Hvordan veiledning kan bidra til problemløsning og begrepsbygging innenfor objekt-orientert programmering

Terje Samuelsen [1,2] og Jens Kaasbøll [2]

[1] Avdeling for Informasjonsteknologi, Høgskolen i Østfold [2] Institutt for informatikk, Universitetet i Oslo

Sammendrag

Selv om det har vokst fram en oppfatning om at studenter lærer programmering på laben og ikke i auditoriet, så vet vi lite om hvordan labveiledning bør foregå. I denne studien prøves det ut en alternativ måte å veilede på. Metoden er hentet fra Schoenfeld [13] for bruk i matematikken, der veilederen stiller 3 spørsmål som skal få studentene til å reflektere over sin framgangsmåte. Hos oss førte dette til at studentene raskt lærte å løse programmeringsproblemer og klarte å anvende objektorienterte begreper. Til forskjell fra tidligere forskning ble studentene raskt fortrolige med veiledningsformen. [1]

Innledning

Et meget høyt antall studenter har droppet kurset underveis eller strøket begynnerkursene i programmering ved universitetene. Flere tiltak for å rette på denne tilstanden har vært forsøkt. Blant annet er det prøvd å bruke eldre studenter for å assistere nye studenter ved laboratorium og klasseromsøvinger [2], la studenter evaluere hverandres kode [15], programmere i par [9], spesiell oppfølging av kvinnelige studenter [3] og av studenter som tar kurset på nytt [14]. Disse har alle har gitt positive resultater, men har ikke ført til de store forbedringene. En av grunnene til at så få studenter gjennomfører denne type kurs kan være at selve undervisningsoppleggene som anvendes ikke er de mest egnete.

Ut fra et konstruktivistisk perspektiv, er å lære noe å utbygge sin kompetanse basert på den en allerede har. Tradisjonell programmeringsundervisning, som ofte er forelesningsbasert, tar utgangspunkt i lærerens eller lærebokforfatterens forståelse og forsøker å formidle denne til studentene. Undervisningen risikerer dermed å bomme på studentenes forforståelse. Lærere som har observert begynnerstudenter som løser oppgaver har sett at etter et par uker, så sliter fortsatt noen studenter med første oppgave, mens andre synes det er kjedelig at undervisningen går så sakte fram. Faget er strengt kumulativt, det vil si at en må ha lært ukas tema for å forstå neste undervisningssesjon. Foreleserstyrt progresjon vil dermed føre til at studenter som henger etter får svært lite utbytte av forelesninger.

Banduras [1] mestringsteori sier: *"Perceived self efficacy* denotes people's beliefs about their capabilities to produce designated levels of performance that exercise influence over events that affect their lives." Hva som utvikler "Perceived self efficacy " eller mestringsfølelse hos folk kommer i hovedsak fra fire hovedkilder. Det mest effektive er å bygge en sterk følelse av å beherske gjennom vellykkede eksperimenter. Suksess bygger en robust tro på ens evne til å mestre noe. Fiasko underminerer den, spesielt hvis fiasko skjer før mestringsfølelsen er blitt litt etablert. Den andre måten å kreere eller styrke mestringsfølelsen er gjennom å observere rollemodeller. Det å observere disse, styrker egen tro på at en har kapasitet til å mestre tilsvarende aktiviteter. Det tredje er overtalelse fra andre, dette styrker folks tro på at de har det som

[1] *This paper was presented at the NIK-2007 conference. For more information, see //www.nik.no/.*

skal til for å få suksess. Dette fordi de mobiliserer mer innsats og utholdenhet når problemer oppstår. Den fjerde måten å styrke mestringsfølelsen på er å redusere folks stressreaksjoner og negative opplevelser [1].

En forelesning som studenten forstår lite av, og som også medstudentene sier var ubegripelig, har allerede ødelagt de to førstnevnte kildene for styrking av mestringsfølelsen. Når studenten deretter forsøker å løse ukas oppgave på laben, og den første tilbakemeldingen er:

Syntax error: Missing {

så har hun også fått en negativ reaksjon, dvs. svekket kilde fire.

