Ska teknologen tillverka tentan?
– Erfarenheter av studentmedverkan i utformningen av skriftlig examination

Björn Regnell
Software Engineering Research Group, Telecom/LTH

***Abstract*—De teknologer som går kursen ETS170 Kravhantering har i grupp, veckovis lämnat in förslag på tentamensproblem med tillhörande bedömningsmall. Om dessa är av tillräckligt hög kvalitet enligt givna riktlinjer renderar de bonuspoäng som tillgodoräknas tentamen och problemen har också en reell chans att bli inkluderade i själva tentamen. I denna artikel beskrivs det praktiska upplägget och erfarenheterna från två tillämpningar av studentmedverkan i tentamensutformningen. Erfarenheterna visar att nästan alla studenter är villiga att försöka uppnå bonus. Genom riktlinjerna för vad som krävs för att få bonus ges en möjlighet att styra mot djupinlärning och samtidigt ge tydliga mål för vad som krävs vid examinationen. Studenternas arbetsbelastning blir jämnare fördelad över tiden och de får en kontinuerlig återkoppling på var de står i lärandeprocessen. En diskussion ges kring resursbehovet för detta kursmoment. Slutsatsen är att arbetsinsatsen från examinator vid bedömningen av studenternas tentamensproblemförslag är ytterst väl investerad tid med tanke på lärandeutfallet.**

***Nyckelord* — Examination, motivation, skriftlig tentamen.**

I. Introduktion

En stor utmaning i undervisningen vid tekniska högskolor är att förmå studenterna att studera kontinuerligt under kursens gång snarare än att vänta till omedelbart före en skriftlig tentamen som ofta avslutar en kurs. Olika insatser görs för att motivera studenterna att kontinuerligt arbeta med sitt lärande under hela läsperioden, t.ex genom duggor och återkommande inlämningsuppgifter. I kursen ETS170 Karvhantering har under två år tillämpat en speciell form av kontinuerlig examination där inlämningsuppgifter i form av tänkta tentamensproblem med tillhörande bedömningsmall har efterfrågats. Belöningen har varit bonuspoäng som kan tillgodoräknas resultatet på den skriftliga tentamen samt möjligheten att det inlämnade problemet, om det är av hög kvalitet, faktiskt inkluderas i den skriftliga tentamen.

II. Genomförande

Redan på första föreläsning introduceras syftet och utformningen av inlämningsuppgifterna. Riktlinjerna förmedlas i föreläsningsanteckningar och på kursens hemsida [2]. Följande riktlinjer definierade vad som är ett bra tentamensproblem:

- Examinerar helt eller delvis ett eller flera av kursens inlärningsmål
- Kräver djupare förståelse
- Av utredande eller värderande karaktär
- Kopplar ihop olika delar av teorin
- Rimligt att lösa på max 30 min för en som borde godkännas utifrån kursens inlärningsmål

Ett bra tentamensproblem ger 2p bonus på den skriftliga tentamen och med 6 inlämningsmöjligheter ger det max 12 bonuspoäng som adderas till tentamensresultatet som kan vara max 100p (bonusen kan alltså utgöra max 12% av maxpoängen).

Följande ytterliggare riktlinjer ges:

- Det ska finnas en bedömningsmall för hur tentamensproblemet ska poängsättas.
- Bedömningsmallen ska även innehålla exempellösningar på alla delfrågor.
- Lösningen skall innehålla referenser till relevanta delar av kurslitteraturen.
- Ett tentamensproblem ska ge max 10p.
- Det ska framgå av bedömningsmallen exakt hur examinator ska dela ut dessa 10p.
- Tentamensproblemet ska vara på svenska (med eventuella specialtermer även på engelska inom parentes vid behov).
- Tentamensproblemet ska lämnas i via e-post i html.
- Nya varianter premieras! Det är ej tillåtet att plagiera rakt av, tex ur listan med studenternas tidigare problem som läggs ut på kurswebben eller från extentor.
- Problemet skall lämnas in gruppvis senast 1 vecka efter varje övningstillfälle (sen inlämning ger ej bonus).
- Det är tillåtet att dela upp arbetet mellan gruppmedlemmarna.

Både under 2004 och 2005 så utnyttjade i stort sett alla grupper denna möjlighet att få tentamensbonus och endast enstaka grupper missade ett eller två bonustillfällen genom sen inlämning.

Efter varje gruppvis inlämning så gör kursansvarig en bedömning av de inlämnade tentamensproblemen och avgör om de får 0, 1 eller 2 poäng i bonus för varje individ. Kursansvarig skriver också en kort motivering till beslutet som läggs ut på kurswebben tillsammans med tentamensproblemet. På så sätt får alla studenter tillgång till en omfattande katalog av tentamensproblem med varierande

kvalitet men med kursansvarigs förklaring till varför de är bra eller mindre bra.

