Bruk av PC-sal under eksamen i videregående programmering

Erlend Tøssebro
Institutt for Data og Elektroteknikk
Universitetet i Stavanger

Sammendrag

Faget "BID100, videregående programmering" er obligatorisk for studentene ved dataingeniørlinja ved universitetet i Stavanger (UiS). I dette faget skal studentene lære seg ulike teknikker innen objektorientert programmering. Det å programmere med penn og papir er svært ulikt det man gjør i praksis, samt at mange studenter synes det å ha eksamen med penn og papir er gammeldags. Det er derfor gjennomført uttesting av en alternativ evalueringsform hvor studentene får bruke PC under eksamen. De første testene brukte en todelt eksamen hvor en del var vanlig penn og papir og en del var på PC hvor studentene kunne programmere i et halvveis realistisk miljø. Dette ble gjort for at den velkjente eksamensformen skulle være en sikkerhetsventil i tilfellet eksperimentet ikke virket etter hensikten. Det var en del praktiske problemer med todelt eksamen så de siste to årene ble en rein PC-eksamen brukt. Det var høy strykprosent på den ordinære eksamenen og noen praktiske problemer på kontinuasjonseksamen men disse problemene vil forhåpentligvis bli borte etter hvert som man samler erfaringer med eksamensformen.

1 Innledning

Faget "BID100 programmering, videregående" er det andre programmeringsfaget som datastudenter ved Universitetet i Stavanger har, det første er BID230 grunnleggende programmering. Begge disse fagene er på 10 studiepoeng. BID100 hadde tidligere vanlig skriftlig eksamen.

Læringsmålene i BID100 inkluderer at studentene skal kunne skrive objektorienterte programmer, skal kunne skrive grafiske brukergrensesnitt, skal kunne enkel bruk av tråder og skal kunne enkel programmering av applikasjoner som snakker sammen over et nettverk. Studentene har hatt noe objektorientert programmering og grafiske grensesnitt i grunnkurset, men erfaringsmessig mestrer de ikke dette etter grunnkurset.

For å skrive objektorienterte programmer trenger studentene ikke bare å kunne selve språket (Java i dette tilfellet), men må også kunne abstrahere (for å lage gode klassestrukturer), og må kunne problemløsing. På en klassisk eksamen med penn og papir kan man teste teorikunnskapene til studentene, samt at man kan teste kunnskapene deres om språket og språkstrukturer. Det er imidlertid vanskelig å få testet problemløsing, abstraksjonsevne eller evne til å lage gode klassestrukturer med en slik eksamensform.

Man skriver vanligvis programmer i et utviklingsmiljø som gir direkte tilbakemeldinger på skrivefeil og feil bruk av programmeringsspråket på samme måte som Word gir feilmeldinger på stavefeil og grammatiske feil. Dessuten har man som regel tilgang til oppslagsverk hvor navnene på alle kommandoene og funksjonene er oppgitt.

I følge [1] kapittel 5 er evalueringsformen viktig for hva studentene faktisk konsentrerer seg om. Siden det viktigste læringsmålet for dette faget er at studentene skal kunne objektorientert programmering, bør dette faget ha en evalueringsform som tester dette. Vanlig teorieksamen på penn og papir kan ikke teste dette. Et alternativ som

Denne artikkelen ble presentert på NIK2012. For mer informasjon, se //www.nik.no/

muligens ville vært bedre er å ha en eksamen på PC-sal hvor de kan programmere i et mer reelt miljø, hvor studentene kan bruke et utviklingsmiljø og har tilgang til oppslagsverk. Derfor har det blitt gjennomført tester på hvor gode slike eksamener er som evalueringsform og hvor gjennomførbare de er i praksis. Faget BID100 ble valgt i stedet for BID230 siden BID100 har færre studenter og det er bedre å gjøre en slik test med en mindre gruppe. BID230 har omtrent 2,5 ganger så mange studenter som BID100 siden det er flere ingeniørlinjer som har BID230 som obligatorisk fag.

Flere studier har blitt gjort i utlandet rundt eksamensform for programmeringsfag, men de har stort sett konsentrert seg om det første programmeringsfaget, mens videregående programmering er det andre programmeringsfaget og bygger på et tidligere fag, grunnleggende programmering. [2] er en studie fra flere land hvor de har brukt en 1,5 timers eksamen på lab for å teste basis programmeringsferdigheter. De fikk noe av den samme karakterfordelingen som ble resultatet av eksamensformen i denne artikkelen og fikk en høyere strykprosent. De beskriver også ulike kriterier for å vurdere en besvarelse.

I [3] beskrives en evalueringsform på lab som egner seg for mange studenter. De bruker bare 30 minutter pr. student slik at mange studenter kan ha eksamen på samme PC etter hverandre og hvor både faglærer og sensor er til stede under eksamen for å se hvordan studentene jobber. Dette er et av kriteriene som blir evaluert.

For å teste ferdigheter i objektorientert programmering er begge disse eksamensformene for korte. For å lage et objektorientert program trenger studentene tid. 4 timer, som er normal eksamenstid ved Universitetet i Stavanger, er egentlig også i korteste laget.