Veiledning på laben eller i klasserommet er vanlige undervisningsformer i etterkant av forelesning. Gjennom veiledning som tar utgangspunkt i studentens arbeid kan læreren tilpasse sine forklaringer og spørsmål til studentens nivå, slik at studenten klarer å løse oppgavene og dermed styrker sin mestringsfølelse gjennom suksess.

Trening på laben er nødvendig for de aller fleste studenter for å oppnå programmeringsferdighet, men å lære programmering fordrer også en forståelse av begreper og prinsipper. Det hjelper lite at en student løser hundre oppgaver ved å kopiere og endre koden en smule hvis hun ikke forstår begrepene programmeringsspråket, prinsippene for programutførelse og hvordan man bestemmer funksjonalitet og setter sammen et program for å løse et gitt problem innen et domene. Forelesninger har ofte presentasjon og forklaring av teori som sitt siktemål, og tidligere forskning om læring av ferdigheter basert på prinsipper har vist at følgende rekkefølge generelt er å foretrekke [8]:

1. Presentasjon av teori
2. Presentasjon av eksempler
3. Oppgaveøvelser
4. Diskusjon med veileder om oppgavene
5. Spørsmål og hjelp på et seinere stadium etter at undervisningen er over.

Så vidt vi vet har ingen forskning blitt utført som kan vise om denne rekkefølgen også er den beste for programmeringslæring. Den stemmer overens med hvordan programmeringsutdanning vanligvis organiseres, men som tidligere nevnt er resultatene ofte dårlige, og for lang tid brukt til forelesninger kontra veiledning kan være en årsak.

Også forskning om hvordan veiledning kan utføres og hvilke resultater den gir er mangelvare, så denne studien sikter mot å evaluere en spesiell veiledningform med hensyn på læring av programmering og objektorienterte begreper. Fordi en ukjent veiledningsform også utgjør noe som må læres, så vil studien også vurdere studentenes mestring av denne formen.

Det ble gjennomført en eksperimentundervisning i objektorientert programmering med 6 begynnerstudenter over 2 ½ dag. Deler av studentarbeidet ble videofilmet og materialet ble analysert kvalitativt for å finne spor etter hvordan lærernes aktiviteter hadde påvirket studentenes mestring av programmering og av de objektorienterte begrepene.

Veiledning

I mangel av veiledningsforslag innen informatikk, så ble det lånt en metode som er utviklet av Schoenfeld [13] for undervisning i problemløsning innefor matematikk. Metoden fokuserer på å bygge innsikt i temaer, noe som er i god overenstemmelse med en av hensiktene med innføringskurs i programmering. Schoenfeld skiller klart mellom teknikker for problemløsning som kan anvendes ved rutineøvelser hvor

problemkonteksten forteller studenten hvilken teknikk han skal anvende (for eksempel hvilken matematisk formel), og arbeid med det han omtaler som virkelig problemløsning hvor problemkonteksten ikke forteller studenten hvilken teknikk vedkommende skal velge.

Schoenfelds resultater er basert på det som omtales som problemløsningsklasser som jobber med tekstoppgaver. Selv om programmering ikke er det samme som matematikk, så har tekstoppgaver i matematikken noen av de samme egenskapene som programmeringsoppgaver: Studentene får utdelt en tekst som spesifiserer oppgaven, mens løsningen lages ved hjelp av formaliseringer. Løsningsprosessen går også i flere trinn i begge typer oppgaver, og studentene må selv finne løsningsmetoden. Studentene er delt inn i grupper på tre eller fire personer. Mens studentene løser tekstoppgaver driver læreren veiledning.

Idéen bak opplegget er at studenter ofte kjører seg fast i løsning av matematikkoppgaver, og de fortsetter som oftest på den kursen som de staket ut innledningsvis uten å reflektere over om de nærmer seg målet eller om de burte starte forfra med en annen strategi. Ved å stille tre spørsmål til studentene, ønsker Schoenfeld å få dem til å reflektere over sitt eget arbeid og hvor det leder hen. Han brukte de følgende tre spørsmålene og advarte studentene med at han ville kunne stille dem disse når som helst:

- Hva (eksakt) er det du gjør? (kan du beskrive det presist?)
- Hvorfor gjør du det? (Hvordan passer det inn i løsningen?)
- Hvordan hjelper det deg? (Hva vil du gjøre med resultatet når du får det?)

