Moodle som stöd för aktivt lärande

Anna Axelsson och Sandra Nilsson, *Datavetenskap, LTH*

***Sammanfattning*—För att lära sig programmera måste studenterna själva vara aktiva och öva programmering på dator. Med ett omfattande kursinnehåll går det av resursskäl inte att göra all nödvändig övning obligatorisk därför har vi digitaliserat övningsuppgifter som tidigare varit på papper i ett försök att få studenterna att göra fler övningar på med en mer aktiv approach.**

Försöken har mottagits väl och en majoritet av studenterna menar att de lärt sig mer jämfört med vad de hade gjort på traditionella pappersövningar.

***Nyckelord*—Datavetenskap, programmering, spelifiering, aktivt lärande, Moodle, CodeRunner**

I. Introduktion

I en tid då digitalisering är viktigare än någonsin menar vi att LTH måste fortsätta att vara unikt med sitt campus med närhet till lärare och medstudenter. Samtliga programmeringsgrundkurser är uppskattade för sin goda lärartäthet och goda möjligheter till att fråga och diskutera med de som kan ämnet. Däremot håller vi ofta ganska stora kurser, från 70 till 240 studenter per kursomgång och då kan det också vara svårt att säkerställa att varje individ får tillräcklig övning.

Till aktivt lärande kan räknas alla metoder eller aktiviteter som engagerar studenter att vara aktiva i meningsfulla lärandeaktivieter. Det är viktigt att lärandeaktiviteterna designas kring viktiga lärandemål i kursen [1]. Kanske kan våra digitala plattformar hjälpa oss så att fler uppmuntras till aktivt lärande och får just den träning och hjälp som behövs.

Moodle är en fri lärplattform som används på många universitet runt om i världen [2]. Vi använder Moodle i flera av våra kurser för forum, övningar, distribution av material, hantering av kamratgranskning av uppsatser, enkäter och operativa utvärderingar mm.

Det finns många färdiga moduler till Moodle. Under det senaste året har vi börjat använda modulen CodeRunner som gör det möjligt för studenterna att interaktivt lösa mindre programmeringuppgifter online [3, 4]. Studenterna skriver kodavsnitt som kompileras och testkörs. De får snabb feedback där det framgår vad som fungerat och vad som misslyckats.

II. Pedagogisk frågeställning och hypotes

Det enda sättet för en student att lära sig programmera är att aktivt öva programmering under hela kursens gång. Taxonomin är inte nödvändigtvis lika linjär som i de traditionella taxonomierna eftersom programmering redan på ett tidigt stadium kräver att studenten tillämpar tekniken och skapar egna lösningar. I början sker det troligen genom att söka efter ledtrådar, härma tidigare lösningar och prova sig fram men för att uppfylla lärandemålen måste studenten förstå tillräckligt för att tillämpa och skapa egna lösningar, även på nybörjarkurserna [5].

För att säkerställa att alla övar har våra campusgrundkurser i programmering ett antal obligatoriska programmeringslaborationer. Laborationerna måste redovisas muntligen, alltid på plats i datorsal. De flesta studenter behöver dock mer övning än så för att knäcka koden. Vissa behöver öva lite, andra behöver öva väldigt mycket. Det är av resursskäl svårt att göra all nödvändig övning obligatorisk för alla, därför har vi tidigare år tillhandahållit frivilliga salsövningar där studenterna löst mindre övningsuppgifter på papper och jämfört sin lösning med ett lösningsförslag. Övningarna har engagerat en del studenter, men har samtidigt dragits med en del problem:

1) Vissa studenter har slaviskt tittat i facit till övningarna istället för att själva reflektera och de studenter som haft svårast för programmering har också varit minst benägna att faktiskt öppna en dator och testa själv

2) nybörjarstudenter finner det svårt att veta om just deras svar fungerar lika bra som i facit, vilket ibland hämmat den egna uppfinningsrikedommen (en programmeringsuppgift kan alltid lösas på många olika sätt) och

3) Det har varit svårt att få studenterna att göra alla frivilliga övningar varpå lärandet blivit lidande och resurser i form av salstid och övningsledare bitvis varit uppbokade förgäves. Mer om dessa problem nedan.

