Studentaktiva programmeringsföreläsningar

Rapport från pedagogiskt utvecklingsprojekt 2014.

Jody Foo, Anders Fröberg, Erik Berglund, Jalal Maleki Camilla Kirkegaard, Sah
Linköping University
{förnamn.efternamn}@liu.se

Abstract

- Mål: Att tillhandahålla en möjlighet för studenter att prova och experimentera med programmering under föreläsning
- Utveckling av Codela
- Olika sätt vi använt Codela på
- Fortsatt användning av Codela

1. Introduktion: Föreläsningar och laborationer i programmeringskurser

Många programmeringskurser som ges på Institutionen för datavetenskap vid Linköpings universitet har följande kursmoment: föreläsning, lektion och programmeringslaboration (i fortsättningen kallad för *laboration*). Programmeringsföreläsningar innehåller liksom föreläsningar i andra ämnesområden moment av att föreläsaren går igenom olika begrepp och teori. Syntax och nyckelord som används när man skriver kod (programmeringskod) gås också igenom då detta förrutom strukturen och uppbyggnaden av den kod man skriver också är väldigt viktig. Att förstå vad man ska skriva och hur datorn sedan tolkar detta, "tänka som en dator", är något som beskrivits som att man använder en mental *Notational Machine* \cite{boulay_difficulties_1986}.

Andra moment på föreläsningar som också förekommit i projektdeltagarnas kurser är "live-kodning", där föreläsaren programmerar på sin dator och studenterna har möjlighet att se vad som händer, samt direkt och anonym återkoppling från studenter med hjälp av studentresponssystem.

Något som återkommit med jämna mellanrum i kursutvärderingar på kurser som kursdeltagarna medverkat i har varit att det många gånger gått för lång tid mellan föreläsning (den teoretiska genomgången) och laborationerna (den praktiska övningen). Ett eventuellt problem med detta är att studenterna inte får möjlighet under föreläsning att upptäcka eventuella fel i sin mentala Notaional Machine. Det kan t.ex. vara så att de tror att de förstått ett visst begrepp eller en viss mekanism, men när de sedan får omsätta denna förståelse i praktiken på en laboration, märker de att deras menatala model inte stämmer överrens med verkligheten.

Syftet med detta projekt är att försöka hitta ett sätt att låta studenterna få möjlighet till praktisk återkoppling även på föreläsning.

1.1. Projektdeltagare

Pedagogiska medel för detta projekt söktes av följande personer vid institutionen för datavetenskap: Jody Foo (huvudsökande), Anders Fröberg, Erik Berglund, Camilla Kirkegaard, Jalal Maleki.

2. Projektets syfte

Målet med detta pedagogiska utvecklingsprojekt har varit att introducera en ny typ av lärmoment som integrerar studentaktiva inslag från datorlaborationer med lärardrivna inslag från traditionella föreläsningar. Som vi presenterat tidigare i denna rapport (se [#introduktion]), så har ett förekommande problem med ett traditionellt upplägg av en programmeringskurs varit det tidsmässiga avståndet mellan laborationer och föreläsningar.

Undervisning i ämnen som inte kräver speciell utrustning för att utföra vissa moment har haft möjligheten att kombinera olika lärandemoment i större grad än t.ex. programmeringsundervisning där en dator med en specialiserad programmeringsmiljö ofta behövts.

Syftet med detta projekt har varit följande:

1. Att *tillhandahålla en platform som gör det möjligt för studenterna att genomföra programmeringsövningar* på medhavd dator eller surfplatta utan några större förberedelser.
2. Att undersöka och få erfarenhet av *vilka förändringar i kursdesign, kursmomentsplanering och resursanvändning* som behövs eller krävs

på grund av införandet av denna nya lärande aktivitet, antingen som komplement eller som ersättning för existerande moment.

Ett sätt att rama in detta projekt är således att se det som en ansats att tillföra *Blended Learning* \cite{sharma_blended_2010} till programmeringsundervisning. Ett annat sätt att rama in projektet är att se det som en ansats till att skapa och tillhandahålla en miljö som möjliggör aktivt lärande på en programmeringsföreläsning. Vi ser också att en satsning som denna bidrar till ett aktivt lärande under föreläsningar. Tidigare studier har visat att detta, precis som i andra ämnesområden är något som har en positiv påverkan på studenternas lärande, se t.ex. \cite{mcconnell_active_1996}.

3. Metod: Överväganden och utveckling av systemet – Codela

Tidigare erfarenhet hos projektdeltagarna av att använda olika typer av IKT-inslag på föreläsningar och kurser föranledde kravet på att det tilltänkta systemet skulle kräva så lite förberedelser och administration *vid användning under föreläsning*.