Forfatteren er meget klar på at det er et viktig poeng at studenten er kjent med at det er disse spørsmål som vil bli stilt i en veiledningssituasjon. Dette for at settingen ikke skal oppleves som en overraskelse og at de skal være kjent med at spørsmålene skal stimulere til kritisk tenking og lede dem til å lage et løsningsforslag. Veiledning med disse spørsmålene ga den klare effekten at studentgruppene stoppet opp og diskuterte løsningsstrategien sin, og dermed klarte alle gruppene oppgaver som grupper uten veiledning kjørte seg fast på.

Med bakgrunn i hans resultater og likheter mellom egenskaper knyttet til tekstoppgaver innenfor de to disipliner ble de tre spørsmålene valgt som veiledningsform for eksperimentet.

Begrepsforklaring

Forståelse av objektorienterte begreper kommer ikke med morsmelken, derimot vil mange begynnerstudenter allerede ha en rik forståelse av begrepene klasse, objekt, metode etc., Som vist av bl.a. Pea [11] bygger studentene sin forståelse av programmeringsbegrepene på sin tidligere forståelse av de samme ordene, noe som ofte fører til tvetydigheter og feilaktige oppfatninger.

En forelesning vil kunne presentere programmeringsbegrepene entydig, men en foreleser vil ikke så lett oppfatte dersom studentene assosierer klassebegrepet med for eksempel en skoleklasse. Foreleseren har i en slik situasjon mindre mulighet for å bekrefte og korrigere studentenes eventuelle usikre begrepsoppfattelse. At det finnes et omforent begrepsapparat er viktig, siden begrepene til sist skal kodes og brukes i programmet. I følge Eckerdal og Thuné [4] er det også av største viktighet i det første programmeringskurset at studentene på et tidlig stadium for en konsistent forståelse av sentrale begreper som objekt og klasse.

Intensjonen med å bruke de tre spørsmålene var at vi skulle knytte disse til sentrale objektorienterte begreper, istedenfor å rette dem mot selve programkoden.

Vi valgte å bruke begrepene: Objekt, klasse, sub-klasse, modularisering og gjenbruk. Vårt argument var ønsket om å stimulere studentenes fokus på de sentrale begrepene og "That focusing on these crucial aspects in teaching can help the students to gain a good understanding, and thus avoid different kinds of misconceptions" [4].

Strategien i didaktikken var basert på å introdusere begrepene via en minimal presentasjon med etterfølgende diskusjon i plenum. Så ble det delt ut (typisk 3-5 linjer) kort skriftlige forklaringer som også inneholdt en oppgave som studentene skulle programmere i Java. Oppgavene var å programmere en robots bevegelse i et kart i verktøyet Karel J og litt Robocode. Til den første aktiviteten fikk studentene en kildekodemal.

Metode

Aktuelle kandidater til eksperimentet var studenter som var tatt opp på matematisk/naturvitenskapelig fakultet ved Universitetet i Oslo og som skulle ta INF1000 Innføring i programmering. Vi sendte ut et tilbud til et randomisert utvalg om å delta på et 2 ½ dagers introduksjonskurs i programmering. 20 studenter meldte seg på og 6 av disse, alle menn omkring 20 år, dukket opp. Det var 2 instruktører, inkludert førsteforfatteren av denne artikkelen, som foresto kursgjennomføring i en terminalstue hvor studentene skulle sitte i par å jobbe. I tillegg var det to observatører som fulgte eksperimentet. Innsamling av data ble gjort ved notater og videoopptak (2 siste dager). Opptakene dekker diskusjoner i plenum samt arbeidet til to grupper. Studentene gav tillatelse til at videoopptakene kunne brukes i forskningsøyemed. En student reserverte seg mot å få ansiktet med på opptaket, men utover i kurset forsvant denne reservasjonen.