Bedömning av ett tentamensproblem med tillhörande skriftlig motivering och webbpublicering tar ca 15 minuter per inlämnat tentamensproblem. År 2005 var det 14 grupper (3-4 studenter per grupp) som lämnade in i stort sett varje vecka i 6 veckor vilket krävde en arbetsinsats av kursansvarig på ca 21 h totalt för bedömning i kursen. Lägg därtill tid för kommunikation av riktlinjerna och enskilda samtal med studenter kring tentamensproblemen och en total tidsåtgång för detta kursmoment uppskattas till 30 h per kurstillfälle.

Ungefär hälften av problemen på den slutliga tentamen är baserade på studenternas tentamensproblem. Dock tas de ej rakt av, utan förbättras och omformuleras i den mån det krävs för att vara lämpliga på tentamen. De bestäa tentaproblemen väljs ut som dessutom passar in i tentamen vad avser täckning av områden. Inget studenttillverkat tentamensproblem har tagits med helt utan modifieringar.

III. Exempel

De flesta inlämnade problemen fick bonus. Exempel på ett inlämnat tentamensproblem som ej gav bonus (endast redigerat på rubriknivå):

Fråga: Det finns många stilar man kan använda sig av för att specificera krav. Två vanligt förekommande är feature requirements och computer-centric use cases. Samtidigt som dessa stilar båda beskriver en produkts egenskaper, skiljer de sig på ett flertal sätt. Beskriv skillnader mellan de ovan nämnda stilarna med avseende på följande egenskaper: (a) Förståelse hos kunden (3p), (b) Förståelse hos utvecklaren (3p)

(c) Varför lämpar sig inte computer-centric use cases som kravspecificering vid utveckling av COTS[1]? (2p)

(d) Vilka är nackdelarna med feature requirements i ett COTS-projekt, var har man mest nytta av dem? (2p)

Svar (a): Feature requirements är enklare för kunden att utforma och själv förstå eftersom man kan använda ett vardagligt språk när de skrivs. Computer-centric use cases använder däremot ett språk som är centrerat kring datatermer och hur programmet arbetar. Innehållet i ett computer-centric use case lämnar användaren utanför och beskriver endast vad systemet utför. För kundens intresse ligger mer vilka uppgifter som användaren kommer att utföra.

Bedömningsmall: 2 poäng ges om man endast har skrivit ner för- och nackdelar för de båda stilarna i förhållande till användaren. (1 Poäng för Computer centric use case och 1 poäng för feature requirements.) 1 poäng ges om man även har jämfört egenskaperna mot varandra, totalt max 3 poäng på uppgiften.

Svar (b): Feature requirements kan lämpa sig bra för utvecklaren, men det kan lätt bli så att kunden specificerar för många krav, eller krav som strider mot varandra. Computer-centric use case blir enklare för utvecklaren att förstå vad han ska göra eftersom uppgiften lyfts fram och blir tydligare än i ett feature requirement. Möjligheten att enkelt ta fram ett UML-diagram direkt från kraven är större i ett computer-centric use case. Computer-centric use cases kan vara för detaljerade, det vill säga att de kan vara nere på designnivå när de utformas. Detta begränsar utvecklaren på vilket sätt han kan designa lösningen av problemen, vilket gör att han inte kan hitta alternativa lösningar. Feature requirements ger utvecklaren mycket större frihet att såväl tolka som att designa lösningen.

Bedömningsmall: 2 poäng ges om man endast har skrivit ner för- och nackdelar för de båda stilarna i förhållande till användaren. (1 Poäng för Computer centric use case och 1 poäng för feature requirements.) 1 poäng ges om man även har jämfört egenskaperna mot varandra, totalt max 3 poäng på uppgiften.

Svar (c): Kraven från computer-centric use cases är för detaljerade för att fungera på domännivå, vilket gör att de inte riktigt är anpassade för COTS. Abstraktionen ligger på designnivå vilket gör att computer-centric use cases lämpar sig bättre för att skapa aktivitetsdiagram och hjälpa utvecklaren att implementera gränssnitt.

Bedömningsmall: 1 poäng om studenten tar med lämplig orsak till varför COTS inte är ett lämpligt alternativ. 1 poäng om studenten anger vad man bör använda computer-centric use cases till.