I år har vi bytt ut pappersövningarna i några kurser, för att istället låta studenterna öva kursinnehållet genom interaktiva övningsuppgifter på Moodle. Studenterna skriver kodavsnitt som kompileras och testkörs. De får snabb feedback och ser vilka testfall som fungerar.

Vi vill undersöka om vi kan råda bot på de tre problem som nämns ovan. Vår förhoppning är att dessa interaktiva övningsuppgifter ska göra studenterna mer aktiva än på vanliga pappersövningar. Med anknytning till problemen ovan tror vi att:

1) Det kan vara en fördel att studenterna faktiskt måste koda "på riktigt" även på övningarna, de som tycker att ämnet är svårt har också varit minst benägna att prova sig fram och löser därför gärna pappersövningar som kan te sig mindre skrämmande. I och med att den möjligheten försvinner kanske fler kommer öva i mer verklighetsnära situationer. Vidare finns det inget färdigt facit att titta på så fort man kört fast, istället tvingas studenterna tolka felmeddelande och ändra i sin programkod tills den fungerar. Men ett facit har inte bara nackdelar, ibland kan det hjälpa en medveten student att komma vidare. Därför tillhandahåller vi resurstider i sal för frågor och hjälp samt kontinuerlig support via e-mail för att hjälpa de som kör fast på uppgifterna. I vissa fall har vi även gjort filmer som förklarar vanliga problem och sådant som kan vara svårt för novisen att själv komma på första gången.

2) Detta sätt att lösa övningarna kan uppmuntra till

kreativitet då de inte hålls tillbaka av våra lösningsförslag. Studenterna kommer gå vidare med sin egen lösning istället för att, som på pappersövningarna, förkasta den helt om den skiljer sig för mycket från lösningsförslaget.

3) Det kan kännas mer angeläget att lösa programmeringsuppgifter när det inte är på papper (med hänvisning till ämnets natur). Men digitaliseringen av övningsuppgifterna ger oss också möjlighet att använda spelifiering (eng. gamification) för att locka studenterna att lösa fler uppgifter. Spelifiering innebär att man använder speldesignelement i andra sammanhang än i spel [6].

Varje övning består av relativt få frågor som gäller ett avgränsat område. Vi vill att studenterna ska känna att de "bockat av" ytterligare en sak, och varje övning ska inte kännas övermäktig att ta tag i. Moodle visar studentens individuella framåtskridande på kursen och varje student kan tydligt se hur många övningar som är avklarade. Nu gör vi även försök med en modul som ger studenterna poäng för varje avklarad övning och dessa poäng motsvarar en nivå (level) på kursen. Alla studenter kan se en anonym lista över hur deras kursare ligger till. Genomförda övningar gör att studentens level ökar och på så vis hoppas vi trigga ett fortsatt arbete med övningarna med hjälp av samma mekanismer som i vanliga datorspel.

En liknande användning av spelifiering med poäng och nivåer har provats med gott resultat i [7]. Där ledde spelifieringen till ökad motivation och aktivitet bland studenterna.

III. Resultat

Vi har provat konceptet på tre olika kurser som alla varit lite olika och visssa pågår fortfarande. På en nu avslutad kurs genomfördes en frivillig enkätundersökning gällande de interaktiva övningarna. Enkäten visar att en majoritet av studenterna trodde att de lärt sig mer via övningarna på Moodle än vad de skulle gjort med vanliga pappersövningar. Det är emellertid svårt att fråga en grupp studenter som, i detta fallet, läser sin första programmeringskurs och därför inte har något att jämföra med.