Siden vi primært ønsket å studere studentenes opplevelse og reaksjoner på opplegget, er det brukt en kvalitativ tilnærming [6], [7] med et interpretivt perspektiv [10]. Det var planlagt å ha en kvantitativ sammenligning mellom våre studenter og de øvrige som skulle ta eksamen ved semesterslutt, men med så få deltagere (6) på vårt kurs ville den eksterne validitet av dette resultatet bli for svak.

Resultat og analyse

Programmering

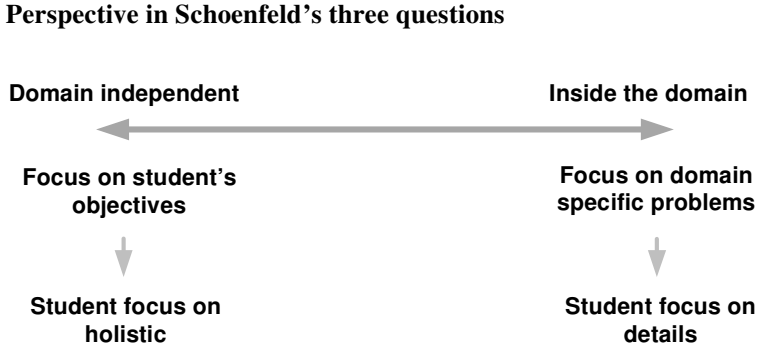
Først et eksempel på at de 3 spørsmålene hadde den ønskede effekten. I en oppgave skulle studentene få roboten til å plukke opp en piper langt inne i en labyrint som er en kvadratformet spiral. En gruppe laget en kode som flyttet en robot 8 skritt til høyre, snudde den til venstre, flyttet den 8 skritt oppover og instruktørens kommentar var: "Hvorfor gjør dere dette akkurat sånn? Hvordan passer dette inn resten av programmet?" De begynte å forklare hvordan de tenkte løsningen og oppdaget selv at roboten bare skulle gå åtte skritt rett frem tre ganger før antall skritt ble redusert. En foreslo å bruke en løkke som ble redusert med en for hver gjennomgang. Så sa en annen: "Vi kan bruke den testen om den har veggen foran seg, og så snu til venstre." De prøvde å implementere dette, da ropte gruppen på instruktøren: "Skal vi regne med at roboten vet at den skal svinge til venstre eller vet den ingen ting?" I dette ligger det at de referer til en aktivitet hvor de skulle forestille seg at roboten ikke kan se kartet i fugleperspektiv. Gruppen endte opp med å først lage en versjon hvor roboten alltid

svingte til venstre, for så å forbedre programmet med å sjekke til høyre, rett frem og til sist til venstre.

For å illustrere noe av vanskeligheten med å formulere spørsmålene, så følger et eksempel fra en oppgave som besto av å programmere roboten til å bevege seg i en trapp. Oppgaven skal lede til at det lages en modul for et trinn og så skal denne gjentas flere ganger. En gruppe fikk ikke roboten til å bevege seg slik gruppen håpet. Instruktørens første spørsmål til gruppen var ”Hva gjør programmet?” Studentene startet da med en gjennomgang av koden hvor de fortalte hva de enkelte setningene gjorde. Etter en tid avbrøt instruktøren med: ”Hva vil dere at objektet skal gjøre, og hvilke deler/moduler vil dere bruke for å få til det?” Tanken med spørsmålet ”Hva gjør programmet?” var at studentene skulle forklare hva programmet fikk objektet på skjermen til å gjøre. Vi kan se at spørsmålet inviterer studentene til å respondere med å beskrive koden i de enkelte setningene. Dette siden setningen retter seg mot programmet i motsetning til Schoenfelds form, som retter seg mot hva studenten gjør og ønsker å gjøre. Det er grunn til å tro at dette er et eksempel på at dersom studenten oppfatter at spørsmålet retter fokus mot syntaksen, så vil studentens fokus også være det, noe som er i overensstemmelse med mesterlære [5]. Ser vi på formen instruktøren brukte etter å ha avbrutt deres forklaring, ser vi at formuleringen er mer lojal ovenfor formen til de tre spørsmålene.

