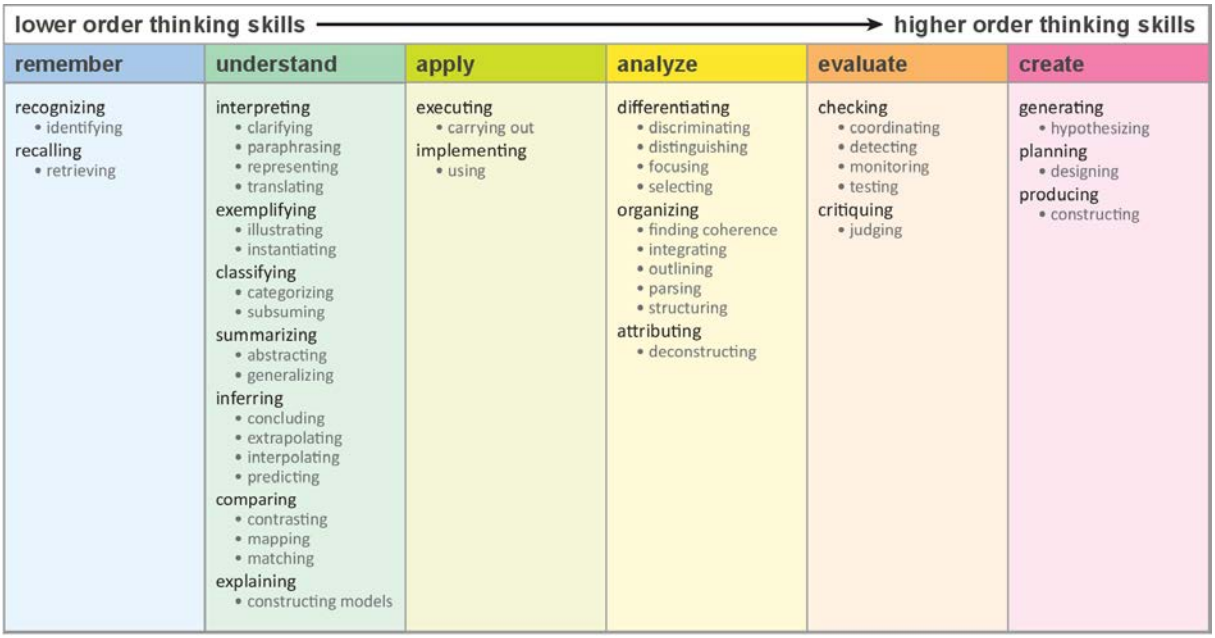

Bild 3: Dimensioner av kognitiva processer baserande på Blooms taxonomi (Anderson et al. 2000).

Uppgifter som har som mål att utreda om studenterna minns eller har förstått grundläggande termer eller koncept är inte meningsfulla i öppen bok-tentamen eftersom lösningen ofta kan kopieras från föreläsningsskriptet. Fokuset i den här examinationsformen ligger hellre på de högre kunskapsnivåerna (*analyze*, *evaluate* och *create*) (Elmgren & Henriksson, 2019). Det kan hjälpa att angripa problemet som identifierats, bland annat, i en studie som Trowald (1997) beskriver. Närmare 70 % av tentamina mäter kunskap bara genom att testa studenternas förmåga att kunna definiera begrepp, minnas, tillämpa och förklara fakta, samt att göra beräkningar som motsvarar nivå 1–3 (*remember*, *understand* och *apply*) i Blooms taxonomi.

Från lärarnas perspektiv är dessa uppgifter ofta enklare att formulera och granska men från studenternas perspektiv främjar sådana tentamina inte deras förmåga att applicera och transferera kunskaper som motsvarar högre nivåer av lärandet. Men det är dessa typer av uppgifter som ofta är rimliga i början av tentamen för att ge studenten möjlighet att ”komma in” i tentamen och att övervinna eventuell nervositet genom att kunna svara på enklare frågor.