Ulike evalueringsstrategier har ulike styrker og svakheter og [4] og [5] beskriver noen slike. Et poeng er at mens gruppearbeid ofte er best for læring så får man ikke evaluert individuelle ferdigheter gjennom slikt arbeid og dårlige studenter kan ofte skjule seg bak arbeidet til flinke studenter i samme gruppe. Derfor har de brukt totalt fire ulike evalueringsformer i sitt fag i [4]: Øvinger hvor studentene kan hjelpe hverandre men leverer hver for seg, Practica på lab hvor studentene må jobbe hver for seg med oppmålt tid, Case Study hvor studentene kan jobbe og levere i større grupper og en slutteksamen. [5] bruker litt andre kategorier, men de er stort sett tilsvarende. Dessuten har [5] sett på hva studentene foretrekker og deres studenter foretrekker enten lab-eksamener med åpen bok eller individuelle øvinger med karakter framfor gruppeøvinger med karakter eller tradisjonelle skriftlige eksamener.

I BID100 programmering, videregående er det et prosjekt i flere deler som utgjør øvingsbiten, og hvor studentene kan jobbe to og to og også har lov til å spørre andre studenter om hjelp. Prosjektet teller ikke på karakteren. Grunnen til det er beskrevet i kapittel 2.2.

Universitetet i Agder har startet med å implementere eksamener på PC-sal i en rekke fag [6], mye på grunn av at studentene er vant til å bruke PC fra videregående skole og synes det er gammeldags å ha papireksamener på universitetet. De har testet ulike rammeverk men ikke funnet et de er helt fornøyd med.

For å beskrive eksperimentene og bakgrunnen for dem vil det først bli gitt en oversikt over de ulike alternativene som ble vurdert for evaluering av BID100. Deretter vil det komme to kapitler om to litt ulike eksamensformer som har blitt prøvd og erfaringene som er gjort med dem. Deretter kommer en kort konklusjon og oversikt over videre arbeid som er tenkt gjennomført.

2 Alternativer for evaluering av BID100

Tre alternativer ble opprinnelig vurdert som evalueringsform for BID100. De blir beskrevet i kapittel 3.1 til 3.3.

2.1 Vanlig eksamen med penn og papir

Før prosjektet som er beskrevet i denne artikkelen ble påbegynt, ble BID100 evaluert med en tradisjonell eksamen. Det vil si en firetimers eksamen hvor studentene skal skrive sin besvarelse med penn og papir og bare har lov til å ha med seg en enkel kalkulator. Studenter vil i følge [1] og [7] ofte konsentrere seg om det som er relevant til eksamen og prøve å gjøre så lite som mulig på øvingene underveis, så man må teste det som man ønsker at studentene skal lære. Å skrive programmer med penn og papir blir fort veldig omfattende, så oppgaver som involverer programmering må være veldig små. Det er dessuten et lite realistisk arbeidsmiljø. Når man programmerer har man normalt tilgang til en del dokumentasjon, et program som sjekker at man har programmert riktig, samt mulighet til å testkjøre programmet. Siden det å lære seg programmering er et viktig læringsmål for emnet, er en vanlig skriftlig eksamen en lite egnet evalueringsform for dette emnet.

2.2 Karakter på prosjektet

Faget BID100 har lenge hatt et programmeringsprosjekt i flere deler. Ideen er at studentene skal utvikle et enkelt spill og bygge på det med flere og flere egenskaper etter som de lærer seg hvordan å programmere dem.

Et alternativ for å evaluere programmeringsferdighetene er å la prosjektet telle med i karakteren. På den måten blir programmeringsferdighetene deres testet gjennom prosjektet og teorikunnskapene deres blir testet på eksamen. Problemene med denne løsningen er beskrevet i resten av kapittel 2.2:

Forfatterens erfaring fra sin studietid er at hvis studentene får karakter på prosjektet i ett fag men ikke de andre, så vil studentene jobbe mer med faget hvor de får karakter på prosjektet og mindre på de andre fagene, slik at de vil gjøre det dårligere i de andre fagene.

Med det nåværende godkjent/ikke godkjent systemet på prosjektet kan man etter noen år gjenbruke prosjektoppgaver. Det kan man ikke hvis det blir gitt karakterer siden det vil gi en uforholdsmessig stor fordel til de som kjenner noen som har hatt dette prosjektet tidligere eventuelt klarer å skaffe sen en kopi av en tidligere versjon på annet vis. Vanlige plagiatkontrollsystemer fungerer dårlig på programkode. Derfor blir det betydelig mer jobb for faglærer å både gi karakter på prosjektene og hele tida finne på nye prosjekter.

Selv for en nylagd prosjektoppgave kan det fins eksisterende løsninger på nettet som studentene kan kopiere og det vil være mye arbeid for faglærer eller studentassistent å sjekke dette.

