LINKÖPINGS UNIVERSITET

AutoMaster

Design, implementation och utvärdering av ett läroverktyg.

Erik Kanebrant

2014-06-08

Kandidatuppsats inom Kognitionsvetenskap

Institutionen för Datavetenskap

ISRN: LIU-IDA/KOGVET-G--14/009--SE

Handledare: Arne Jönsson
Examinator: Mathias Broth

Sammanfattning

Detta arbete utgör specifikation, implementation och utvärdering av ett läroverktyg sett till hur man säkerställer att en ensam utvecklare lyckas i varje enskild problemdomän. Detta nås genom en metodik för kravinsamling och specificering grundad i Co-Creation, designmönster och kodstrukturering som stöd i själva utvecklingen samt agil utvecklingsmetod sett ur perspektivet av en ensam utvecklare.

Innehållsförteckning

1 Inledning 1
1.1 Master-projektet 1
1.2 Avgränsning 2
1.3 Frågeställning 2
2 Bakgrund 2
2.1 Specifikation och kravgenerering 3
2.2 Kognitiva artefakter som stöd i utveckling 3
2.3 Designmönster 3
2.4 Kodstruktur och kommentering 4
2.5 Stöd i agila utvecklingsmetoder 5
2.5.1 Problematik för ensam utvecklare 5
3 Metod 6
4 Kravinsamling 7
4.1 Diskussion med projektledare 7
4.2 En första pappersprotyp 8
4.3 Fokusgrupp 9
4.3.1 Genomförande 9
4.3.2 Resultat 10
5 Specifikation 11
5.1 Funktionella krav 11
5.2 Identifierade krav för framtida utveckling 11
5.3 Interaktion och navigering 12
5.3.1 Hub and spoke 12
5.3.2 Grafisk design 13
5.4 Systemarkitektur och designmönster 14
5.4.1 Model-View-Controller 14
6 Implementation av systemarkitektur 15
6.1 Java och Spring Framework 15
6.2 Datalagring och hantering 15
6.2.1 Hibernate 16
6.2.2 Java Entities 16
6.2.3 Inversion of Control 17
6.2.4 DAO 17

6.2.5 Domänlogik 18
6.3 Vyer och presentation på skärmen 18
6.3.1 CSS 19
6.3.2 JavaScript och jQuery 20
6.4 Övriga verktyg och webbserver 20
6.4.1 Spring Security 20
6.4.2 Lösenordsgenerering 20
6.4.3 Apache Maven 20
6.4.4 Apache Tomcat 7 21
7 Sprinter 21
7.1 Sprint 1 21
7.2 Sprint 2 21
7.3 Sprint 3 22
7.4 Sprint 4 22
7.5 Sprint 5 22
8 Utvärdering 23
8.1 Intern slututvärdering 23
8.2 Användarfeedback från formulär 23
8.3 Avslutande utvärdering 24
9 Diskussion 25
Litteraturförteckning 28
Appendix A 1
Appendix B 1

1 Inledning

Detta arbete syftar till att specificera, designa och implementera en första del av ett läroverktyg under arbetsnamnet AutoMaster. Ett verktyg där elever genom läsförståelsetester får hjälp att bedöma sin läsförmåga och med hjälp av denna information kunna ta del av texter av en relevant svårighetsgrad. Denna första del ska dock vila på en systemarkitektur designad och implementerad för det slutliga kompletta verktyget. Fokus i detta arbete ligger på hur en ensam utvecklare och designer givet dessa kriterier lyckas specificera, designa och implementera ett användbart och önskvärt system. Detta säkerställs genom en välgrundad metodik för kravinsamling och specificering, väl valda metoder och designmönster för implementation samt en relevant utvecklingsmetodik.

1.1 Master-projektet

Detta verktyg är den tänkta slutprodukten av ett större projekt, Master (Mining textual data for simplified reading), ett samarbete mellan Linköpings Universitet, Göteborgs Universitet samt Uppsala Universitet för att främja elevers språkutveckling och kunskapsbyggande genom individanpassade texter. Projektgruppen består av forskare inom fälten läsbarhet, språkinlärning, lingvistik, språkteknologi samt pedagogik.

Projektets syfte är att information ska vara tillgänglig för samtliga elever och inte bara dem som har den rätta läsnivån för texten den presenteras i. Detta uppnås genom att förse eleven med individanpassade texter samt språkanpassning av befintliga texter. I utgångsläget riktar sig projektet till elever mellan 10 och 15 år.

För att uppnå detta utvecklas ett mätverktyg för att gradera elevens läsförmåga, dessa mått används för att skapa en läsprofil för eleven med syftet att förse eleven med läsmaterial som matchar dennes profil. Det ingår även i projektet att automatiskt sammanfatta text och förenkla den till nivåer anpassade för enskilda elever.

Projektets tänkta slutprodukt är en webbaserad tjänst där lärare kan:

- Låta elever (10 – 15 år) genomföra läsförståelsetest för texter på olika svårighetsnivåer i syfte att få fram en profil för sin läsförmåga för dessa texter.
- Med hjälp av profilen få stöd för att söka fram texter på sådan svårighetsnivå som eleven kan klara av att läsa på ett tillfredsställande sätt.
  eller;

- Ta andra texter och förenkla/sammanfatta dem till en sådan svårighetsnivå som passar den enskilda eleven

Projektet varar över en treårsperiod och under varje år testas texter av en viss genre; skönlitteratur, samhällsorientering och naturvetenskap. Varje genre testas på elever i årskurs 4, 6 och 8. Syftet är att trimma måtten på läsbarhet och utvärdera testmetoden, i utgångsläget utförs dessa tester genom pappersutskick som sedan returneras. Efter att resultaten analyserats och elevens läsprofil genererats skickas resultatet tillbaka till ansvarig lärare.

1.2 Avgränsning

Detta arbetes syfte är att designa och utveckla en första version av webbverktyget. Verktyget ska kunna användas i vanliga webbläsare, i utgångsläget främst på persondatorer. Systemet bör utformas på ett sådant vis så det enkelt kan påbyggas, vidareutvecklas och användas på flera plattformar. Grunden ska läggas för systemets tänkta fulla funktion men gränssnitt ska kunna implementeras och lanseras för separata funktioner under projektets gång. I detta arbete kommer den del av systemet som skall användas för att kommunicera testresultat till lärare att färdigställas med fungerande gränssnitt. Automatisk textsammanfattning och omskrivning kan sättas åt sidan för stunden för att istället implementeras i ett senare skede.

1.3 Frågeställning

För att säkerställa att en ensam designer och utvecklare kan skapa ett användbart och önskvärt system kapabelt att hantera påbyggnad och vidareutveckling parallellt med projektets övriga arbete behöver följande frågor besvaras:

- Hur når man på bästa sätt systemets krav på funktionalitet och koncept med en tydlig förankring i projektets tänkta syfte?
- Hur bör systemarkitekturen designas och implementeras för att stödja den parallella vidareutvecklingen och fungera väl för projektets framtida behov?
- Hur stödjer man på bästa sätt den ensamme utvecklaren sett till utvecklingsmetodik och stöd i själva implementationen?

2 Bakgrund

För att säkerställa en lyckad design- och utvecklingsprocess grundar sig detta arbete i teori gällande kravgenerering, designmönster och kodstruktur samt agil utvecklingsmetodik.

2.1 Specifikation och kravgenerering

För att generera en korrekt specifikation av systemet används metoder tagna främst från fälten tjänstedesign och interaktionsdesign med utgångspunkt i Co-Creation. Detta är en metod som involverar flera parter i designprocessen för att genom högt i tak nå gruppens alla tankar, den hjälper till att förankra systemet hos intressenter men även för att styrka framtida samarbete och förståelse genom en uppfattning av delat ägande av projektet (Stickdorn & Schneider, 2011).

2.2 Kognitiva artefakter som stöd i utveckling

Utveckling av denna typ av system innebär ett användande av ett tio-tal olika programmeringsspråk och ännu fler ramverk och bibliotek. Själva systemarkitekturen i sig, i sin objektorienterade natur, består av långa kedjor av funktioner där anrop inte sällan skickas vidare i tio-tal steg. Detta är en arbetsdomän som i sin natur lämpar sig väldigt illa för människor att arbeta i och koden måste konstrueras på sådant vis att den som arbetar med den behöver hålla så få saker som möjligt i minnet (Wilson, o.a., 2013). Människan beskrivs av Donald Norman som ologisk, ogrammatisk och felande, i tillägg till detta är människan lättdistraherad och långa tankegångar är som en följd väldigt kognitivt krävande (Norman, 1993). Att då arbeta i en strikt logisk domän som är oförlåtande för dessa brister är för att underdriva väldigt problematiskt för en ensam människohjärna.