Studien från Berggren et al. (2015), som genomfördes på KTH i Stockholm, visade att över 90 % av studenterna som deltog i en digital examination var positiva och kunde tänka sig att ha den nya typen av examination som standard i framtiden. Fördelar med digitala examinationer för studenterna omfattar reducerad stress under själva examinationen, skrivandet går lättare med ett tangentbord jämfört med en penna men också eftersom svaren lättare kan korrigeras, studenterna känner sig mer anonyma och den snabbare återkopplingen och bedömningen.

Men också för de som undervisar i kursen innebär den digitala examinationen fördelar. Berggren et al. (2015) uppskattar att 20 % av kurstiden måste användas för granskning och bedömning av konventionella examina och tiden kan halveras med hjälp av digitala examina. Det finns en viss inarbetningstid vid byten till nya online-system men denna krävs bara första gången innan rutin skapas. Särskild bedömning av tentamina blir enklare och mer rättvis

genom införandet av digital examination. Problem såsom oläsbar handskrift finns inte när texten skrivs digitalt och automatiserad bedömning kan hjälpa till att minska skillnader i bedömningen mellan olika granskare. Möjligheten till automatiserad bedömning finns inte för alla ämnen och alla typer av frågor (till exempel fritext eller skisser) men ofta kan möjligheterna skapas genom att forma om frågorna till, till exempel, flervalsfrågor eller frågor med givna svarsalternativ.

Det är svårt för en granskare att gå genom programkod och bedöma om den är korrekt i både syntax och semantik. Det finns ofta flera olika möjligheter att använda och kombinera anvisningar. Utöver det, är programmen ofta för komplexa för att kunna utreda manuellt om de producerar det önskade beteendet (Higgins et al., 2005). Det finns dock möjlighet att genomföra automatiserade tester för att kunna bedöma om ett program uppfyller vissa krav. Higgins et al. (2005) presenterar och jämför olika system (*computer-based assessment systems*) som kan användas för att stödja eller automatisera bedömningsprocessen.

Ett annat övervägande som måste göras i det här sammanhanget är tentamensstrukturen. Jevinger & von Hausswolff (2016) diskuterade för- och nackdelar hos två potentiella examinationsformer inom Java-programmeringskurser: en större uppgift i motsats till många mindre frågor eller uppgifter. Författarna beskriver hur båda formerna av tentamina bidrar till att uppnå kursmålen och argumenterar hur de kan kombineras för att kunna utreda olika typer av kunskaper.

5 Digitaliseringspotential av programmeringskurs DA343A på Malmö Universitet

Efter att ha presenterat och diskuterat olika metoder och verktyg för digital undervisning av programmering, analyserar det här kapitlet hur dessa ansatser kan användas i föreläsningar, laborationer och examinationer. Det sker också med hänsyn till kursmålen av den presenterade kursen DA343A på Malmö Universitet.

5.1 DA343A (Objektorienterad programutveckling, trådar och datakommunikation)

Kurs DA343A på Malmö Universitet är en obligatorisk kandidatkurs inom programmen för systemutvecklare och för datateknik och mobil IT, som är en högskoleingenjörsutbildning. Båda programmen ges vid fakulteten för teknik och samhälle och kurs DA343A ingår i den andra terminen. Förkunskapskraven för kursen, som förmedlar fördjupade kunskaper inom programutveckling, består av 9 HP från kurs DA339A (”Objektorienterad programmering”), en introduktionskurs för grundläggande kunskap inom programutveckling.

Den aktuella kursplanen för DA343A[4] definierar tio lärandemål uppdelat i tre grupper: *kunskap & förståelse*, *färdighet & förmåga* och *värderingsförmåga & förhållningssätt*. Under kunskap & förståelse finns fyra kursmål som syftar på att studenten ska visa teoretisk kunskap och förståelse om olika relevanta ämnen som till exempel utvecklingsprocessen för att ta fram programvara, designmönster och objektsamlingar. Kategorin färdigheter & förmåga innehåller praktiska kursmål, som till exempel att studenten ska kunna använda designmönster, implementera system samt dokumentera och granska programvara. Under värderingsförmåga & förhållningssätt finns det ett kursmål som kräver att studenten kan reflektera över för- och nackdelar hos alternativa lösningar eller implementeringar. Jämfört med Blooms taxonomi (se kapitel 4) täcker kursmålen alla dimensioner från att minnas och

[4] https://edu.mah.se/sv/Course/DA343A

tillämpa olika designmönster, att analysera och värdera programvara till att skapa nya lösningar.