Et annet poeng som taler mot å gi karakter på prosjektet er dette: I følge [1] kapittel 5 og 6 er det viktig å skille mellom formativ evaluering (som har som hovedformål å fortelle studenten hvor han/hun er i faget og bidra til læring) og summativ evaluering (som har som hovedformål å rangere studentene for framtidige arbeidsgivere eller for søking til et masterstudium). Eksamen er først og fremst summativ evaluering. Ved å gi karakter på prosjektet som teller på sluttresultatet vil også prosjektet bli en del av den summative evalueringa mens ved bare å gi godkjent/ikke godkjent kan tilbakemeldinger på prosjektet brukes mer som formativ evaluering.

2.3 Firetimers eksamen på PC-sal uten nett

Ideen bak dette forslaget er at det avholdes en vanlig firetimers eksamen som setter karakteren, men at studentene får et mer realistisk miljø å programmere i. Det vil si at studentene skal få sitte på hver sin datamaskin hvor de har tilgang til en del dokumentasjon samt det programutviklingsmiljøet som de er vant til å bruke fra prosjektet. Studentene får også ta med seg læreboka i faget. For å hindre at studentene samarbeider[1] så kan ikke datamaskinene ha tilgang til internett.

En fordel med denne eksamensformen sammenliknet med karakter på øvingene er at studentene ikke kan søke opp løsninger på internett (siden de ikke har tilgang til det under eksamen) eller skaffe løsninger av tidligere studenter. En annen fordel er at studentene ikke i samme grad vil ignorere de andre fagene i løpet av semesteret for å få en god karakter på programmeringsfaget.

Fire timer er ganske kort tid til å skrive et program selv med de normalt verktøyene tilgjengelig. Derfor må programmeringsoppgavene fortsatt være ganske små, men likevel betydelig større enn de kan være med penn-og-papir eksamen. En fordel med dette er at de blir lettere å sette karakter enn på et større prosjekt. En ulempe er at det begrenser hva slags oppgaver man kan gi samt at studenter som er trege til å programmere ikke blir ferdige selv om de hadde klart oppgaven med mer tid.

Selv om den har ulempene med at oppgavene må være små og noen studenter ikke blir ferdige selv om de hadde klart oppgaven med mer tid så har denne eksamensformen flere fordeler sammenliknet med både karakter på prosjektet og vanlig teorieksamen. Derfor ble denne formen valgt framfor de to andre.

3 Første test: Separate teori- og programmeringseksamener

I starten var det stor usikkerhet om og hvordan eksamen på PC-sal ville fungere. Derfor fikk faget BID100 i en overgangsperiode to eksamener: En teorieksamen med penn og papir og en eksamen på PC-sal. På denne måten ble det skapt en viss sikkerhet. Hvis opplegget med eksamen på PC-sal ikke fungerte kunne man falle tilbake på teorieksamenen. De to eksamenene var som følger:

Teorieksamenen: Dette er en klassisk eksamen med penn og papir uten hjelpemidler. Denne ble brukt siden det er det er den forrige eksamensformen og både forfatteren og arbeidskollegaer har erfaring med denne eksamensformen. Denne eksamenen var ment å teste de teoretiske kunnskapene til studentene.[2]

Eksamen på PC-sal (programmeringseksamen): På denne eksamenen sitter studentene og skriver på PC og har tilgang til læreboka, et programutviklingsverktøy, dokumentasjon om Java samt programmer som er standard på PC-er i dag slik som Office-pakken. Studentene har ikke tilgang til internett. Denne eksamenen var først og fremst beregnet på å teste de praktiske programmeringsferdighetene og problemløsningsferdighetene til studentene.

3.1 To separate eksamener som evalueringsform

Todelte eksamener ble brukt i to år, og erfaringen var at eksamen på PC-sal (programmeringseksamen) i hovedsak har fungert etter planen. Erfaringene med programmeringseksamen som evalueringsform er samlet for alle fire årene de er gjennomført i kapittel 4.

[1] Eksamen skal være en test på individuelle ferdigheter. Studentene har mulighet til å jobbe i toergrupper på prosjektet.

[2] Teorien i BID100 er slikt som hvordan lage et program som gjør to ting tilsynelatende samtidig og hvordan lage et program som reagerer på brukeren i stedet for å gå sekvensielt gjennom instruksjonene.

3.2 Praktiske erfaringer med å ha to eksamener

Det å ha to eksamener skaper en del ekstra arbeid for både faglærer og eksamenskontoret. Faglærer må rette to sett eksamener. Eksamenskontoret måtte skaffe lokaler og vakter til begge eksamenene. Faglærer og eksamenskontoret må også avgjøre hva som skjer hvis en student stryker på en eksamen og ikke på den andre, eller hvis noen er sjuk under den ene eksamenen men ikke den andre. Hva om noen vil forbedre karakteren på den ene men er fornøyd med den andre? Å stryke en student på hele opplegget hvis vedkommende strøk på en av de to eksamenene virket unødvendig strengt samt at dette ville økt strykprosenten ytterligere (strykprosenten er allerede på 30-35%). Dessuten hadde ikke teorieksamenen fungert så godt som sikkerhetsventil hvis en stryk på programmeringseksamenen betød stryk på hele faget. Løsningen som ble valgt er at studentene får en total poengsum på begge eksamenene og stryker hvis den totale poengsummen blir for lav. Etter en del diskusjon fikk studenter lov til å konte på enkeltdeler av eksamenen, men dette førte til mer arbeid for faglærer og eksamenskontor for å kombinere poengsummer fra eksamener tatt på ulike tidspunkt.