Människan är dock väldigt bra på att kompensera för dessa brister med hjälp av kognitiva artefakter, föremål eller koncept som i samspel med användaren utför arbete på en betydande högre nivå än den isolerade individens förmåga (Norman, 1993). Ett flitigt användande av kognitiva artefakter ger ett starkt stöd för en ensam utvecklare och designer i en okänd problemdomän att lyckas designa och implementera ett system. Följande punkter används för att ge stöd i dessa problematiska områden och säkerställa en lyckad utvecklingsprocess:

- Designmönster
- Kognitivt stöd i själva kodstrukturen
- Stöd i agila utvecklingsmetoder

2.3 Designmönster

Designmönster bygger på abstrakta kognitiva representationer i syfte att ge utvecklaren stöd i designen och strukturen hos ett system (Mangalaraj, Nerur, Mahapatra, & Price, 2014). Utan designmönster som stöd är det svårt att finna kriterier för när ett visst problem är löst då lösningen ofta arbetas fram parallellt med att problemet definieras (Zhang & Budgen, 2012). Utan stor tidigare erfarenhet av liknande system och utan stöd i designmönster blir resultatet allt som oftast en unik otestad lösning på var och ett av de problem som utvecklaren möts av (Gamma, Helm, Johnson, &

Vlissides, 1995). Detta resulterar i en kodstruktur som både är väldigt svår för utomstående att ta del av men även i många snarlika lösningar på flera väldigt likartade problem som med rätt designprincip från första början hade kunnat rymmas i en och samma generiska lösning. Designmönster låter en oerfaren utvecklare ta del av en stor samlad erfarenhet av generisk problemlösning och design och själv använda detta för att skapa egna fungerande lösningar (Zhang & Budgen, 2012). Ett designmönster kan alltså ses som en behållare av en stor mängd erfarenhet inom en specifik problemdomän där de bästa generiska lösningarna och designprinciperna ryms. Det faktum att designmönster är kunskapsbaser som nyttjas och utvecklas av sina användare gör att de kan ses som externa kognitiva artefakter (Mangalaraj, Nerur, Mahapatra, & Price, 2014). I tillägg till detta tillåter en tydlig dokumentation över vilka designmönster som används en smidig kommunicering av koden till framtida utvecklare av systemet (McConnell, 2004).

2.4 Kodstruktur och kommentering

Utvecklare i mindre projekt tenderar att designa funktioner och klasser medan de byggs, även i mindre system leder detta till inkonsekvens och en kodstruktur som till slut blir väldigt svår att överblicka (McConnell, 2004). Även om man följer en större designmönsterprincip är det viktigt att noga tänka igenom hur en enskild klass eller funktion ska struktureras, detta görs enklast genom att i pseudo-kod beskriva hur den färdiga koden kommer att se ut och vad den gör (McConnell, 2004). Kohesionen i klasser ska vara så hög som möjligt, med detta menas att det som ryms i en klass ska vara tätt sammankopplat till klassens syfte (McConnell, 2004). Hög kohesion förenklar betydligt arbetet för utvecklaren då tydligt sammankopplade funktioner underlättar för det annars väldigt krävande kognitiva arbetet att bibehålla samband mellan en rad abstrakta, till synes orelaterade begrepp. Varje klass ska utgöra en tydlig informationsinkapsling, med detta menas att det ska vara uppenbart vad som ska skickas in, men lika uppenbart vad som kommer tillbaka, utan att behöva sätta sig in själva klassens struktur. Vidare ska klasserna i sig och de paket de tillhör ska ordnas på ett hierarkiskt vis som återspeglar den inbördes funktionaliteten. Detta hjälper utvecklaren att navigera i koden och arbeta i en isolerad nivå där information och funktionalitet kan bearbetas hierarkiskt istället för att överflödas av all relaterad information på en och samma gång (McConnell, 2004). Egna kommentarer i själva koden hör till de viktigaste redskapen för att hjälpa sig själv, men även andra som ska ta del av koden. I ett system som ämnar vara så styrt som möjligt av designmönster och de diskuterade principerna så används kommentarer främst i syfte att förklara designval och eventuella avvikelser från dessa.

2.5 Stöd i agila utvecklingsmetoder

Grundtanken bakom agil utveckling är i mångt och mycket en tydlig involvering av olika intressenter och en iterativ arbetsgång där målet nås genom avklarande av mindre väldefinierade deluppgifter (Jongerius, 2012). Det iterativa arbetssättet medför flera fördelar. För det första kräver det att projektet i varje iteration, sprint, skapar någonting som kan köras och testas av användare och intressenter. Syftet är att mellan varje iteration involvera användare och intressenter för att behålla en tydlig förankring i användarcentrerad design men även för att utifrån fungerande kod kunna specificera konkreta uppgifter och önskade funktioner inför nästa iteration (Jongerius, 2012).

Agil utvecklingsmetod knyter projektet till ett fysiskt rum med en central arbetsvägg, i många fall en whiteboardtavla, som fungerar som projektets arbetsyta. I regel ryms denna vägg av en lista över projektets deltagare och intressenter, user stories och övergripande schema för projektet, men kanske viktigast är projektets faktiska uppgifter som skrivs ner på lappar tillsammans med en uppskattad lösningstid och fästs vid väggen. Detta system av projektmedlemmar och artefakter som knyts till ett rum kan ur ses som ett kognitivt system (Sharp, Robinson, Segal, & Furniss, 2006). Styrkan ligger i att projektets information presenteras på en gemensam fysisk arbetsyta med resultatet av ett effektivt kognitivt system som distribuerar sin kunskap och kapacitet över såväl projektmedlemmar som artefakter (Sharp, Robinson, Segal, & Furniss, 2006).

2.5.1 Problematik för ensam utvecklare

Denna utvecklingsmetod är dock starkt knuten både till en större utvecklingsgrupp och arbetande i par. Parprogrammering går ut på att en person styr, d.v.s. skriver själva koden, medan den andra personen observerar och kommenterar olika lösningar med förhoppningen att en syntes ska uppstå som är mycket starkare än vad den enskilde utvecklaren skulle ha producerat (Nosek, 1998). Då bägge dessa viktiga aspekter oundvikligen uteblir är det viktigt att diskutera vilka konsekvenser detta får för systemet och hur det kan tänkas förebyggas eller mitigeras.

Studier har visat på att parprogrammering är som mest effektivt när man parar ihop en expert med en mindre erfaren utvecklare, experten bör då styra arkitekturen av systemet och den mindre erfarna utvecklaren skriva den mesta av koden (Dawande, Johar, Kumar, & Mookerjee, 2008). Detta höjer prestationen hos det sammansatta paret sett till kvalitet på slutprodukten och sammanlagd tidsåtgång än om de hade arbetat individuellt (Dawande, Johar, Kumar, & Mookerjee, 2008). Förhoppningen är att en del av denna typ av expertis istället fås genom en kraftig användning av designmönster, då dessa som diskuterats tidigare bidrar med expertis inom just systemarkitekturen.

I bristen på input och synande av koden är en lösning att lägga ett problem man arbetat på åt sidan i några dagar, när man återgår till problemet har man blivit av med de tankegånger som präglade

problemlösningen senast och man kan förhoppningsvis se på det med nya ögon (McConnell, 2004). Idén med ett gemensamt projektrum kommer inte heller att finnas kvar. Kvar är dock en arbetsyta för projektet med uppgifter, scheman och övrig information som kommer att fungera likt den större whiteboardtavlan, även i användningen av en ensam utvecklare fortsätter dessa artefakter att vara kraftfulla verktyg.

3 Metod

Inbördes metodik diskuteras mer utförligt när de återkommer i sina respektive faser, i arbetet nyttjas följande arbetsgång:

- Intervju med projektledare i iterationer för att få fram en första prototyp.
- Utvärdering av prototyp i fokusgrupp med övriga projektmedlemmar.
- Utformande av en specifikation bestående av funktionella krav, grafisk design samt det övergripande designmönster som lämpar sig bäst för både tjänsten i sig men även till sättet den kommer att utvecklas.
- Val av plattform och övriga ramverk samt implementation av systemarkitekturen.
- Utveckling i iterationer där tjänstens utveckling kontinuerligt demonstreras och diskuteras med projektledare.
- Utvärdering av slutlig version med övriga projektmedlemmar innan lansering.
- Utvärdering av användarfeedback efter lansering och slututvärdering av systemet med projektmedlemmar.

Projektmedlemmarna är experter på sina respektive områden inom pedagogik, lingvistik, läsbarhet och språkinlärning. I tillägg till detta svarar projektet mot ett ansökningsdokument som specificerar gränserna för systemets omfång. Utifrån dessa premisser antas alla funktionella krav kunna nås genom arbete med projektets medlemmar utifrån principen Co-Creation. Slutanvändare, d.v.s. lärare och elever, rekommenderas istället att involveras när gränssnitt, interaktion och designspråk ska utformas mer grundligt.