En student som läser om kursen skriver dock:

"*Vilket fantastiskt initiativ att ha uppgifterna här på moodle. Jag tog kursen för 4 år sedan och tar om den nu, vilken skillnad! Enormt mycket bättre nu och mer lärorikt. Starkt jobbat tjejer!*"

Dock efterlyser vissa studenter lösningsförslag vilket också kan tyda på att vi borde vara tydligare med möjligheten att fråga och kanske stärka resurserna med frågestunder och film ytterligare.

IV. Slutsatser och diskussion

Att lämna studenterna helt ensamma med interaktiva övningar skulle troligen motverka sitt syfte och få många att tappa programmeringslusten. Men med rätt stöd tycker vi att det verkar trigga ett mer aktivt lärande som stimulerar till att prova mera själv. Det är därför viktigt att vi stärker det interaktiva materialet med resurstider och filmer.

Vi har lagt ner en hel del tid på att formulera övningsuppgifterna och skriva testkoden i CodeRunner, men vi tror att det är en investering som lönar sig. Vi kan då rikta undervisningen till de som behöver hjälp just nu, istället för att boka upp lokaler för pappersövningar till samtliga studenter varje vecka. De som inte dyker upp hos oss får feedback via systemet, de får veta om deras lösning är rätt och i de mån de känner att de förstår vad de gör behöver de kanske inte fråga så mycket. En bonus är att vi ser exakt hur många övningar respektive student faktiskt gör.

Vi kommer även att undersöka sambandet mellan tentaresultat och hur många övningar som genomförts.

Framöver vill vi gärna utvärdera om systemet med utökad spelifiering (levels) kan trigga fler studenter att göra många övningar. Men vi vill också titta närmare på de påstådda baksidorna med spelifiering som kan vara att studenten lär sig mer då de inte har press på sig, möjligen på grund av att de då lägger mer tid på varje uppgift istället [8]

De interaktiva övningarna har inte gjort att vi har fått mindre studentinteraktion. Tvärtom har vi fått många frågor som också varit intressantare eftersom frågorna på pappersövningarna ofta handlade om lösningen i facit.

Vi kan konstatera att lärplattformar är viktiga även för campuskurser och det finns starka skäl att inte låsa fast sig till en och samma. Vi vill lyfta fram att olika lärplattformar kan ha olika fördelar, just denna möjligheten finns på Moodle och inte, vad vi känner till, någon annanstans.

Referenser

[1] M. Prince, Does Active Learning Work? A Review of the Research. *Journal of Engineering Education, 93: 223-231.* 2004

[2] MOODLE – lärplattformar (open source). https://moodle.org. Senaste access: 2018-10-30.

[3] CodeRunner – plugin till MOODLE. http://coderunner.org.nz. Senaste access: 2018-10.30

[4] R. Lobb and J. Harlow. Coderunner: A tool for assessing computer programming skills. *ACM Inroads, 7(1):47–51, Feb.* 2016.

[5] U. Fuller, CG. Johnson, T. Ahoniemi, D. Cukierman, I. Hernán-Losada, J Jackova, E Lahtinen,TL. Lewis, DM Thompson, C Riedesel, E. Thompson. Developing a computer science-specific learning taxonomy. *InACM SIGCSE Bulletin 2007 Dec 1 (Vol. 39, No. 4, pp. 152-170). ACM.*

[6] S. Deterding, D. Dixon, R. Khaled, and L. Nacke, "From game design elements to gamefulness: defining gamification*", In Proceedings of the 15th International Academic MindTrek Conference: Envisioning Future Media Environments, September 28-30, 2011, Tampere, Finland, ACM, pp. 9-15*.

[7] G. Barata, S. Gama, J. Jorge, D. Goncalves, "Engaging Engineering Students with Gamification", *2013 5th Int. Conf. Games Virtual Worlds Serious Appl.*, pp. 1-8, 2013.

[8] P. Carvalho, M. Gao, B. Motz and K. Koedinger. Analyzing the relative learning benefits of completing required activities and optional readings in online courses. *Proceedings of the 11th International Conference on Educational Data Mining. Buffalo, New York.* 2018.