3.3 Konklusjon: Kjør programmeringseksamen aleine

Etter to års erfaring med programmeringseksamen samt erfaringene med at det er en del merarbeid med å ha todelt eksamen, ble det bestemt å kjøre bare programmeringseksamen og gjøre et forsøk på å bake noe teori inn i denne eksamenen. Dette vil forhåpentligvis gi mindre arbeid for faglærer, studentene og eksamenskontoret.

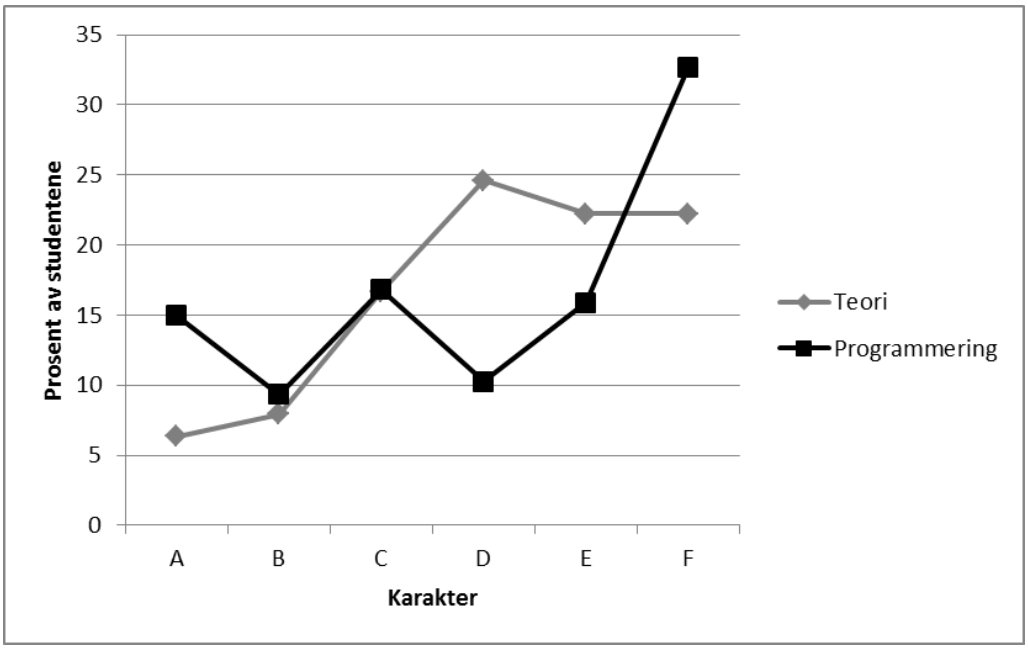
4 Programmeringseksamen som evalueringsform

Programmene studentene skriver som besvarelser kan lastes inn i et utviklingsmiljø hvor man kan kjøre og se gjennom dem. Å sette karakter på programmene tar derfor noe kortere tid enn for en tradisjonell eksamen.

Eksamener på det teknisk-naturvitenskapelige fakultetet bruker som regel en form for kriteriebasert evaluering ([1] kapittel 5) hvor det er en veiledende skala som sier hvilken karakter studenten skal få basert på hvor mange prosent av oppgavene vedkommende har klart. Programmeringseksamenen har også brukt denne formen for evaluering.

Karakterene fra disse programmeringseksamenene har fulgt ulike fordelinger, men ingen har vært i nærheten av den vanlige normalfordelingen med gjennomsnitt på C. En sammenlikning mellom karakterfordelingen på programmeringseksamenene og de tradisjonelle eksamenene er gitt i figur 1. Figuren er basert på summen av karakterfordelingene fra mange eksamener. Teori-kurven er basert på de tradisjonelle eksamenene fra 2005, 2006 og 2007 samt teoridelene til de todelte eksamenene i 2008 og 2009. Programmering-kurven er basert på programmeringsdelen av de todelte eksamenene i 2008 og 2009 samt de reine programmeringseksamenene i 2010 og 2011. Figuren er bare basert på ordinære eksamener og utelater kontinuasjonseksamener siden de domineres av svake studenter og derfor har en fordeling med mange på de dårlige karakterene for begge eksamensformene.

Programmeringseksamenen høsten 2009, hvor studentene ikke trengte å lage et grafisk brukergrensesnitt, hadde den mest ekstreme karakterfordelingen med mange A-er og mange E-er og få mellom. En mulig forklaring er at denne eksamenen stort sett testet basis programmeringsferdigheter, noe som de som fikk A har skjønt ordentlig mens de som fikk E kunne en del teknikker men ikke skjønte systemet.