För att kunna uppfylla alla kursmål, består kursen av fyra olika kursmoment: *föreläsningar, laborationer, workshops* och *grupprojekt*. Medan föreläsningarna syftar till att förmedla teoretisk kunskap och att lägga grunden för tillämpandet av programmeringen, är målet med de andra tre undervisningsformerna att möjliggöra det praktiska lärandet och att främja olika praktiska förmågor. Detta sker genom praktiska uppgifter och handledning under laborationerna, flera workshops som syftar på att analysera och lösa olika uppgifter i grupper och ett större grupprojekt som simulerar samarbetet inom ett programutvecklingsprojekt med hänsyn till yrkeslivet. Eftersom programutveckling kan jämföras med ett hantverk, är skapandet av rutiner en viktig del av utbildningen.

5.2 Lämplighet av metoderna för digitalisering av olika moment inom kurs DA343A

Utifrån presentationen av hur kurs DA343A genomfördes fram tills vårtermin 2020, innan behovet för digital undervisning uppstod, diskuteras i det här kapitlet lämpligheten av de olika metoderna för digitaliseringen av olika moment inom kursen samt utmaningar som kan uppstå. Hittills handlar det dock bara om teoretiska funderingar som ännu inte har omsatts och som förmodligen kommer att utprovas under kursen som ges under vårterminen 2021.

Av de tre kursmoment som analyserades i den här artikeln, verkar digitaliseringen av föreläsningar vara enklast att genomföra. Inspelning av undervisningsvideor om olika kursinnehåll eller digitala föreläsningar via en programvara för videokonferenser, är två lämpliga möjligheter som kan omsättas med enkla tekniska verktyg. Dock medför skiftet från konventionella till digitala föreläsningar också en förlust av dialog och återkoppling. Läraren saknar studenternas mimik, som kan vara en indikator för problem med förståelsen, men också möjligheten att direkt kunna reagera på återkoppling eller frågor genom att till exempel kunna förklara en teori på ett annorlunda sätt eller att ge fler exempel. Samtidigt måste studenterna ställa sina frågor via mejl och kan därför ofta inte profitera från de andra studenternas frågor. En lösning för det här problemet kan vara att använda ett online-diskussionsforum där studenterna kan ställa och diskutera sina frågor offentligt. Men tröskeln att ställa en fråga i ett online-forum kan också uppfattas högre eftersom det kan verka jobbigare.

En annan mellanväg kan vara föreläsningar genom en mjukvara för digitala möten (till exempel Zoom, Microsoft Teams eller BlueJeans) där läraren håller föreläsningen ”live”. Det ger möjlighet till återkoppling under tiden och inspelning samt uppladdning av föreläsningen i efterhand utan att distrahera studenterna. Inspelade föreläsningar och ytterligare videor hjälper vid presentation av olika potentiella angreppssätt för vissa problem och bidrar därigenom till uppnåendet av olika lärandemål, till exempel att kunna reflektera över alternativa objektorienterade lösningars för- och nackdelar samt att visa kunskap i olika relevanta ämnen.

I början av pandemin, när distansundervisning skulle genomföras utan förberedelsetid, var skiftet till videokonferenssystem ett enkelt sätt att möjliggöra lärandet på distans. De didaktiska anpassningar som krävs, är mindre jämfört med inspelningen av videor som inte ge studenterna möjligheten till följdfrågor eller återkoppling men inte hellre ge läraren möjlighet att kunna reagera på den. Med hänsyn till vårtermin 2021 planerar vi inspelning av korta videor (ca 10–15 minuter) om olika teman som kompletteras av en kortare digital föreläsning.