Svar (d): Den som skriver kraven till en COTS-produkt har svårt att se till hela marknadens behov. Detta gör att han blir influerad av redan existerande verktyg, samarbete med andra organisationer samt affärsmässiga förtjänster. Produkten blir med andra ord inte utvecklad för att främja kundernas intressen utan de egna, eller konkurrenternas.

Bedömningsmall: 1 poäng om studenten tar upp lämpliga nackdelar med feature requirements i COTS-projekt. 2 poäng om studenten har en mer utförlig beskrivning om nackdelarna.

Följande (oredigerade) motivering gavs av kursansvarig för varför det ej blev någon bonus:

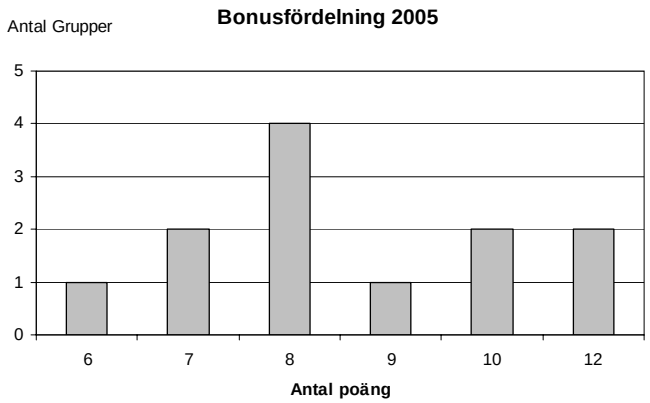
"Frågor som försöker jämföra två tekniker tycker jag i grunden är jättebra. Det kräver att man tänker efter på djupet och att man kan värdera teknikerna i ett sammanhang, som i denna fråga där teknikerna sätts i relation till en viss projekttyp. Tyvärr är denna fråga omöjlig att ha på tentan då den är tvetydig - "COTS-projekt" är ett luddigt begrepp och det är inte klart vad som avses: offertförfrågan, upphandling eller utveckling av generisk hyllprogramvara? Ännu värre är att det är direkt fel svar i bedömningsmallen. Vid utveckling av COTS-produkter kan man mycket väl ha krav på produkt eller designnivå och i många utvecklingsprojekt där man tar fram produkter för en öppen marknad så sker validering inte med slutanvändare på samma sätt som när ett kontrakt eller offertunderlag finns. Dock är det vid inköp eller upphandling av COTS ofta meningslöst att ha krav på lägre nivå än domännivå eftersom det ofta handlar om att jämföra befintliga produkter som ju är som de är om man inte har möjlighet att köpa speciell kundanpassning. Om poängen med frågan är att kolla förståelse av vikten av att ligga på domännivå i vissa sammanhang, så är frågan inte väl vald eftersom feature requirements ofta hamnar på produktnivå snarare än domännivå, medan use cases eller task description rätt använda passar utmärkt på domännivå. 0p"

IV. Resultat

Både år 2004 och 2005 blev detta kursmoment mycket lyckat ur både kursansvarigs och studenternas synvinkel. Studenterna tog tentaprobleminlämningarna på mycket stort alvar och 2004 klarade sig alla som deltog i tentamen, trots att kursansvarig anser att tentan är omfattande och krävande. År 2005 var det endast 3 av 40 som deltog i tentamen som blev underkända.

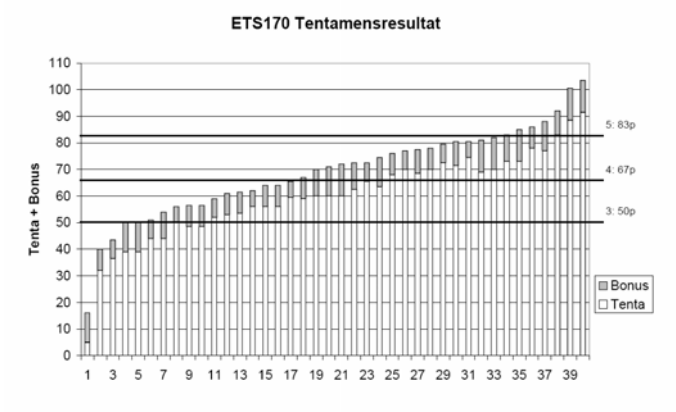
Figur 1 visar bonusutfallet för 2005 för de 14 grupperna som lämnade i max 6 inlämningar, en gång i veckan. Diagrammet visar att det var 2 grupper som fick maximal bonus och den gruppen som fick lägst erhöll 6 poäng i bonus.

[1] COTS betyder Commercial Off-The-Shelf Software och avser generiska programvaruprodukter som säljs till många slutkunder på en open marknad.