Figur 1: Karakterfordeling på teori og programmeringseksamener

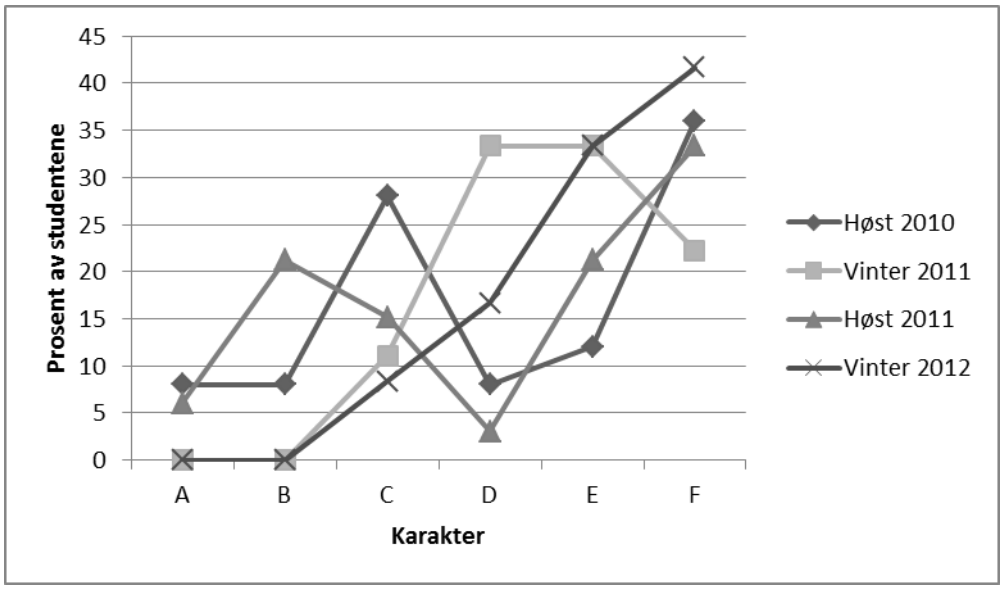
Det var en høy strykprosent på den første reine programmeringseksamenen, ordinær eksamen høst 2010. Denne eksamenen hadde en enkelt programmeringsoppgave som telte 65% av karakteren, to teorioppgaver som telte 10% hver og en oppgave som handlet om å finne og fikse feil i et eksisterende program som telte 15%. Programmeringsoppgaven på 65% var fra faglærer sin side først og fremst ment å være en test på objektorientert programmering (en sentral del av faget), men enkle figurer i Java 2D ble brukt for å teste dette. Faget inneholdt nok Java 2D til at studentene skulle kunne gjennomføre oppgaven, men det var en liten del av pensum som få hadde lest skikkelig på. Tilbakemeldninger i etterkant[3] tyder på at studentene så seg blinde på Java 2D biten og ikke så hvor liten den egentlig var.

Strykprosenten var mye lavere på kontinuasjonseksamenen, vinter 2011. På denne eksamenen telte den største oppgaven 40% av eksamenen. Den store oppgaven var også laget slik at den bare testet kjernepensum uten å bruke mer perifere deler av pensum. De perifere delene av pensum var i mindre oppgaver, slik som en 10% oppgave om UML sekvensdiagram eller en 15% oppgave rundt Java teknologier. 40% oppgaven hadde ingen referanser til slike tema.

Bortsett fra den høye strykprosenten har karakterfordelingen vært mer ”riktig” på den ordinære eksamenen høst 2010 enn på de tidligere programmeringseksamenene. Dette kan skyldes at den var bedre til å skille de virkelig flinke fra de ikke fullt så flinke. Den ordinære eksamenen hadde en topp på C og en topp på F, og brukte hele skalaen (Se Figur 2, linja Høst 2010). Kontinuasjonseksamener har vanligvis ikke normal fordeling av karakterene siden de flinke studentene som regel har bestått på den ordinære eksamenen så karakterfordelingen på den (Figur 2, linja Vinter 2011) sier ikke så mye som karakterfordelingen på den ordinære eksamenen.

En konklusjon som kan trekkes fra dette er at ingen av oppgavene bør være for store. En stor oppgave blir fort sårbar ovenfor det at en student ikke har fått med seg den

[3] Tilbakemeldingene er fra et elektronisk studentevalueringsskjema for emnet som faglærer hadde lagt ut på intranettet og hvor faglærer bevisst hadde satt fristen til å være noen dager etter eksamenen, samt samtaler med noen enkeltstudenter.



Figur 2: Karakterfordeling på programmeringseksamenene

bestemte delen av pensum som oppgaven refererer til. Det er bedre med flere mindre oppgaver. Dette vil muligens gi mer arbeid til faglærer siden faglærer muligens må skrive en del kode som deles ut på eksamenen. Dette for at oppgavene skal bli små nok til at studentene kan greie å gjøre alle på den tilmålte tida.

Ordinær eksamen høst 2011 hadde samme struktur som konten, men oppgave 1 (40% oppgaven innen objektorientert programmering) hadde en del utdelt kode for å gjøre den enkel nok. Av 42 oppmeldte så var den en som ikke møtte, 8 som trakk seg inder eksamen og en karakterfordeling vist i Figur 2, linja Høst 2011. Strykprosenten(33%) var lavere enn for ordinær eksamen i fjor men fortsatt høyere enn i mange andre fag.