Här ska videorna förknippas med praktiska exempel och studenterna ska få möjlighet att ställa och diskutera personliga frågor.

Skapandet av integrerade online-miljöer för lärandet av programmeringsspråk är tidskrävande för läraren. Klassiska programmeringslaborationer består av flera uppgifter som ska lösas efter varandra i en datorsal där läraren och några studenter som kommit längre i sin utbildning handleder kursdeltagarna. I en digital övning måste läraren förutse vanliga fel som kan göras i varje uppgift för att kunna ge motsvarande återkoppling eller tips på ett automatiserat sätt. Här kan erfarenhet från fysiska laborationer vara nyttiga för att kunna uppskatta var problem kan uppstå och hur de kan lösas med respektive tips. Eftersom tids- och handledningsbehovet är ganska olika mellan studenter, kan erbjudandet av digitala övningar också bidra till att ge studenterna det individuella stöd de behöver. Det är också lärandemålet att kunna använda designmönster, objektsamlingar, trådar, strömmar och datakommunikation i programvara, som uppnås här.

En stor utmaning i digitalisering av kursen är skiftet från en salstentamen till digital examination. Det beror på att examinationen måste ta hänsyn till alla tre kursmålsgrupper. En skriftlig salstentamen är inte lämplig för att kontrollera eller utreda om alla kursmål blev uppnådda, till exempel att dokumentera programvara eller att implementera system. Det är därför det hittills har funnits tre olika bedömningsformer i kursen. Dessa är, vid sidan av tentamen, också praktiska inlämningsuppgifter och en större programutvecklingsuppgift som utreder praktiska skickligheter. Genom skiftet till digitala hemtentamina uppstår nya möjligheter att integrera uppgifter som utreder olika kunskapsformer som till exempel att skapa eller utveckla nytt enligt Blooms terminologi. På det sättet kan dessa tre bedömningsformer delvis kombineras igen. Kursmålet att kunna implementera system och att kunna använda vissa funktioner, kan utredas tillsammans med andra kursmål, till exempel att visa fördjupad förståelse för användningen av UML och att visa kunskap om trådar, när examinationen sker vid en dator. Studenten kan i så fall utveckla ett system som använder flera tråder för att lösa en uppgift, beskriva systemet med hjälp av ett UML-diagram och slutligen implementera systemet som en användbar programvara. Det motsvarar också den praktiska utvecklingsprocessen av programvara.

6 Sammanfattning

I den privata och individuella programmeringsutbildningen har användningen av digitala verktyg och metoder etablerats. Det finns många sätt att förmedla kunskap i både teori och praktik som också är lämpliga för användningen i den högre utbildningen. Från lärarens perspektiv kan det dock finnas trösklar: inarbetning i nya plattformar och programvaror, mer omfattande online-tentamina jämfört med skriftliga tentamina och förmågan att kunna förutse vanliga fel och problem för att kunna ge nyttig återkoppling. Samtidigt sjunker behovet av inarbetning för att lära sig nya verktyg på lång sikt och de nya undervisningsformerna ger nya möjligheter att styrka den konstruktiva läkningen mellan kursmålen och undervisningsmomenten. Sammanfattningsvis kan det konstateras att det redan finns många utmärkta online-resurser som kan användas och utmaningen är att integrera dem i undervisningen på ett konstruktivt sätt som skapar ett mervärde för studenternas lärande.

Litteratur

Ala-Mutka, K. M. (2005). A survey of automated assessment approaches for programming assignments. *Computer science education*, 15(2), 83-102. https://doi.org/10.1080/08993400500150747

Anderson, L.W., Krathwohl, D.R., & Bloom, B.S. (2000). *A Taxonomy for Learning, Teaching, and Assessing: A Revision of Bloom's Taxonomy of Educational Objectives*, Longman. ISBN 978-0801319037.