Etter at instruktøren hadde avbrutt deres forklaring fikk spørsmålet en form som rettet seg direkte mot hva studentene ønsket at objektet skulle gjøre. Deres forklaring beskrev først de ønskede bevegelsene til objektet, så startet de med å beskrive de enkelte elementene som de brukte. Under dette sa en av studentene: ”Roboten gjør ikke sånn, den går i veggen, hvorfor det?” Da sier den andre ”Ja, men, vi må snu den før den starter på neste modul” Problemet var at førstegangsutførelse av modulen hadde som utgangspunkt at roboten startet med retning nord og avsluttet med retning vest. Dette medførte at neste gang modulen skulle utføres fortsatte roboten mot vest og traff en vegg.

Det å formulere spørsmål fra domenet matematikk som Schoenfeld bruker, til vårt domene som er programmering av datamaskiner, kan være en utfordring. Tradisjonelt har det vært slik at når studenter skal lære objektorientert programmering, så har det vært naturlig at fokus hos undervisningspersonalet og studenten har vært mot fagspesifikke problem. I Schoenfelds eksperiment tolkes hans formuleringer til å rette seg mot at studenten skal uttrykke hva vedkommende ønsker å oppnå med sitt forslag. Modellen i Figur 1 illustrerer sammenhengen mellom bruk av fokus i spørsmålene og hvilken studentreaksjon dette kan bidra til.



Figur 1 Spørsmålfokus og studentrespons [12]

Utgangspunktet er at utformingen av spørsmålene i varierende grad kan tilpasses det aktuelle domenet. Dette illustreres av den todimensjonale matrisen. Teksten under den lange linjen beskriver hvilken fokus teksten i spørsmålene kan ha. De små pilene illustrerer teksten: ” Kan lede til at”.

Begge eksemplene som er omtalt tidligere er illustrasjon på at domeneuavhengige formuleringer kan lede til at studentene i sin forklaring prøvde å begrunne og se sin egen kode i en helhet i løsningen fremfor å beskrive den enkelte kodelinje. Med andre ord fokus på struktur og vurdering i løsningen.

Selv om vi under planleggingen var enig om at fokus i veiledningen skulle ligge på de tre spørsmålene, dokumenterer videoopptakene at i praksis ble det litt annerledes. Spesielt er det fra veiledningen under aktivitetene, at det kan registrere at fokus dreier fra det å veilede på begrepsforståelse ved hjelp av de tre spørsmålene, til fokus mot kompetanse på programsetningenes syntaks.

Vi tar tre representative formuleringer som ble brukt av instruktørene. Den første er: ”Hvorfor er testbetingelsen din slik?” Denne fikk som svar:” Er det ikke sånn det skal være i en If-setn?....skal det være en else også?” Her kan det virke som studenten legger til grunn at det er en feil i setningen og foreslo noen muligheter uten først å evaluere koden opp mot hva som var hensikten. Til leserens informasjon kan det sies at testen fungerte, men hadde en unødvendig kompleksitet som gruppen i samarbeid med veileder fikk forenklet. En gruppe hadde brukt en while setning mens instruktøren oppfattet koden slik at det skulle være en If-setning og sa: ”Hvorfor bruker du while?”. Den ene studenten henvendte seg til den andre på gruppen og sa: ”vi skulle vist ha brukt en if”. Svaret indikerer at gruppen hadde to alternativ, men valgt feil. De gjorde ingen evaluering av sitt produkt når de fikk spørsmålet. De bare konstaterte at de skulle ha valgt andre alternativet selv om det faktisk kunne være slik at instruktøren ikke hadde innsikt i hva studentene ønsket å gjøre. Det tredje eksemplet er: ”Kan du bruke else i en while setning?”. Dette er en formulering som direkte henleder mot at det ikke skal være en else eller det kan være et hint om at de kanskje skulle brukte en if setning. Svaret fra gruppen var: ”Å, det skal være en if”. Med denne rettelsen fikk de fortsatt ikke koden til å fungere slik de håpet. Etter litt veiledning og arbeid kom de frem til at de skulle bruke en while løkke med en if – else konstruksjon.