Den nåværende tenkningen rundt oppgavedesign er som følgende:

- Basis objektorientert programmering: 40%
- 3 oppgaver fra ulike deler av teoripensum som godt kan involvere noe programmering: 15% hver
- 1 oppgave som går ut på å finne feil i en eksisterende kode: 15%

Denne oppbygningen er valgt fordi:

- Den første oppgaven skal teste basis programmeringsferdighetene til studentene, og bør derfor ikke ha så mye utdelt kode. Samtidig har det vist seg vanskelig å lage en oppgave som både tester objektorientert programmering og som kan løses på 1,5 timer av en gjennomsnittsstudent. Mye av dette er egentlig pensum i grunnleggende programmering, men det har vist seg at mange studenter fortsatt sliter med det.
- De tre neste oppgavene skal teste at de har forstått teorien i faget. En enkelt eksamen kan ikke teste alle delene av pensum så et utvalg (ulikt for hvert år) blir gjort. Her skal de skrive småprogrammer som ikke krever så mye tid så fremt de har forstått teorien. Teorien i dette faget er teorien bak ulike programmeringskonstruksjoner slik som teorien bak objektorientert programmering, noe teori om bruk av tråder og noe teori om nettverk.
- Den siste oppgaven kan teste enten basis programmeringsferdigheter eller mer avansert kunnskap. Den er inkludert fordi dette er noe studentene kommer til å måtte gjøre mye framover siden en viktig del av programvareutviklingen er å finne og fikse feil i programmer.

Ut fra resultatene virker det som om de mest teoretiske oppgavene var de med lavest poengsnitt blant studentene selv om teorien de trengte sto i læreboka, og de hadde lov til å ta med læreboka på eksamen.

Generelt sett er dette en oppgavetype hvor det er vanskeligere å lage oppgaver, men lettere å rette dem. De samme verktøyene som studentene får bruke under eksamen kan brukes etterpå til å sjekke koden deres og det er i mange tilfeller lett å se hva de har fått til. De eneste besvarelsene som kan være vriene å rette er de som har lite kjørbar kode men har skrevet en del uten å ha fått det til så godt at det kan kjøres.

5 Studentenes tilbakemeldinger

Studenter er ofte skeptiske til nye eksamensformer, men de tilbakemeldingene studentene har gitt tyder på at de er i all hovedsak positive til den nye eksamensformen. De negative tilbakemeldingene som har kommet har vært av to typer. Noen har handlet om enkeltoppgaver slik som 65%-oppgaven på eksamenssettet høst 2010. Resten har handlet om at studentene synes de får for lite tid. En merkbar forskjell mellom programmeringseksamener og tradisjonelle eksamener er at studentene bruker mye mer tid på programmeringseksamenene. På programmeringseksamenene sitter de fleste studentene ut tida, mens på tradisjonelle eksamener har de fleste allerede levert etter tre timer. Med de nåværende eksamensreglene ved UiS er det dessverre ikke mulig å gi eksamener som varer lengre enn fire timer for et fag på ti studiepoeng.

6 Erfaringer med den praktiske gjennomføringen av programmeringseksamen

Det er mer jobb for oss på instituttet under gjennomføringen. Dessuten må IT-drift være med på laget og det kan bli en del ekstra jobb på dem også.

IT-drift må sette opp en PC-sal med spesialbrukere. Disse spesialbrukerne har tilgang på en harddisk som ligger på nettet og ikke lokalt (F:), men de har ikke tilgang til internett. Datamaskinene må ha en slik nettverksharddisk slik at studenten ikke mister det han/hun har gjort selv om datamaskinen studenten sitter ved svikter. Det har enda ikke vært problemer med at PC-ene svikter under eksamen, men det er egentlig bare et spørsmål om tid før det skjer. I et slikt tilfelle vil studenten kunne flytte over til en annen datamaskin og fortsette arbeidet sitt med bare et lite avbrekk.

IT-drift har laget et system for å opprette spesialbrukerne fra forrige avsnitt. Dette må gjøres bare en gang for alle fag som skal ha denne formen for eksamen. Det må imidlertid ofte gjøres på nytt hvis det blir installert nye versjoner av Windows eller andre programmer som er essensielle for nettverket på universitetet. Dette blir derfor en ekstra byrde for IT-drift hvis de skal oppgradere operativsystemet eller programvaren på universitetet.

Dessuten må slike systemer testes grundig. En av de første gangene programmeringseksamen ble kjørt på V102 så viste det seg at noen maskiner fortsatt fikk tilgang til internett. Hver enkelt maskin måtte derfor sjekkes og man måtte kjøre et program på de som fortsatt hadde tilgang til nettet slik at internett var blokkert slik som det skulle. Disse testene tok 1,5 timer for forfatteren og en IT-ingeniør for en eksamen med 40 studenter. Seinere år har dette problemet oppstått bare en gang og da på et annet rom, så man må ta stikkprøver på den eller de PC-salene som brukes for å sjekke at oppsettet virker etter hensikten.