Då metoderna för kravinsamling är inriktade på funktionalitet, struktur och koncept så är det problematiskt att motivera en grafisk design utifrån detta. Systemet kräver dock en grafisk design dels för att stödja utvecklingsarbetet, men även för att tillgodose för sina första användare innan ett mer utförligt designarbete sett till interaktion och grafik utförts. Den grafiska designen i systemet

utgår därför från beprövade design-principer valda efter tänkt målgrupp och användning med förhoppningen att detta ska vara ett tillräckligt bra substitut för stunden.

4 Kravinsamling

Kravinsamlingen är uppdelad i en första del av diskussioner med projektledare samt en andra del bestående av en fokusgrupp med övriga projektmedlemmar. Förhoppningen är att detta arbetssätt ska resultera i en kravspecifikation som motsvarar projektets tänkta syfte med fokus på funktionalitet och koncept.

4.1 Diskussion med projektledare

Syftet med dessa diskussioner är att först skapa en bred bild över det tänkta systemet för att sedan reducera denna till tillräckligt tydliga funktionella krav i syfte att skapa en första pappersprototyp. Denna pappersprototyp är endast avsedd att visa på hur systemet är utformat, vad som är möjligt att göra för användaren, hur funktionalitet grupperas samt hur navigeringen fungerar. Den grafiska designen är alltså för stunden ovidkommande och är så småningom föremål för egen undersökning och specificering.

Det första mötet fick formen av en fri diskussion med tanken att i breda drag komma åt övergripande koncept, systemets huvudsyfte, tilltänkta användare samt hur systemet ska tillgängliggöras. Utifrån detta möte kunde det fastställas att systemet skulle uppnå följande övergripande kriterier:

- Systemets huvudsakliga syfte är att fungera som ett verktyg för avgörande av en elevs läsnivå baserat på läs- och ordförståelsetester.
- Systemet ska användas av elever för att utföra sagda tester. Det ska även användas av lärare för att ta del av elevens testresultat.
- Systemet ska vara webbaserat och kunna nås och användas i en vanlig webbläsare.

Utifrån dessa första kriterier skapades en grov skiss över systemet. Denna skiss är inte i någon mening en bild av hur ett färdigt system kommer att se ut. Tanken är att genom visualisering enklare kunna nå de krav på funktionalitet och användande som systemet kräver (Goodwin, 2009) och på så vis reducera de övergripande kraven från det första mötet till mer konkreta idéer inför nästa diskussion.

I efterföljande iteration diskuterades främst tekniska funderingar som uppstått i det tidigare skissandet. De kretsade kring en mängd små frågor som i vilken ordning tester ska tas, om dessa är

tidsbegränsade, hur elevens result ska representeras etc. Efter detta diskuterats började en mer komplett bild av systemet bildas. På samma vis som i tidigare iteration skapades nya skisser över systemet och fler frågor, fast på en ny nivå, uppstod.

I den sista iterationen låg frågorna främst på konceptnivå eller berörde mer övergripande designval. Ett urval av frågorna som diskuterades:

- Väljer man i vilket ämne man vill göra testet, eller måste man göra alla tre test på en gång?
- Är detta en öppen plattform, kan man skapa en egen användare på sidan? Eller får eleven användaruppgifter av läraren?
- Används tjänsten av eleven tillsammans med läraren? Sitter eleverna i helklass eller är läraren direkt tillgänglig under hela testet?
- Påverkar resultatet på det ena testet profilen i ett annat ämne?
- Vill man kunna återta testet när som eller ska detta tidsbegränsas?
- Hur bestäms vilken text som presenteras först i ett givet test?
- Hur strikta regler ska det finnas när ett test väl har påbörjats gällande tidsbegränsning, möjlighet att göra om, korrigera eller backa?

Frågor gällande huruvida texter ska lagras i systemet och lärarens roll i elevens användande kvarstod och skrevs ner för att diskuteras med övriga projektmedlemmar.

4.2 En första pappersprotyp

Utifrån de idéer, krav och funktionalitet som framkommit ur diskussion med projektledare gjordes en första pappersprototyp över systemet efter de tankar som nämndes inledningsvis i föregående avsnitt. Pappersprototypen består av ett papper i A5-format för varje vy i systemet. Alla knappar i prototypen har en motsvarande vysida. Prototypen är medvetet gjord i så enkel grafisk design som möjligt och i ett obundet format (det finns inget skalfönster som visar om systemet körs i en webbläsare eller i en windowsmiljö etc.), se figur A.1 i appendix för exempel. Detta är för att följande fokusgrupp inte ska börja driva åt diskussioner kring grafisk utformning och tycke kring färg och form utan istället fokusera på systemets funktionalitet.

Pappersprototypen utvärderades inför fokusgruppen i ett möte med projektledare som följde de former som var tänkta för fokusgruppen. Inledningsvis presenterades endast start-vyn och diskussionen drevs i mångt och mycket av ”och om jag trycker där, vart kommer jag då?” varpå nya vyer vändes upp och följdes av en öppen diskussion kring deras innehåll och funktion.

Resultatet var en diskussion som kretsade kring funktionalitet och koncept. Pappersprototypen fungerade väldigt väl som stöd i diskussioner kring önskvärt beteende och hur man såg på systemet i stort. Att ha denna diskussion kring en konkret systemdesign fungerade även som en bekräftelse att bägge parter var ense om systemets syfte.

Denna utvärdering av prototyp och metod ledde till några revideringar av själva pappersprototypen, främst i gruppering av funktioner i olika vyer, men framförallt styrkte det metodens möjligheter att frambringa diskussioner kring funktionalitet snarare än det så mycket mer lättillgängliga grafiska utformandet.

4.3 Fokusgrupp

Fokusgrupp som metod för kravinsamling från övriga projektmedlemmar valdes dels av praktiska skäl, projektmedlemmarna är annars utspridda men ett gemensamt möte var inplanerat där en timme kunde vigas åt diskussioner kring utveckling av verktyget. Fokusgrupp är även en god metod att använda i början av ett projekt som grund för vidare arbete (Goodwin, 2009). Inför fokusgruppen skickades en lista ut med förberedande frågor att fundera kring innan själva fokusgruppen tog plats. Denna lista bestod främst av projekttekniska frågor så som hur elevens integritet behandlas i systemet, i övrigt delgavs bara att de skulle få ta del av en första pappersprotoyp att ge synpunkter på. Då projektet är nystartat och övriga projektdeltagare ännu inte har skapat sig en detaljerad bild över hur systemet kommer att se ut så var tanken att inte ställa frågor kring detta i förhand för att istället kunna få med denna första reflektion i själva fokusgruppen.

4.3.1 Genomförande

Fokusgruppen varade i en och en halv timme och bestod av fri diskussion kretsande kring pappersprototypen och korta scenarietexter. Formen behölls så fri som möjligt, frågor uppkom när någonting var oklart eller vad för idéer som låg bakom specifika designval men i stort var det en öppen diskussion samtliga deltagare emellan.

Som komplement till pappersprototypen i syfte att enkelt få igång en diskussion och kunna styra den till funktionalitet om den började fokusera på utseende skapades ett fåtal användarscenarier. Scenarier i kombination med en lo-fi pappersprototyp riktar in diskussionen mot funktionalitet och koncept. (Arvola & Johansson, 2007). Scenarier konkretiserar annars abstrakta tankar kring användande (Goodwin, 2009) och lämpar sig därför väl för att fokusera diskussionen kring en specifik typ av användning som är lättförståelig för deltagarna i fokusgruppen. De scenarier som skapades inför fokusgruppen var följande:

”En elev pendlar kraftigt i sina resultat och det krävs fler texter än vanligt för att bestämma en läsprofil. Efter att ha lämnat in svaren till den fjärde texten väljer hen att avsluta för att fortsätta med nästa text imorgon.”

”Läraren loggar in i systemet för att förbereda texter inför morgondagens lektion. Efter att ha valt ut en text av varje svårighetsgrad skickas de ut till klassen, för en elev väljer läraren istället en text av en högre svårighetsgrad än vad profilen visar.”

”… För detta kan läraren inte hitta relevanta texter för alla läsnivåer utan bestämmer sig för att göra en automatisk omskrivning av en text till flera svårighetsgrader. Efter att ha läst igenom de omskrivna texterna skickar läraren texterna till eleverna i klassen.”

”Läraren har märkt av en elev som gjort snabba framsteg i sin läsning och loggar in i systemet för att låta eleven ta testen på nytt. Nästa gång eleven loggar in måste hen göra testen på nytt för att få en läsprofil och komma åt textdatabasen. ”

”Läraren ska söka efter en text. Först väljs huvudämnet samhällskunskap för att sedan kompletteras med geografi. Läraren skriver även in sökorden Sverige och demografi och väljer även att filtrera på tre svårighetsgrader.”