Ansarian, L., & Teoh, M. L. (2018). *Problem-based language learning and teaching: an innovative approach to learn a new language*. Springer. https://doi.org/10.1007/978-981-13-0941-0

Berggren, B., Fili, A., & Nordberg, O. (2015). Digital examination in higher education: Experiences from three different perspectives. *International Journal of Education and Development using ICT*, 11(3). http://ijedict.dec.uwi.edu/viewarticle.php?id=2037

Bloom, B. S.; Engelhart, M. D.; Furst, E. J.; Hill, W. H.; Krathwohl, D. R. (1956). Taxonomy of educational objectives: The classification of educational goals. Handbook. I: *Cognitive domain*. New York: David McKay Company. ISBN 9780679302117.

Cascaval, R. C., Fogler, K. A., Abrams, G. D., & Durham, R. L. (2008). Evaluating the benefits of providing archived online lectures to in-class math students. *Journal of Asynchronous Learning Networks*, 12, 61-70. https://eric.ed.gov/?id=EJ837515

Cavanaugh, J. (2005). Teaching online-A time comparison. *Online Journal of Distance Learning Administration*, 8(1). https://www.westga.edu/~distance/ojdla/spring81/cavanaugh81.htm

DAXX (2020). *How Many Software Developers Are in the US and the World?* https://www.daxx.com/blog/development-trends/number-software-developers-world

Elmgren, M. & Henriksson A.-S. (2019): *Universitetspedagogik*. Studentlitteratur. ISBN 9789144108353.

Fenton, N., & Bieman, J. (2014). *Software metrics: a rigorous and practical approach*. CRC press. ISBN 978-1439838228.

Francis, J. (1982). A case for open-book examinations. *Educational Review*, 34(1), 13-26. https://doi.org/10.1080/0013191820340102

Fredrikson, Ben (2018). *Ranking Programming Languages by GitHub Users*. https://www.benfrederickson.com/ranking-programming-languages-by-github-users/

Gerrevall, P. (2009). Om examination och betygsättning. I: Stigmar, M. (utg.) *Högskolepedagogik: att vara professionell som lärare i högskolan*. Liber. ISBN 978-9147093427.

Higgins, C. A., Gray, G., Symeonidis, P., & Tsintsifas, A. (2005). Automated assessment and experiences of teaching programming. *Journal on Educational Resources in Computing* (JERIC), 5(3), 5-es. https://dl.acm.org/doi/10.1145/1163405.1163410

Hrastinski, S. (2018). *Digitalisering av högre utbildning*. Studentlitteratur AB. ISBN 9789144119724.

Jevinger, Å., & Von Hausswolff, K. (2016). Large programming task vs questions-and-answers examination in Java introductory courses. I *2016 International Conference on Learning and Teaching in Computing and Engineering (LaTICE)* (pp. 154-161). IEEE. https://doi.org/10.1109/LaTiCE.2016.25

Laurillard, D. (2013). *Rethinking university teaching: A conversational framework for the effective use of learning technologies*. Routledge. ISBN 9781315012940.

Salanci, Ľubomír (2015). Didactics of Programming. *International Journal of Information and Communication Technologies in Education*, 4(3), 32-39. https://doi.org/10.1515/ijicte-2015-0012

Stigmar, M. (2009). Att utvecklas som universitetslärare. I: Stigmar, M. (utg.) *Högskolepedagogik: att vara professionell som lärare i högskolan*. Liber. ISBN 978-9147093427.

Paulin, Dan (2019). *Digital tentamen – Vad tycker studenter om att tentera med dator?* https://www.chalmers.se/sv/konferens/KUL/2019/Documents/Abstracts/Paulin%20Digital%20salstentamen%20KUL2019%20190109.pdf

Trowald, N. (1997). *Råd och idéer för examinationen inom högskolan*. Högskoleverket. ISSN 1400-9498.

Williams, L., & Kessler, R. (2002). *Pair programming illuminated*. Addison-Wesley Longman Publishing Co., Inc. ISBN 978-0201745764.