De første gangene denne eksamensformen ble gjennomført var det en IT-kyndig til stede på PC-salen(e) hele tida for å håndtere datatekniske problemer. Det kunne være enten faglærer eller en IT-ingeniør. Dette kom i tillegg til de vanlige eksamensvaktene.

Det viste seg etter hvert at det var såpass få problemer at det ikke var nødvendig å være til stede hele tida. Man bør imidlertid ha en IT-kyndig for hver PC-sal som brukes som kan kontaktes av eksamensvaktene hvis det oppstår problemer.

Innlevering må skje elektronisk[4], så på slutten av eksamenen må flere datakyndige være til stede for å kopiere filene inn på minnepinner. Minnepinnene brukes som sikkerhetskopi i tilfelle noe går galt. IT-drift vil samle sammen alle F-katalogene i en stor .zip fil og sende denne til faglærer. Foreløpig har faglærer selv vært med på å samle inn besvarelser men ideelt sett skal ikke faglærer ha noe med den prosessen å gjøre siden faglærer kan se hvem det er som har de kandidatnumrene som faglærer personlig tar imot.

De ordinære eksamenene i 2010 og 2011 samt kontinuasjonseksamenen vinter 2012 ble gjennomført på UiS sin største PC-sal med 70 PC-er (rom V-102). Dette rommet er stort nok til å ha alle studentene på ett sted selv om det ble både trangt og dårlig luft under eksamen høst 2011, da det var 42 studenter der. IT-drift hadde lagd et opplegg for eksamenen som fungerte knirkefritt. Problemet er at V-102 er et svært mye brukt rom så faglærer må få vite eksamensdatoene svært tidlig for å kunne bruke det. Ellers risikerer man at det allerede er bestilt av noen andre.

For kontinuasjonseksamenen vinter 2011 var det ikke mulig å bruke dette rommet siden det var reservert til andre undervisningsaktiviteter før faglærer fikk vite eksamensdatoen. To mindre PC-saler måtte derfor brukes i stedet. Dette gikk utover de øvingsaktivitetene som normalt skulle foregått der. Det kan generelt være et problem å skaffe rom til kontinuasjonseksamener siden de går midt i semesteret på UiS.

På det ene av de to rommene[5] som ble brukt for kontinuasjonseksamenen vinter 2011 fungerte ikke opplegget til IT-drift helt, slik at man fortsatt kunne bruke internett på dem. Dette ble ikke oppdaget før klokka 14:30 dagen før. Derfor ble det en lettere panikkartet testing av nødløsninger den dagen og før eksamenen dagen etter. Nødløsningen som ble valgt[6] fungerte men er lite optimal siden den førte til at enkelte operasjoner på PC-ene ble svært trege samt at studentene måtte lagre på den lokale harddisken (C:). Hvis en av de maskinene hadde sviktet under gjennomføring hadde vi risikert at studenten hadde mistet alt arbeidet sitt!

Under gjennomføring av kontinuasjonseksamen vinter 2012 var det to studenter som hadde fått innvilget krav på å sitte i eget rom med bærbar PC. Dette første til en del ekstra arbeid for både IT-drift (for å sette opp PC-ene) og faglærer (for å teste at de virker etter hensikten). Desto flere slike spesialkrav det blir fra studentene desto mer arbeid blir det å gjennomføre denne typen eksamen.

7 Konklusjoner

Å vurdere programmeringseksamener er noe mindre arbeidskrevende enn å vurdere en vanlig teorieksamen, men det er mer arbeid å lage oppgavene. Generelt er arbeidsbelastningen for faglærer i å lage oppgave og vurdere resultatene til sammen omtrent lik som for en vanlig teorieksamen.

Det har foreløpig vært betydelig mer ressurskrevende å gjennomføre programmeringseksamener. Faglærer og IT-drift har nå fått lagd et standardisert system

[4] Oppgavene rettes ved å laste dem inn i samme utviklingsmiljø som studentene bruker til å skrive dem. Derfor er å skrive ut koden på papir uaktuelt.

[5] Dette rommet (KE E455) hadde et spesialoppsett for å kunne bruke en del utstyr som var unikt til dette rommet. Derfor hadde ikke IT-drift oppdatert Novell til siste versjon. Den eldre versjonen av Novell forstod ikke hele oppsettet som IT-drift hadde lagd for eksamensbrukerne.

[6] Nødløsningen var å la PC-ene starte, logge inn med eksamensbrukere, starte de programmene som skulle startes, og deretter trekke ut nettverkskablene.

som gjør det mindre arbeidskrevende nå enn i starten. Selv om vi har gjennomført dette opplegget i fire år nå, er det fortsatt mer arbeid med eksamensgjennomføring enn for en ordinær eksamen. IT-drift må teste eksamensbrukerne på PC-salen og eksamensbrukerne må ofte settes opp på nytt når man for eksempel oppdaterer til en ny versjon av Windows. En IT-kyndig må være tilgjengelig på telefon til en hver tid, men systemet er såpass stabilt nå at vedkommende trenger ikke lengre være til stede på PC-salen. Den IT-kyndige kan være faglærer eller en labingeniør.