”Efter att ha valt svenska som ämne och markerat de elever hen letar efter texter till gör läraren en sökning i textbanken. Hen väljer ämnet skönlitteratur och lägger till sökordet saga.”

4.3.2 Resultat

Detta ledde både till en tydlig kravlista på önskvärda funktioner och ett tydligare koncept i stort. Denna diskussion med pappersprototyp som stöd gav även deltagarna möjligheten att konkretisera hur de hade föreställt sig systemets utformning. Genom att peka och flytta på papper uppenbarade sig vissa skilda tankar kring systemets egentliga syfte och deltagarna förstod på så vis bättre varandras visioner. Detta var till ett stort stöd för arbetet med specifikationen då systemets utformning inte nås enbart genom en lista över funktionella krav. För att utforma en fungerande specifikation som tjänar sitt syfte så måste man ha förstått det önskade konceptet, eller vad intressenterna faktiskt vill se för slutprodukt (Goodwin, 2009). Vissa av scenarierna bygger på de frågor som kvarstod efter de inledande diskussionerna med projektledare och var till stor hjälp för att reda ut dessa frågor inför specificering av systemet.

5 Specifikation

Specifikationen utgörs av en lista på övergripande funktionella krav, en lista över identifierade krav för framtida utveckling, en beskrivning av huvudsakligt designmönster samt beskrivning av designspråk och navigationsschema.

5.1 Funktionella krav

Nedan följer en lista över funktionella krav som ska rymmas i denna utvecklingsfas.

- Lägga grunden för att låta elever (10 – 15 år) genomföra läsförståelsetest för texter på olika svårighetsnivåer och få fram en profil för sin läsförmåga för dessa texter.
- Lagra resultat från redan utförda tester.
- Elevens resultat ska lagras som två sammanvägda poänger för läsning samt antal rätt ord på ordtest. Detta ska göras för varje text eleven testas för.
- De tre olika måtten ska generera en resultattext efter särskilda kriterier. Denna ska innehålla:
    - Beskrivning av texten i fråga.
    - Beskrivning av hur man lyckats med läsförståelsefrågorna.
    - Beskrivning av hur man lyckats med ordförståelsetestet.

- För varje text med tillhörande ord- och lästest genereras en resultattext över elevens prestation.
- Lärare ska kunna logga in i systemet och ta del av var och en av sina elevers resultat för samtliga texter de läst.
- Läraren ska enkelt kunna skriva ut elevens resultat.
- Elevens namn får inte lagras i systemet, istället ska ett ID lagras som elevens lärare sedan kan använda för att identifiera en elev.
- De användartyper som ska finnas är lärare, elever och administratörer.
- Systemet ska vara webbaserat.
- Systemet ska vara plattformsoberoende i så stor utsträckning som möjligt.
- Systemet ska vara mål för vidareutveckling och påbyggnad över en tre-års-period.

5.2 Identifierade krav för framtida utveckling

Dessa krav ska inte implementeras i detta skede, dock så har de framkommit ur kravinsamlingen. De är heller inte krav i traditionell mening då vissa av dem fortfarande kan vara föremål för kraftig förändring. De används i implementeringen och vidare specifikation för att ge en tydligare bild av

systemets övergripande koncept. Vid avvägningar i implementationen av övriga funktionella krav tas dessa krav fortfarande i beaktning.

- Bearbeta texter genom att sammanfatta dem och skriva om dem till lätt svenska anpassad efter elevens läsprofil.
- Från elevens samlade prestation inom ett ämne kan elevens läsprofil bestämmas.
- Eleven ska kunna ladda upp en egen text eller länka till en text på nätet för att få reda på vilken läsbarhetsnivå texten befinner sig på i syfte att se om den lämpar sig för dennes läsprofil.
- Systemet ska enbart lagra texter som används i själva testen eleven utför, det ska alltså inte finnas någon textbank att söka i.
- En sammanvägd läsprofil ska gälla för en viss tidsperiod, sedan ska eleven testas på nytt.
- Läraren ska kunna gå in i systemet och ändra så att en elev kan göra om testen innan denna tid gått ut.
- Läraren ska kunna använda tjänsten för att söka fram texter individanpassade till sina elevers läsprofiler.
- Det ska finnas möjlighet för läraren att kommunicera med sina elever genom tjänsten.
- Läraren ska kunna ladda upp egna texter eller länka texter från hemsidor för att få reda på textens läsnivå.
- Genom en klasslista med elevernas sammanvägda läsprofil ska läraren sedan kunna sammanfatta och/eller göra en automatisk omskrivning av texten till samtliga klassens elevers läsprofiler.

5.3 Interaktion och navigering

Systemet kommer att följa principen en aktivitet per varje vy, med detta menas att den information som för stunden presenteras på skärmen enbart gäller för den aktivitet som just då pågår. Det finns alltså inga konstanta menyer, inga fasta element eller annat innehåll som stannar kvar i vyn vid byte av aktivitet. Den enklaste formen av ett interaktivt system är alltid, om de funktionella kraven tillåter det, små isolerade aktiviteter (Tidwell, 2011) och detta bör eftersträvas i så stor utsträckning som möjligt.

5.3.1 Hub and spoke

Detta designmönster lämpar sig väl för systemet då det är aktivitetsbaserat, med detta menas att en inloggad användare har ett visst antal aktivitet som systemet låter dem göra. När användaren väl loggat in i systemet kommer den första vyn att fungera som nav för övriga sidor. Användaren utgår alltid ifrån en och samma sida, från denna sida nås alla aktiviteter och navigering utgår ifrån att

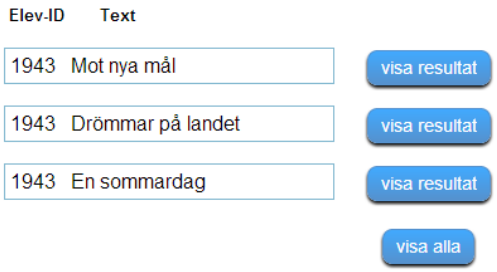
användaren väljer en aktivitet för att sedan gå tillbaka till navet vid byte av aktivitet. Alla sidor har tydliga tillbakaknappar för att återgå till navet. Denna teknik är även fördelaktig för enheter med mindre skärmyta än en traditionell dator (Tidwell, 2011) och är därför fördelaktigt för att hålla systemet så plattformsoberoende som möjligt.

5.3.2 Grafisk design

Då de första användarna av systemet känner till det som ett universitetsbaserat forskningsprojekt valdes ett kallt färgschema som inger en professionell och strikt känsla, färgerna går därför i blått och vitt (Tidwell, 2011).

Figur 1. Exempelsida för valt designspråk.

Knapparna följer samma färgschema men är tonade på ett sätt som tydliggör att de är klickbara. Knappar presenteras konsekvent bredvid korresponderande element. Med detta menas att innehåll i sig inte är klickbart utan nås genom en knapp bunden till innehållet.



Figur 2. Design av knappar och positionering.

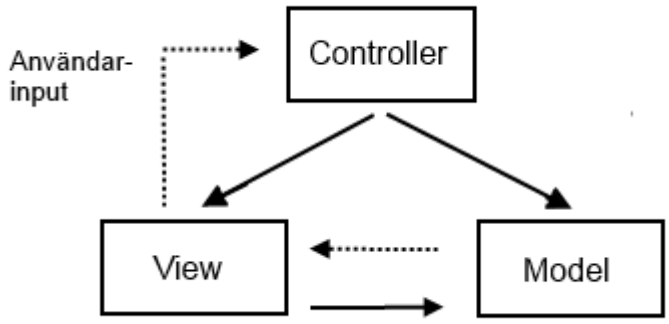
5.4 Systemarkitektur och designmönster

Premisserna för systemets önskvärda funktionalitet, flexibilitet för framtida vidareutveckling samt önskade plattformsoberoende ligger till grund för val av övergripande designmönster att följa vid implementering.

5.4.1 Model-View-Controller

Model-View-Controller är ett designmönster som syftar till att separera data-modeller, skärmrepresentationer och behandling av användarinput från varandra (Gamma, Helm, Johnson, & Vlissides, 1995). Utan denna designprincip till hands klumpas dessa gärna ihop till ett och samma objekt innehållande data, dess presentation på skärmen samt interaktionen med den samma, med följden av återupprepning av likartad kod och en mer svåröverskådlig systemarkitektur.

Modellen är en representation av någonting verkligt, i praktiken är det data eller gruppering av relaterad data. View är vad användaren möts av när denne använder systemet och Controller avgör vilka modeller som ska presenteras i vyn samt hur användarens input påverkar systemet (Dey, 2011).