Formen på spørsmålene i de tre eksemplene ovenfor påpeker bare feil uten å motivere studentene til å evaluere produktet opp mot hensikten med koden. Vi hevder at dersom veileder hadde brukt de tre spørsmålene, kunne det i større grad ha bidratt til at studentene generaliserte kunnskapen. Dette kan også ha bidratt til å forsterke mestringsfølelsen hos studentene ved at de kunne finne feil og korrigere uten slike klare hint som eksemplene illustrerer.

Det å bruke de tre spørsmålene kan gi studenten en opplevelse av at veileder ikke vil svare på det problem som vedkommende formulerer. Eksempel på dette er at en gruppe i en tidlig aktivitet pekte på skjermen og sa: ”Vi er usikker, skal vi skrive: If front is clear, move?” Instruktøren svarte med to helt andre spørsmål: ”Hva vil dere at objektet skal gjøre? Hva er hensikten?” Dette er en situasjon som kan få enkelte studenter til å føle at de ikke får svar på det de ønsker. Dette kan komme av at studenten er mer resultatorientert enn opptatt av å forstå sammenhenger på et mer overordnet nivå.

Studentenes reaksjon på instruktørens spørsmål var et smil og: ”e....e....den (roboten) skal ikke treffe veggen. Hvis det ikke er klart skal den snu.” Instruktør: ”Hvordan stemmer det med koden? Har dere gjort noe før som kan hjelpe dere?”

Gruppens svar kan tyde på at de vet hva roboten skal gjøre og de er høyst sannsynlig bare usikker på syntaksen i kodesetningen, så de prøver å få korrekt kode fra instruktøren, heller enn å prøve å bygge på tidligere kunnskap. I denne og lignende

situasjoner kunne vi ikke registrere noen negative reaksjoner, bare litt smil og at de prøvde å omformulere forespørselen.

Hittil har vi fokusert på kommunikasjonen mellom student og veileder. Resultatene viser at studentene også brukte de tre spørsmålene internt i gruppen under arbeidet. En student sa under oppsummeringen: ” Når dere har snakket med oss er det lettere å fortsette og prate om de samme tingene”. En illustrasjon av dette er fra en gruppe som arbeidet med en aktivitet: ”Hva gjør egentlig roboten i den modulen?” Her spør den ene studenten den andre i gruppen om å få en utdyping av det roboten gjør, istedenfor å be om en beskrivelse av koden. Når den andre studenten svarer beskriver han det bevegelsesmønster som koden fremskaffer hos roboten, fremfor å beskrive selve kodesetningene. Et annet eksempel: ”Hvorfor bruker du modulen fra ur_robot, istedenfor fra ur_robot2?” Her peker studenten på skjermbildet hvor koden vises. Formen på spørsmålet indikerer at han ønsker en beskrivelse av forskjellen på tjenestene (metodene) i klassene opp mot problemet. En annen fra samme aktivitet: ”Kan den klassen hjelpe oss?” Kommentaren kom mens en gruppe lette etter klassedefinisjoner som de hadde brukt tidligere i kurset.

Under en av oppsummeringene, kommenterte studentene at spørsmålene hjalp dem til å tenke selv og som en sa: ”…tvinger oss til å overføre kunnskap fra en situasjon før, til det vi holder på med”. I dette og lignende uttalelser legger vi at de måtte vurdere tidligere utførte aktiviteter, og prøve å finne noe som kunne overføres til den aktuelle situasjonen, noe som kan bekreftes av denne uttalelsen: ”Kan den klassen hjelpe oss?”. Dette ”noe som kunne overføres” ser jeg av observasjonene også inneholder element av klipp og lim, samt begrepsanvendelsen.

Det siste eksemplet er også en illustrasjon på at studentene særlig anvendte de tre spørsmålene der studentene opplevde at de hadde litt alvorlige problemer, som for eksempel, mistanke om at de var på feil spor. Dette var jo nettopp et av målene med å anvende de tre spørsmålene, at studentene skulle motiveres til å være kritisk til sine egne forslag.