Det er også begrensninger på hvor store fag man kan gjennomføre denne eksamenstypen på. UiS sin største PC-sal, V-102, har 70 arbeidsstasjoner, men de står mye tettere enn de bør gjøre under en eksamen. På en vanlig PC-sal så står maskinene såpass tett at studenten lett kan se hva naboen holder på med. For å bruke PC-saler på eksamen bør studentene sitte mer spredt (tilsvarende spredt som på en tradisjonell eksamen) samt at man bør ha skillevegger som gjør det vanskeligere å se på andre sine skjermer. Dette fører til at i praksis kan man bruke bare halvparten av PC-ene på en vanlig PC-sal. Det å bare bruke halvparten av Pc-ene gir også en ekstra sikkerhet slik at hvis en PC slutter å virke så er det mange reserve-PC-er.

Det høyeste antall studenter som har møtt på en enkelt eksamen er 42. Det gikk bra på V-102, men bare så vidt. 42 er derfor det høyeste antall studenter vi kan ha i dette rommet under en programmeringseksamen. For større fag må man bruke flere datarom og man må sette opp alle disse PC-salene med eksamensbrukere samt kontrollere at de virker og sjansen for at noe går galt vokser for hver ny PC-sal man tar i bruk. Alternativet er å kjøre eksamenen på flere ulike tidspunkt.

En annen erfaring er at PC-salene må reserveres lang tid i forveien, noe som gjør at eksamensdatoen må fastsettes før endelige datoer er tilgjengelige.

Eksamenen bør ha flere mindre oppgaver i stedet for en stor oppgave. En stor oppgave har riktignok den fordelen at den bedre tester studentenes evne til å løse mer kompliserte, realistiske problemer. Ulempen at studenter som ikke har fått med seg en av delene av pensum som oppgaven tester stryker er imidlertid verre.

Studentene har i hovedsak vært fornøyde med den nye evalueringsformen selv om studenter vanligvis er skeptiske til alternative evalueringsformer. Den eneste kritikken som har kommet er at studentene synes det er for kort tid med fire timer til en slik eksamenstype og at studenter som er litt trege til å programmere får for kort tid og kunne ha fått til oppgaven med mer tid.

Eksamensformen er godt gjennomførbar for et fag på størrelsen til BID100, med 25-40 studenter. Begrensningen på antall studenter kommer av hvor store PC-saler som er tilgjengelige. Å bruke denne eksamensformen på et fag med flere studenter vil bare være mulig hvis man kan reservere nok PC-saler og man er trygg på den praktiske gjennomføringen. Det siste er fordi man nå trenger en datakyndig på hver PC-sal, så for å bruke eksamensformen på et fag med mange studenter må man involvere mange dataingeniører som eksamensvakter. Det vil derfor være noen år enda før man kan ta i bruk denne eksamensformen på det grunnleggende programmeringsfaget som har rundt tre ganger så mange studenter som BID100.

8 Planer videre

Faget BID100 vil bli gjennomført som nå i bare ett år til. De nye studieplanene vil ha et annet fagoppsett så faget går ut. Vi på datagruppa på UiS diskuterer om vi skal bruke denne eksamensformen i noen av de nye fagene men har ikke kommet til en konklusjon enda.

9 Referanser

1. Strømsø, Lycke, Lauvås (reds.): Når læring er det viktigste: Undervisning i høyere utdanning. Cappelen akademiske forlag, 2006
2. Michael McCracken et al (2001): A multi-national, multi-institutional study of assessment of programming skills of first-year CS students. SIGCSE Bulletin 33(4), pages 125-180.
3. Bennedsen, J. og Caspersen, M. E. (2007): Assessing Process and Product – A Practical Lab Exam for an Introductory Programming Course. ITALICS vol. 6, issue 4, october 2007.
4. Chamillard, A. T. and Braun, K. A. (2000): Evaluating Programming ability in an Introductory Computer Science Course. Proceedings of SIGCSE 2000, pages 212-216
5. Barros, J. P et al: Using Lab Exams to ensure Programming Practice in an Introductory Programming Course. I ACM SIGCSE Bulletin - Proceedings of the 8th annual conference on Innovation and technology in computer science education Homepage Volume 35 Issue 3, September 2003, Pages 16-20
6. Universitetet i Agder: Anbefalinger til UiA sitt arbeid med digital eksamen. Prosjektrapport fra prosjektet "digital eksamen", universitetet i agder, høst 2011. URL: http://www.uia.no/no/portaler/student/eksamen/prosjekt_digital_eksamen
7. Pedagogikkurs for nyansatte ved UiS 2006-2007 gitt av professor Kirsten Hofgaard Lycke og professor Gunnar Handal fra Universitetet i Oslo.
8. Per Lauvås og Arne Jakobsen: Exit eksamen, eller? Former for summativ evaluering i høgre utdanning. Cappelen Akademisk Forlag, 2002