Figur 3. Förenklad MVC-modell.

MVC som övergripande designmönster valdes på grund av dess styrkor på en rad punkter:

- Designprincipen håller gränssnittet och presentationen av allt innehåll användaren möts av skilt från datahantering och domänlogik. Detta effektiviserar implementering genom att funktioner för datahantering och domänlogik enbart behöver skrivas en gång och inte på varje ställe som resultatet av dem ska visas. Det leder även till en tydligare kodstruktur som är enklare att följa och arbeta i.
- Det är möjligt att fristående vidareutveckla gränssnittet och de vyer systemet innehåller utan att göra stora ändringar i bakomliggande kod.

- Det låter även systemet att vara till stor grad plattformsoberoende då gränssnittet kan omformas för en rad olika enheter och operativsystem så länge som det har möjlighet att kommunicera med det underliggande programspråket.

Även det faktum att implementeringen sker parallellt med ett längre projekt i stort argumenterar för att följa denna modell då systemet blir väldigt mottagligt för förändringar och införandet av oanade funktionella krav, så länge som den grundläggande funktionaliteten är korrekt specificerad.

6 Implementation av systemarkitektur

Specifikationen bestämmer inte vilka programmeringsspråk, ramverk eller plattformar etc. som ska användas vid implementationen. Dessa härleds dock utifrån de funktionella kraven, främst då:

- Systemet ska vara plattformsoberoende i så stor utsträckning som möjligt.
- Systemet ska vara webbaserat.
- Systemet ska hantera säker inloggning.
- Systemet ska lagra en stor mängd data för presentation för användaren.
- Systemet ska vara föremål för vidareutveckling och påbyggnad över en tre-års-period.

Utöver krav från specifikationen är implementationen i val av övriga designmönster och ramverk starkt präglad av att ge stöd åt själva utvecklingen, och utvecklaren.

6.1 Java och Spring Framework

Valet av huvudsakligt programmeringsspråk föll på Java av två anledningar. Java är fritt att använda och stöds av en stor användarbas som förser varandra med guider, diskussionsforum och lösningar på problem. Java är även plattformsoberoende i den utsträckningen att alla enheter med stöd för Java Virtual Machine kan tolka och köra koden utan krav på modifikation (Cadenhead, 2013).

Spring Framework är ett ramverk för effektivare utveckling av webbapplikationer i Java, det ger även ett komplett stöd för utveckling efter designmönstret Model-View-Controller (Deinum, Serneels, Yates, Ladd, & Vanfleteren, 2012). Det ger även stöd för designmönstret Inversion of Control (Dependency Injection) som diskuteras senare.

6.2 Datalagring och hantering

Då systemet hanterar en stor mängd relaterad data med unika nyckelvärden föll valet på att samla data i en relationsdatabas, plattform för detta är MySQL för dess goda stöd i Java och Spring

Framework. Även lagring av systemets användare med användaruppgifter och befogenheter lagras med fördel i denna typ av databas. Användarnas lösenord lagras med kryptering, skulle databasens integritet äventyras så är lösenorden på så vis säkrade. Alla användaranrop som på olika sätt manipulerar databasen kontrolleras för specialtecken för att förhindra SQL-injektioner. Uppkopplingen till den externa databasen och anrop mellan den och applikationen hanteras av en connection pool som fås av JDBC, ett API (application programming interface) för databasuppkoppling i Java. Connection pooling används för att återanvända uppkopplingar mot databasen. I annat fall hade en ny uppkoppling behövts etableras för varje användare varje gång en ny sida visas, något som leder till långa väntetider och ett svåranvänt system (Vohra, 2008).

6.2.1 Hibernate

ORM är en programmeringsteknik för att konvertera data från en relationsdatabas till ett annars inkompatibelt språk, i detta fall Java. Hibernate är ett ramverk som implementerar ORM i Java för att skapa Java-objekt genom anrop till MySQL-databasen. Hibernate använder ett eget SQL-liknande språk för anrop till databasen, Hibernate Query Language. Detta gör att samma anrop kan användas till olika SQL-databaser och behöver således inte ändras om databasen flyttas (Vohra, 2008). När Java-objektet är skapat kan det vidare manipuleras av systemet innan det presenteras för användaren.

6.2.2 Java Entities

För att göra kopplingen mellan applikationen och databasen komplett krävs en representation av databasens tabeller, se figur A.4 i appendix för schema över databasen. För detta används Java Persistence API, JPA. För varje tabell i databasen skapas en motsvarande Java-klass innehållande information om nycklar och kolumner med tillsvarande get- och setmetoder.

```
@Entity(name="Teacher")
@Table(name="teacher")
Public class TeacherEntity {

…

    @Column(name="USERNAME")
    private String username;

…

    public String getUsername() {
        return username;
    }
    public void setUsername(String username) {
        this.username = username;
    }
}
```

Figur 4. Java-Entities för motsvarande databastabeller.

6.2.3 Inversion of Control

Inversion of Control är ett designmönster med syfte att öka modulariteten i ett system och göra det mer mottagligt för framtida expansion av funktionalitet och komplexitet (Gamma, Helm, Johnson, & Vlissides, 1995) men även för att få en mer överskådlig kodstruktur. Principen är att dölja information från implementationsklassen bakom en interface-klass. Ett interface är inom objektorienterad programmering en klass med metoder för att bestämma typen av ett objekt, vad själva objektet sedan innehåller avgörs av implementationsklassen. Denna teknik används i implementationen av DAO, Data Access Object, och domänlogik.

6.2.4 DAO

Data access objekt är ett designmönster för att hantera anrop till databasen. Det bygger på principen att man skapar ett Java-interface med metoder för anrop till databastabeller och deras entiteter. Detta interface implementeras sedan i en sessionsdriven klass som via databasanrop hämtar den data som begärs för stunden.

```
DAO:

public interface TeacherDAO
{
…
    public Integer getIdByName(String name);

}

DAO implemeterat:
@Repository
public class TeacherDaoImpl implements TeacherDAO  {

@Autowired
private SessionFactory sessionFactory;
…
public Integer getIdByName(String name) {
            Integer result;

            Query query = this.sessionFactory.
            getCurrentSession().
            createQuery("select id from Teacher where username = :name").
            setString("name",name);

            result = (Integer) query.uniqueResult();
            return result;
            }

…
}
```

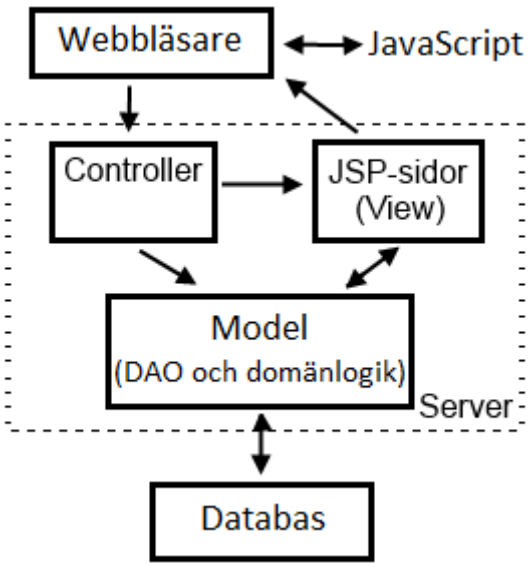
Figur 5. Data Access Object, klasser för interface och implementation

6.2.5 Domänlogik

Domänlogiken, även kallat service eller business logic, är designat på samma sätt som DAO-lagret men är den nivå som sist påverkar objekten innan de skickas till controllern. I detta lager manipuleras data från databasen om så behövs innan den når vyn. I systemet bestäms exempelvis vilken resultattext som genereras genom en funktion som manipulerar elevens testresultat.

6.3 Vyer och presentation på skärmen

Vyfilerna är av typen JavaServer Pages, JSP. JSP är baserat på html men ger i tillägg till vanlig html stöd för servergenererat dynamiskt innehåll i webbsidor, detta låter innehållet på sidan att genereras av den bakomliggande Java-koden. En lista eller objekt skickas från controllern till den begärda sidan och innehållet i dessa presenteras sedan för användaren. Detta kan exempelvis vara en klasslista över elever, denna genereras baserat på vilken lärare som är inloggad och byggs dynamiskt upp av elever kopplade till lärarens klass.



Figur 6. Den övergripande systemarkitekturen i MVC-perspektiv.

JSTL ger stöd för logiska operationer i JSP-filen och används i systemet främst för att visa godtyckligt innehåll från Java-objekt och listor. Prefixet <c:> anges för att använda dessa funktioner. Nedan följer ett exempel från systemet hur kopplingen görs mellan controllern, modellen och vyn.

```
@Controller
public class ResultController {
…
@Autowired
private ResultManager resultManager;
…
@RequestMapping(value="/list", method = RequestMethod.GET)
              public String printResult(ModelMap model) {
…
String name = user.getUsername();
…
model.addAttribute("studentList", resultManager.getStudents(start, stop));
model.addAttribute("username", name);
return "result";
}
```

Figur 7. Controller-klassen.