Veiledningsformen

Flere ganger under kurset fikk vi tilbakemelding fra studentene at de var fornøyd med at vi brukte våre tre spørsmål og prøvde å få dem til å tenke frem forslag og forbedringer selv. Disse tilbakemeldingene er i overensstemmelse med våre observasjoner. Kommunikasjonen ble oppfattet som enkel og grei å forholde seg til. Den vesentlige årsaken til dette kan være at begge parter hadde en viss felles forståelse av hva veilederne kom til å kommentere. Det å ha de tre spørsmålene, gjorde at instruktøren kunne fokusere på å vurdere gruppenes løsningsforslag, knytte dette til begrepsforståelsen som forslaget indikerte, eller som studentene gav muntlig uttrykk for. Dermed behøvde instruktøren ikke antyde noe konkret løsning.

At studentene i løpet av 2 ½ dag ble komfortabel med de tre spørsmålene og forberedte sine svar før de tok kontakt, avviker helt klart fra Schoenfeld sine resultater. Han fremfører at det kan gå over halve semesteret før dette er situasjonen.

Det er flere mulige årsaker til den raske progresjonen. En mulig årsak er at vi hadde små avgrensede aktiviteter som bare tilførte et nytt problem. Vi hadde også hyppig kontakt med gruppene. Studentgruppa var liten og dermed lettere styrbar. Videre kom alle studentene inn i en ny situasjon, slik at de hadde lagt igjen tradisjoner og handlingsfellesskap på sin forrige skole og forventet noe nytt siden dette var første dager på et nytt undervisningssted.

Det siste peker mot at det er mulig å initiere en ny praksis for ferske studenter, men at det bør gjøres så snart som mulig etter at studentene begynner på høyere utdanning.

Begrepsbygging

At vi i denne studien valgte å bruke sentrale begreper innenfor objektorientert programmering (Objekt, klasse, sub-klasse, modularisering og gjenbruk) for å studere i hvilken grad studentene utviklet begrepsforståelse viste seg å fungere godt sammen vår veiledningsmetode. Ut i fra det konstruktivistiske prinsipp ble studentene presentert for et begrep om gangen slik at de kunne bygge forståelsen av et begrep på kunnskap de allerede satt inne med. Prinsippet ble omtalt av studentene som at det gjorde arbeidet med det aktuelle læringsmomentet enklere. En grunn til dette kan være at alle hadde en felles begrepsoppfattelse når de diskuterte i gruppen eller i plenum, en annen at dersom de opplevde at de hadde to problemer var det en indikasjon på en svakhet i løsningsforslaget.

Vårt valg om enkelte ganger å anvende begreper som studentene i veiledningssituasjonen oppfattet som synonyme eller forklarende fikk en positiv mottagelse av studentene, for eksempel: ting (objekt), mal (klasse), del (modul) og arv (sub-klasse). Dette var en anvendelse av ord som var mer styrt av studentenes erfaring enn av krav til akademisk presisjon. Denne formen å kommunisere på beskrev studentene som nyttig og ikke forvirrende selv om de senere fikk en mer presis definisjon av begrepene.

I løpet av kursets 15 timer lærte studentene å løse enkle problem ved å lage Java program. Det at vi la opp til mestringsfølelse hos studentene underveis gav de utrykk for var motiverende for videre læring av programmering, dette også støttes av våre observasjoner. Når det gjelder læring av begrepene avdekket observasjonene at resultatene ikke var uniforme, og vi går derfor gjennom de enkelte begreper.

Objektbegrepet var det første begrepet vi tok opp og som det fremgår av eksemplene i denne artikkelen samt av våre observasjoner var studentene nokså umiddelbart i stand til å forstå hvorledes begrepet anvendes i programmering. Det var også et begrep som studentene hadde en klar og konsekvent anvendelse av under kurset. Når det gjelder begreper som klasse og arv kan vi konkludere med at studentene gjennom kurset utrykte en konseptuel forståelse for begrepene, men studentene hadde et problem med at de glemte hvilken klasse som hadde hvilke egenskaper. Den primære årsaken til dette ligger sannsynligvis i at vi brukte klassebetegnelser som lignet for meget på hverandre, for eksempel Robot, Ur_Robot og Ur_Robot2. Når studentene i tillegg laget egne subklasser som bygget på disse og med navn som lett kunne forveksles, er det ikke å undres over at dette av og til kunne være et problem. Dette indikerer at et innføringskurs bør ha en klarere struktur med hensyn på objekt og klassenavn.