Även om de flesta studenter har tillgång till en egen bärbar dator, har inte alla, av olika anledningar, installerat alla komponenter i den programmeringsmiljö som behövs för att kunna genomföra samma övningar som i datorlabbsalarna eller PULarna[1] som de kallas för på Institutionen för Datavatetenskap. Detta gäller speciellt på introduktionskurser i programmering. Olika kombinationer av operativsystem och andra installerade program på studenternas egna datorer innebär också att *en* uppsättning instruktioner inte räcker för att vägleda noviser att installera och konfigurera en programmeringsmiljö på sin egen dator.

Tre tekniska lösningsspår med olika för- och nackdelar övervägdes:

1. Lokal virtuell miljö: Skapande och distribution av virtuell miljö som körs på studenternas egen dator.
2. Fjärranslutning till virtuell miljö: Uppkoppling mot virtuell miljö som körs på en server.
3. Webbtjänst: Webbaserad platform som nås via en webbläsare.

[1] **P**rogram**u**tvecklings**l**aboratorier

3.1. Lokal virtuell miljö

En *lokal* virtualiserad programmeringsmiljö innebär att man kör ett program på sin *egen dator* som skapar en *virtuell dator* i den egna datorn. Istället att använda en separat fysisk dator, så skapar man alltså en virtuell dator innuti en fysisk dator. En mall för denna virtuella dator förbereds så att samma virtuella dator kan köras av vem som helst. Som alternativ för detta projekt hade det inneburit att studenterna inte skulle behöva installera eller konfigurera en programmeringsmiljö på sin egen dator, utan att en standardiserad programmeringsmiljö skulle kunna förberedas för att sedan distribueras till studenterna som kör den på en virtuell dator.

Detta alternativ för med sig en hel del flexibilitet då man i princip har möjlighet att förbereda samma miljö som finns tillgänglig i labbsalarna. En nackdel är att även om det är färre steg att utföra jämfört med att installera och konfigurera en hel programmeringsmiljö på sin dator, så är det fortfarande ett par steg som måste förberedas av studenten innan föreläsning. Om en förändring ska ske i konfigurationen, eller filer distribueras som ska användas under en föreläsning, behöver detta också förberedas innan föreläsningen, både av teknisk personal, av föreläsaren och av studenterna. Mallarna till de virtuella maskinerna kan också vara av en storlek på några gigabyte. Något som inte är ett lika stort problem som för bara några år sedan, men som fortfarande är en faktor som bidrar till ytterligare logistiska förberedelser.

3.2. Fjärranslutning till virtuell miljö

Ett alternativ till att köra den virtuella miljön lokalt, dvs på sin egen dator, är att köra de virtuella maskinerna på en server som studenterna når via internet. Samma fördelar som att köra en virtuell dator lokalt finns, men med mindre förberedelse. Denna lösning innebär också att vid uppdateringar av programmeringsmiljön, behöver inte studenterna själva göra något, utan det kan ske utan att de behöver vara aktiva. Nackdelarna med att fjärransluta till en virtuell miljö är att den servern som kör de virtuella datorerna måste vara tillräckligt kraftfull för att hantera det antal virtuella datorer som behövs. För att kunna tillhandahålla virtuella datorer till en klass på 50 studenter, behövs en eller flera servrar som har tillräckligt mycket datorkraft att köra dessa 50 virtuella datorerna.

Då projektet började fanns det planer på att tillhandahålla virtuella da-

torer till studenter som läste kurser på Institutionen för Datavetenskap, men detta initiativ var dock inte ännu i drift.

3.3. Webbtjänst

3.4. Utvecklingar som skett efter projektets start

- ThinLic

4. Metod: Användning av systemet i kurser

- beskrivning av olika kurser/studentgrupper vi provat det på
- beskrivning av olika sätt som vi använt Codela på

4.1. Användning i kursen 729G04 Programmering och diskret matematik

Kursen 729G04 är en av de första kurserna som studenterna på kandidatprogrammet för Kognitionsvetenskap läser under deras första termin. Det är även den första obligatoriska programmeringskursen av två på programmet. Studenterna är en relativt heterogen grupp med en bakgrund

4.2. Användning i kursen 729G26 Interaktionsprogrammering

4.3. Användning i kursen ...

5. Resultat: erfarenheter och insikter

- hur många kunde använda det?
- hur många har datorer på föreläsningar?
- användning utanför schemalagd tid?

6. Diskussion

- fortsatt framtida användning

- förbättringar
- använda Codela eller t.ex. jsfiddle?
- tillämpbarhet inom andra ämnen än programmering?