Controllern svarar mot den begärda url-adressen (/list.jsp) och returnerar sedan en ny vy. Notera ”model.addAttribute("studentList", resultManager.getStudents(start, stop));”. Detta är ett anrop till ett interface i domänlogiken. I implementeringsklassen för denna anropas i sin tur interfacet för motsvarande DAO-klass som i vars implementering själva anropet till databasen görs och elevlistan skapas. ”ResultManager” är alltså namnet på domänlogiken för elevresultat.

Controllern anropar result.jsp för visning och skickar en lista över elever (studentList) samt användarnamn för den inloggade användaren (username). Jsp-filen skapar sedan en html-sida med dynamiskt innehåll, i detta fall användarens användarnamn samt en lista över dennes elever.

```
…
<h2 id="banner">Resultat: ${username}</h2>
<c:if  test="${!empty studentList}">
<c:forEach items="${studentList}" var="res">
…
            <div class="texttrow">
                        <div class="result">${res}</div>
            </div>
…
</c:forEach>
</c:if>
…
```

Figur 8. JSP-filen, hur data visas i vyn.

6.3.1 CSS

All layout och design bestäms av en separat .css-fil, själva .jsp-filerna innehåller endast sidans information så som text, knappar etc. Detta medför stora fördelar om gränssnittet behöver designas om. Att helt bestämma designen i en extern css-fil gör även systemet plattformsoberoende i den

bemärkelsen att en unik css-fil kan skapas för varje plattform systemet är tänkt att visas på utan att behöva göra specifika vyfiler för varje typ av enhet så som en surfplatta, hemdator eller mobiltelefon.

6.3.2 JavaScript och jQuery

JavaScript är ett språk som används för att tillföra funktionalitet och dynamiskt innehåll på klientsidan av applikationen. JavaScript är alltså skilt från applikationens bakomliggande kod och kan endast påverka sådant som hanteras av själva webbläsaren, se figur 6. jQuery är ett JavaScript-bibliotek som förenklar användandet av JavaScript och tillåter användandet av en rad olika färdigskrivna funktioner. I systemet används JavaScript för att bestämma vad som händer när användaren väljer att skriva ut från sidan samt JQuery-objektet ”fancybox” för den dialogruta som visas när användaren väljer att logga ut.

6.4 Övriga verktyg och webbserver

6.4.1 Spring Security

Spring Security är ett ramverk för enkel implementering och hantering av inloggning och användarsessioner. Användarnamn, lösenord samt användartyp lagras i databasen. Användarnamn och lösenord kontrolleras vid inloggning och användartypen avgör vilket innehåll användaren kan visa på sidan. I praktiken betyder detta att en lärare möts av ett visst gränssnitt, elever av ett annat och administratörer av ett tredje. I utgångsläget finns endast inloggning för typen lärare men grunden är lagd för att enkelt lägga till fler användartyper.

6.4.2 Lösenordsgenerering

För att skapa krypterade lösenord på förhand skapades ett skript i PHP (ett serverbaserat skriptspråk) som genererar nio-siffriga lösenord med blandade gemener, versaler och siffror samt deras motsvarande hash-kryptering. PHP används dock inte i systemet i övrigt då denna funktionalitet fås av JSP och JSTL som diskuterats tidigare. Anledning till att inte inkorporera detta i systemet var för enkelhets skull, om systemet ska tillåta användaren att skapa sina egna användaruppgifter behöver nya funktioner för detta skapas.

6.4.3 Apache Maven

Maven används till paketeringen av applikationen samt versionshantering av de externa ramverk och moduler som systemet använder sig av. Applikationen paketeras till en fil som sedan kan överföras till en Java-server som åter bygger upp systemet och gör det tillgängligt för användaren.

6.4.4 Apache Tomcat 7

Då applikationens vyfiler är i formatet .jsp i kombination med ett bakomliggande Java-system krävs det att den körs på en kompatibel webbserver. Apache Tomcat valdes därför att utveckla mot då det är en plattformsoberoende webbserver fri att använda för hosting av Java-applikationer.

7 Sprinter

Med den grundläggande systemarkitekturen uttänkt och i stora drag på plats inleddes en serie sprinter, iterationer, av implementering av det faktiska innehållet i systemet. En sprint varade en vecka, i slutet på varje presenterades vad som åstadkommits för projektledare och nästa sprints innehåll diskuterades i form av specificerade uppgifter som sedan tidsskattades. Specifikationens krav på implementation av bestämd funktionalitet samt ett fungerande gränssnitt för läraren sattes som mål för utvecklingsfasen.

7.1 Sprint 1

Inför den första sprinten bestod den körbara koden som kunde visas i ett gränssnitt av en inloggning som validerar användaruppgifter utifrån databasen och visar relevant sida för relevant användartyp. Inför nästa iteration bestämdes följande:

- Lägg in elevresultat i databasen.
- Resultaten ska generera någon typ av text.
- Presentera relevant klasslista för den inloggade läraren.
- Skapa inloggningsuppgifter för de första användarna.
- Lägg till riktiga användaruppgifter i databasen.
- Bygg ett skal för navigationen och de sidor och innehåll som ska finnas.

7.2 Sprint 2

Inför denna presentation har det som specificerats i föregående sprint implementerats. Ett designat gränssnitt och övrig layout saknas fortfarande men koden var körbar och visades som enkla html-element. Inför nästa sprint bestämdes följande:

- Implementera layout och gränssnitt.
- Färdigställ navigationen och de sidor som ska finnas.
- Resultaten ska generera de korrekta texterna.

Det bestämdes även att feedback från övriga projektmedlemmar fick avvakta då avsaknaden av ett designat gränssnitt skulle vara i vägen för en utvärdering.

7.3 Sprint 3

Inför den tredje sprinten presenterades sidan i sin helhet med ett designat gränssnitt och övrig layout på plats. Inför nästa iteration bestämdes följande:

- Lägg till en ny sida där elevens resultat för samtliga texter visas, det blir för mycket att visa alla elever med respektive texter för en hel klass.
- Lägg till utskriftsknapp vid resultattexten.
- Trimma utskriften så att endast relevant information skrivs ut istället för hela sidan.
- implementera formulär för användar-feedback.

7.4 Sprint 4

Formuläret var ännu inte färdigställt, övriga punkter hade åtgärdats. Inför nästa iteration bestämdes följande att göras:

- Lägg till felhantering för databasanrop.
- Färdigställ formulär, implementera på samma vis som det tänkta formuläret för tester från specifikationen med införande av användargenererad data i databasen.

7.5 Sprint 5

Formuläret är implementerat och grunden har lagts för de funktioner som krävs för implementation av test-formuläret. Felhantering har lagts in för vissa anrop men är inte färdigställt. De sista uppgifter som definierades innan lansering var följande:

- ”Skriv här” ska försvinna när man klickar i boxen på formuläret.
- Det ska finnas en möjlighet att presentera elevens alla tre resultattexter på samma gång. Detta ska helst rymma en A4-sida vid utskrift.
- Minska fontstorleken på sidan överlag.
- Gör färdigt implementationen av felhantering.

Dessa punkter åtgärdades innan testningen i projektgruppen genomfördes, dock så kvarstod en del av felhanteringen att implementeras.

8 Utvärdering

Utvärderingen skedde i tre faser. I den första genomförs en intern utvärdering av tjänsten innan den tillgängliggörs för de första användarna. I den lanserade tjänsten har användarna möjlighet att genom ett formulär utvärdera tjänsten. I en sista fas utvärderas systemets prestation överlag samt dess möjlighet till vidareutveckling och påbyggnad tillsammans med övriga projektmedlemmar.

8.1 Intern slututvärdering

Samtliga projektmedlemmar fick tillgång till de skapade inloggningsuppgifterna och ombads logga in och prova att använda tjänsten innan den tänkta lanseringen. I tillägg till detta skickades ett par punkter de ombads att fundera kring under testningen.

- Sett till tidigare diskussioner och övrig input längs vägen, hur upplever ni tjänsten överlag?
- Hur tänker ni kring strukturen som klasser/elever presenteras på? (först en lista över klassens elever och sedan de texter varje elev läst.)
- Hur fungerar navigationen i tjänsten?
- Den grafiska designen har inte fått ett så stort utrymme i utvecklingen men ni får gärna tycka till om hur den känns.
- Tyck även gärna till om mindre saker, så som vad det ska stå i headern på de olika sidorna etc.