Begrepene modularisering og gjenbruk klarte studentene raskt å beskrive og bruke i muntlig kommunikasjon, men å abstrahere og generalisere kunnskapen over i arbeidet med andre løsninger og situasjoner var et større problem enn vi hadde forutsatt. Resultatet indikerer også at opplegget burde ha bygget denne type erfaring i flere trinn, samt at disse to begrepene representerer begreper som krever mer erfaring før studentene behersker anvendelse i programmering.

Konklusjon

Veiledningsmetoden med å bruke tre spørsmål ga et vesentlig bidrag til at studentene klarte å løse mange programmeringsproblemer i løpet av det 15 timers introduksjonskurset. Etter korte forklaringer med etterfølgende diskusjon og

oppklaringsrunder under veiledningen lærte de å anvende begrepene objekt, klasse og sub-klasse i anvendelse av programmeringsspråket Java. Metoden er bedre egnet dersom perspektivet i spørsmålene er domeneuavhengig, enn når spørsmålene har en form som er nært knyttet til programmeringsdomenet. I motsetning til Schoenfeld [13] som poengterer at det kan ta over halve semesteret før studentene blir fortrolige med veiledningsformen, ble studentene i vårt eksperiment fort fortrolige med veiledningsformen.

Resultatene peker mot at labveiledere bør be studenter forklare hva de har gjort og hvordan dette er ment å bringe dem nærmere det produktet de skal lage. De bør også ta opp begreper til samtale under veiledningen.

Referanser

[1] Bandura, Albert; (1998) *Encyclopedia of mental health* (Red: Fredman, H.) San Diego: Academic press

[2] Chase J. D. og Okie E. G.; (2000) Combining Cooperative Learning And Peer Instruction In Introductory Computer Science. *SIGCSE Bulletin* 32, 1, 372-376

[3] Craig A.; (1998) Peer Mentoring Female Computing Students-Does it Make a Difference? *Proceedings of the Third Australasian Conference on Computer Science Education,* ACM, 1, 41-47

[4] Eckerdal, Anna and Thuné, Michael; (2005) ITcCSE'05, June 27-29 2005 Copyright 2005 ACM 1-59593-024-8/05/0006

[5] Nielsen, Klaus og Kvale, Steinar (red, 1999) *Mesterlære, læring som sosial praksis*. Ad Notam Gyldendal

[6] Grønmo, Sigmund; (2004) *Samfunnsvitenskapelige metoder*. Fagbokforlaget

[7] Halvorsen, Knut; (2003) *Å forske på samfunnet,* 4. utgave. Cappelens Akademisk Forlag

[8] Joyce, Bruce and Beverly Showers (1980) Improvising Inservice Training: The Messages of Research. *Educational Leadership.* Feb., 379-385

[9] McDowell C.; Werner L.; Bullock H. og Fernald J.; (2002) The Effects of Pair-Programming on Performance in an Introductory Programming Course. *SIGCSE Bulletin* 34, 1, 38-42

[10] Myers, Michael D. og Avison, David; (red, 2002) *Qualitative Research in Information Systems.* SAGE Publications Inc. Thousand Oaks, CA

[11] Pea, Roy (1986) Language-Independent Conceptual "Bugs" in Novice Programming. *Journal of Educational Computing Research* 2 (1) 25-36

[12] Samuelsen, Terje; (2007) *Hvordan veiledning kan bidra til begrepsbygging innenfor OOP* Cand. Scient oppgave. Universitetet i Oslo, DUO http://wo.uio.no/as/WebObjects/theses.woa/wa/these?WORKID=56755

[13] Schoenfeld, Alan H.; (1992) Learning to Think Matematically: Problem Solving, Metacognition, and Sense Making in Mathematics, i (Red.)Grouws, Douglas *A. Handbook of Research on Mathematics Teaching and Learning* Macmillan Education Ltd. New York, 355-358

[14] Sheard J. og Hagan D. L.; (1999) A Special Learning Environment for Repeat Students, *Proceedings of ITiCSE '99, Cracow*

[15] Zeller, A.; (2000) Making Students Read and Review Code. *SIGCSE Bulletin* 32, 3, 89