Figur 1. Bonusfördelning för de 14 grupperna 2005.

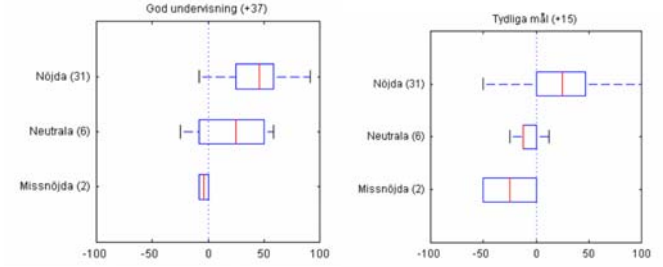
Figur 2 visar fördelningen av tentamenspoängen plus bonus. Det är tydligt att bonusen har spelat roll både för godkänmnande (4 studenter) och för att uppnå ett högre betyg (11 studenter).



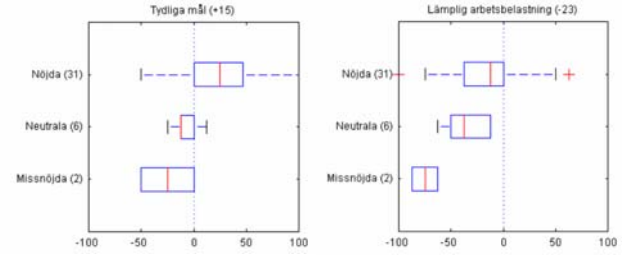
Figur 2. Tentamensresultat och bonus för 40 tenterade 2005.

V. Erfarenheter

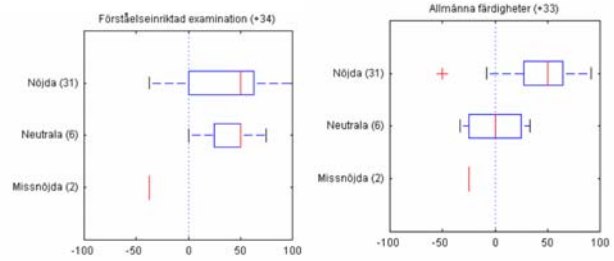
De positiva erfarenheterna från studentmedverkan i utformningen av skriftliga tentamensuppgifter sammanfattas nedan i relation till CEQ-skalorna [1], med stöd från CEQ-analysen 2005. CEQ-analysen innehåller dock studenternas samlade upplevelser av alla kursmoment och inte bara tentamensprobleminlämningarna, så det är svårt att säga hur exakt vad i upplevelsen som beror av detta kursmoment.



- *God undervisning*. Varje vecka får studenterna återkoppling på hur de har förstått teorin genom den skriftliga bonusbedömningen som kursansvarige ger. De kan också läsa de andra gruppernas tentamensproblem med tillhörande bonusbedömning och därmed få en god uppfattning om hur de preseterat i förhållande till kursens kunskapsmål.



- *Tydliga mål*. Då tentamensproblemen bedöms utifrån kursens inlärningsmål och studenterna uppmanas att utforma sina tentamensproblem så att de tydligt fokuserar på målen, blir kursmomentet ett sätt att konkretisera dessa.
- *Lämplig arbetsbelastning*. Genom att chansen till bonus kräver veckovis inlämning så stimuleras studenterna att studera inför tentamen kontinuerligt under kursens gång och arbetsbelastningen blir därmed jämnt fördelad. De låga poängen i CEQ beror förmodligen på att projektet i kombination med övriga kursmoment upplevdes krävande.



- *Förståelseinriktad examination*. För att få full bonus krävs det att frågorna inte bara testar minneskunskap utan går på djupet och testar förmåga att resonera, tillämpa, jämfora och värdera. Denna djupinrikting kommuniceras i kriterierna för bonusbedömningen och befästes i den skriftliga återkopplingen.
- *Allmänna färdigheter*. Studenterna arbetar i grupp med att göra tentamensproblemen och de tränar samtidigt sin skriftliga förmåga.

Slutsatsen av erfarenheterna är att lärarnas arbetsinsattser med detta kursmoment betalar sig väl i inlärningseffekt, med tanke på det goda tentamensresultatet och besparingen i arbetsinsatsen vid omtentamensbedömning.

Referenser

[1] P. Ramsden, *Learning to teach in Higher Education*, Rutledge, 1992.
[2] Kursens hemsida: http://serg.telecom.lth.se/education/ETS170/
[3] Alla tentamensproblem som lämnats in under de två kursomgångarna: http://serg.telecom.lth.se/education/ETS170/tenta/tentaproblem.html