Denna utvärdering gav ett tydligt svar på att tjänsten svarade väl mot vad projektet hade föreställt sig. Det sades ingenting specifikt knutet till de olika punkterna, dock så var de kommentarer som kom in väldigt positiva till tjänsten i sin helhet. Utöver detta resulterade även utvärderingen i identifiering av några små fel och omformulering av vissa texter.

8.2 Användarfeedback från formulär

Formuläret består av åtta frågor, frågorna (3) och (4) är tagna från en svensk översättning av SUS (Göransson, 2011), System Usability Scale, en enkät för utvärdering av användbarhet. Övriga frågor är formulerade med syfte att utvärdera systemet utifrån ett användarperspektiv men även fånga in vilka möjligheter användarna ser med systemet då det är under fortsatt utveckling. Frågorna 1-4 är i Likertskala, fråga 5 är en ja/nej-fråga och frågorna 6-8 besvaras med fritext. Se figur B.9 i appendix för formuläret i sin helhet.

1. Jag föredrar att få tillbaka resultatet i digital form istället för på papper.
2. Det var enkelt att hitta elevens resultat för en specifik text.

3. Jag tycker att verktyget är lätt att använda.
4. Jag tror att många tycker att verktyget är mycket besvärligt att använda.
5. Den typ av test som eleverna utförde är tänkta att i framtiden finnas i detta verktyg, är detta någonting du skulle använda i undervisningen?
6. Om du svarade ja, hur skulle du använda detta verktyg i undervisningen?
7. Vad skulle du vilja kunna göra i verktyget eller använda det till?
8. Övriga tankar.

De svar som inkommit är för få för att dra några egentliga slutsatser men majoriteten svarar positivt på de första fyra frågorna. Ett urval av de kommentarer som inkommit är följande:

Svar på (6):”Som ytterligare instrument för att utvärdera elevens läsning inför betygssättning och omdömen.”

Svar på (7): ”Skulle vilja kunna se vad eleverna klarat o inte klarat d.v.s. få tillgång till själva svarsblanketten.”

Då systemet ännu inte loggar användning kan de få svaren bero antingen på att det är få som svarat på formuläret eller att få av de 60 användare som nu har tillgång till systemet använt sig av det. När loggning väl implementerats och svaret på detta kan nås bör formuläret ses över om det önskas finnas kvar.

8.3 Avslutande utvärdering

Avslutningsvis genomfördes en utvärdering med projektets medlemmar. Denna utvärdering hade formen av en öppen diskussion om verktyget i sig och hur det ska fortsätta sin utformning. Från diskussionen följde att sett till funktionalitet uppfyller systemet de tänkta kraven och de komponenter som implementerats fungerar så som de ska göra i det färdiga verktyget. Överlag målas en väldigt positiv bild upp över hur systemet har utformats. Kvar är dock vissa frågetecken över lärarens pedagogiska roll i det färdiga verktyget, dessa rör i vilken utsträckning eleven ska kunna använda verktyget självständigt och lärarens pedagogiska roll i detta. Å den ena sidan finns ett fullt automatiserat system som låter eleven bestämma sin sammanslagna läsprofil samt själv använda detta för att få tillgång till texter av relevant svårighetsgrad. Det andra alternativet är ett system där läraren väljer ut texter åt eleven baserat på information från systemet om elevens läsförmåga. Vidareutveckling av verktyget måste stödja dessa båda varianter då det heller inte är otänkbart att slutprodukten använder sig av idéer från de bägge. Förhoppningen är att den parallella utvecklingen tillsammans med användarstudier ska reda ut dessa frågor i lagom takt för att när det slutliga valet görs är det förankrat i en enig projektgrupp.

Utöver detta sattes det upp ett antal milstolpar för projektets resterande två år i syfte att stödja den parallella utvecklingen genom att införa en ny komponent åt gången. I ett första skede ska elever kunna svara på testfrågor i systemet, själva texten för testet finns fortsättningsvis på papper tills en designstudie utförts om hur gränssnittet för detta bör utformas. Under det kommande året ska systemet även kunna logga användningen i avseende på hur många som loggar in samt hur de navigerar i verktyget. Det ska även byggas upp en databas av texter och tillhörande testfrågor. I tillägg till detta ska det även skapas sidor som tillåter projektmedlemmarna att i systemet hämta ut och analysera resultatdata från elevers tester. I början av det sista året ska användarna ha tillgång ett funktionsmässigt färdigt system. Det sista året används sedan till användarstudier och design av gränssnitt samt införande av eventuella saknade funktioner.

9 Diskussion

Detta arbete har utrett metoder och tekniker för att säkerställa en väl fungerande slutprodukt skapad av en ensam utvecklare genom samtliga moment från kravgenerering till implementation. Systemets utformning är ett resultat av den behovsbild som specificerades i kravinsamlingen med hänsyn tagen till vidareutveckling, påbyggnadsmöjligheter och det skede projektet befann sig i. Utvärdering med projektmedlemmar visar även att systemet uppfyller den funktion som var tänkt. Arbetet med att säkerställa en stadig grund genom noggranna val av ramverk och designmönster har resulterat i ett system som är mycket tåligt för förändring och införande av nya funktioner. Detta visade sig i det iterativa arbetet i de fem sprinterna, när ändringar av systemet bestämdes mellan sprinter var de enkla att införa trots relativ hög komplexitet, så som införandet av en ny vy som visar resultat för samtliga texter eleven läst.

Systemet innehåller dock inte alla de funktionella krav som återfinns i specifikationen, systemarkitekturen är däremot implementerad för att enkelt införa dessa funktioner. Exempelvis så är feedbackformuläret implementerat på ett sådant vis för att underlätta implementationen av läs- och ordtesten. Avsaknaden av viss felhantering visade sig vara ett problem, ibland uppstår problem med uppkoppling mot databasen och användaren möts av ett generiskt felmeddelande som kan uppfattas förvirrande.

Co-Creation som övergripande metod för kravinsamling har fört med sig flera fördelar, för det första så har det tydligt förankrat systemet i det övriga projektet med resultatet av ett system som svarar väl mot sitt tänkta syfte. För det andra har det nära samarbetet med främst projektledning i

kombination med det iterativa arbetssättet varit väldigt bra för att styra arbetet i en tydlig riktning och föra diskussioner kring idéer och lösningar som ett substitut för en utvecklingsgrupp.

Metoderna för att bestämma gränssnitt, grafisk design och navigering i systemet har fungerat till synes bra sett till de utvärderingar som genomförts. Projektmedlemmarna ställer sig positiva till hur systemet ser ut och navigeras, de få användare som har svarat på formuläret är i regel positiva i frågor gällande användbarhet. För att fullt ut kunna bekräfta detta hade det dock krävts svar från fler användare, något som förhoppningsvis sker när systemet når ut till fler användare.

Det grundliga arbetet med att först designa och bygga hela systemarkitekturen medförde att den senare implementeringen i sprinter enkelt kunde uppnå sitt tänkta syfte, att skapa fungerande komponenter öppna för förändring. Att i implementationen skissa upp klasser och funktioner i förväg i syfte att uppnå en tydlig modularitet med hög kohesion i inbördes klasser gav en väldigt tydlig struktur av systemet som var enkel att arbeta med. Det noggranna valet, och sedan följandet, av designmönster gav ett oerhört bra stöd för själva utvecklingsprocessen. När problem uppstod kunde lösningar på dessa enkelt sökas fram från diverse forum och guider. Problem som uppstår i användandet av designmönster är i regel generiska, med följden av att det finns väletablerade generiska lösningar att finna. Utan denna möjlighet att enkelt finna lösningar på problem hade utvecklingsprocessen sett väldigt annorlunda ut med ett förmodat mindre tåligt system som följd.

Valet av agil utvecklingsmetod har fungerat väldigt väl trots det faktum att utvecklingen utförts av en person. Hade man istället enbart utgått från specifikationen och genomfört hela implementationen utan diskussion och input från projektledare längs vägen hade systemet sett väldigt annorlunda ut och befunnit sig längre ifrån projektets vision. Behovet av att presentera fungerande kod i varje iteration och definiering av specifika uppgifter ger även arbetet ett naturligt flöde, det styr utvecklingen till att istället för att försöka skapa allt på en gång och sedan lappa ihop det bygga fungerande komponenter en för en. För en ensam utvecklare får detta även följden att man automatiskt lägger en uppgift åt sidan när nya uppgifter definieras, dessa kan sedan återbesökas för att granska dem likt en annan utvecklare hade gjort i sammanhanget av ett större team.

Den tydligaste direkt positiva följden av en ensam designer och utvecklare genom samtliga moment är att ingenting förlorats eller förvanskats i kommunikationen mellan de olika faserna. Det är även fördelaktigt att ha en god kännedom om hur själva implementationen kommer att gå till redan under kravgenerering och specificering för att säkerställa att alla krav går att uppfylla samt att inga identifierade krav förutsätter sådant som andra krav kan begränsa.

Det återstår att se hur pass väl systemet i praktiken hanterar storskalig påbyggnad och vidareutveckling. Förhoppningsvis har detta arbete skapat de förutsättningar som krävs för att detta ska ha en stor chans att lyckas.

Litteraturförteckning

Arvola, M., & Johansson, M. (2007). A Case Study of How User Interface Sketches, Scenarios and Computer Prototypes Structure Stakeholder Meetings. *People and Computers XXI – HCI… but not as we know it: Proceedings of HCI 2007.* BCS Learning & Development Limited.

Cadenhead, R. (2013). *Sams Teach Yourself Java in 21 Days (Covering Java 7 and Android).* Indianapolis: Pearson Education, Inc.

Dawande, M., Johar, M., Kumar, S., & Mookerjee, V. S. (2008). A Comparison of Pair Versus Solo Programming Under Different Objectives: An Analytical Approach. *Information Systems Research Vol. 19, No. 1*, 71-92.

Deinum, M., Serneels, K., Yates, C., Ladd, S., & Vanfleteren, C. (2012). *Pro Spring MVC: With Web Flow.* New York: Springer.

Dey, T. (2011). A Comparative Analysis on Modeling and Implementing with MVC Architecture. *International Conference on Web Services Computing (ICWSC)* (ss. 44-49). International Journal of Computer Applications® (IJCA).

Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1995). *Design Patterns: Elements of Reusable Object-Oriented Software.* Indianapolis: Addison-Wesley.

Goodwin, K. (2009). *Designing for the Digital Age: How to create human-centered products and services.* Indianapolis: Wiley Publishing, Inc.

Göransson, B. (2011). *www.rosenfeldmedia.com*. (Rosenfeld Media) Hämtat från Rosenfeld: http://rosenfeldmedia.com/books/survey-design/SUS-svensk.pdf den 08 06 2014

Jongerius, P. (2012). *GET AGILE! Scrum for ux. design & development.* Amsterdam: BIS Publishers.

Mangalaraj, G., Nerur, S., Mahapatra, R., & Price, K. H. (2014). Distributed Cognition in Software Design: An Experimental Investigation of the Role of Design Patterns and Collaboration. *MIS Quarterly, Vol. 38 No. 1*, 249-274.

McConnell, S. (2004). *Code Complete: A practical handbook for software construction, Second Edition.* Redmont: Microsoft Press.

Norman, D. A. (1993). *Things that makes us smart: defending human attributes in the age of the machine.* New York: Basic Books.

Nosek, J. T. (1998). The Case for Collaborative Programming. *Communications of the ACM, 41*(3), 105-108.

Sharp, H., Robinson, H., Segal, J., & Furniss, D. (2006). The Role of Story Cards and the Wall in XP teams: a distributed cognition perspective. *Procedings of AGILE 2006 Conference.* IEEE Computer Society.

Stickdorn, M., & Schneider, J. (2011). *This is service design thinking. Basics - Tools - Cases.* Hoboken: John Wiley & Sons, Inc.

Tidwell, J. (2011). *Designing Interfaces, Second Edition.* Sebastopol, CA: O'Reilly Media, Inc.

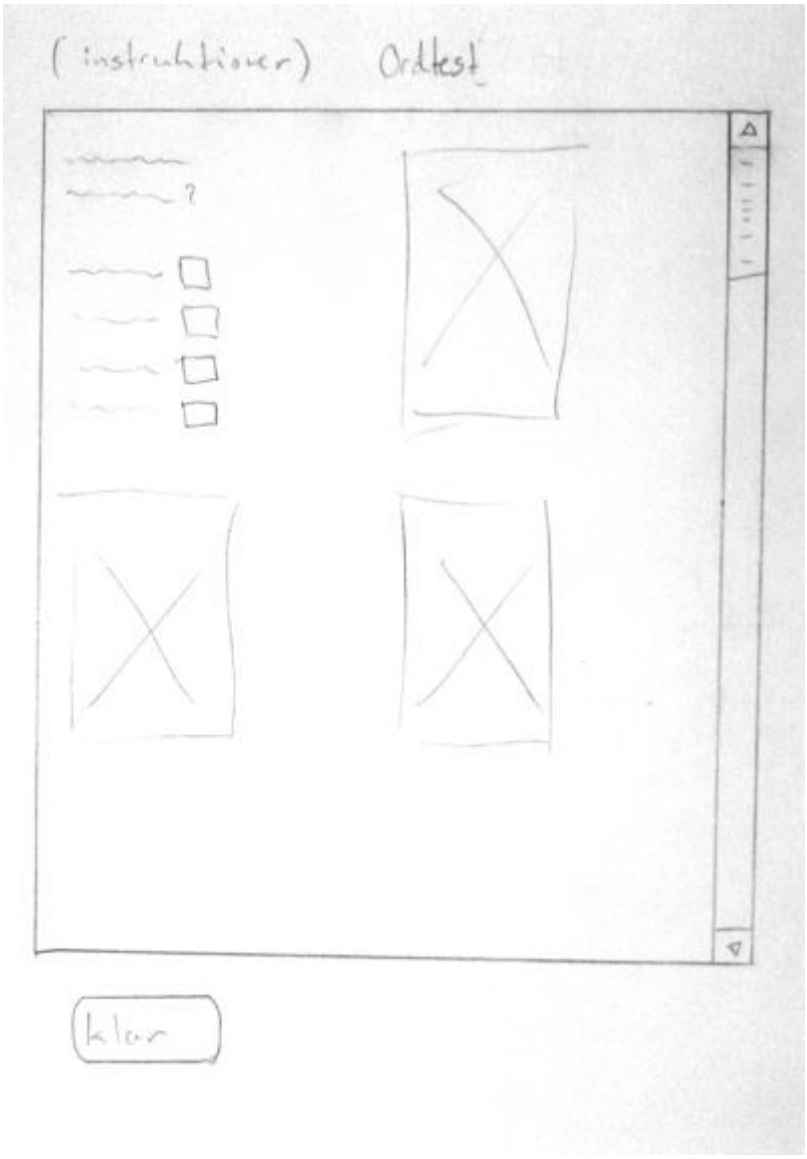
Wilson, G., Aruliah, D., Brown, C. T., Hong, N. P., Davis, M., Guy, R. T., o.a. (2013). Best Practices for Scientific Computing. *arXiv:1210.0530 [cs.MS]*.

Vohra, D. (2008). *JDBC 4.0 and Oracle JDeveloper for J2EE Development: A J2EE Developer's Guide for Using Oracle JDeveloper's Integrated Database Features to Build Data-driven Applications.* Birmingham: Packt Publishing Ltd.

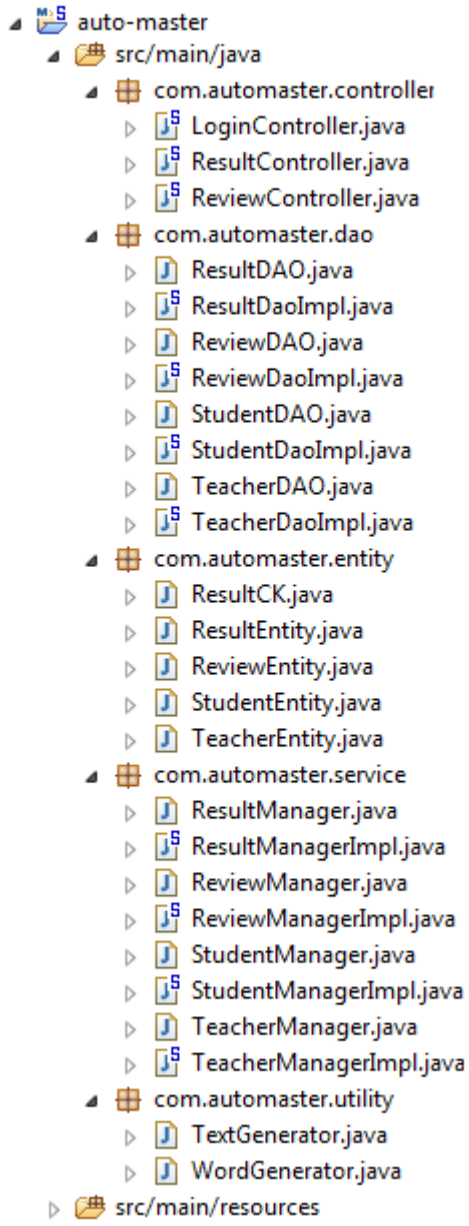
Zhang, C., & Budgen, D. (2012). What Do We Know about the Effectiveness of Software Design Patterns? *IEEE TRANSACTIONS ON SOFTWARE ENGINEERING VOL. 38, NO. 5*, 1213-1231.

Appendix A

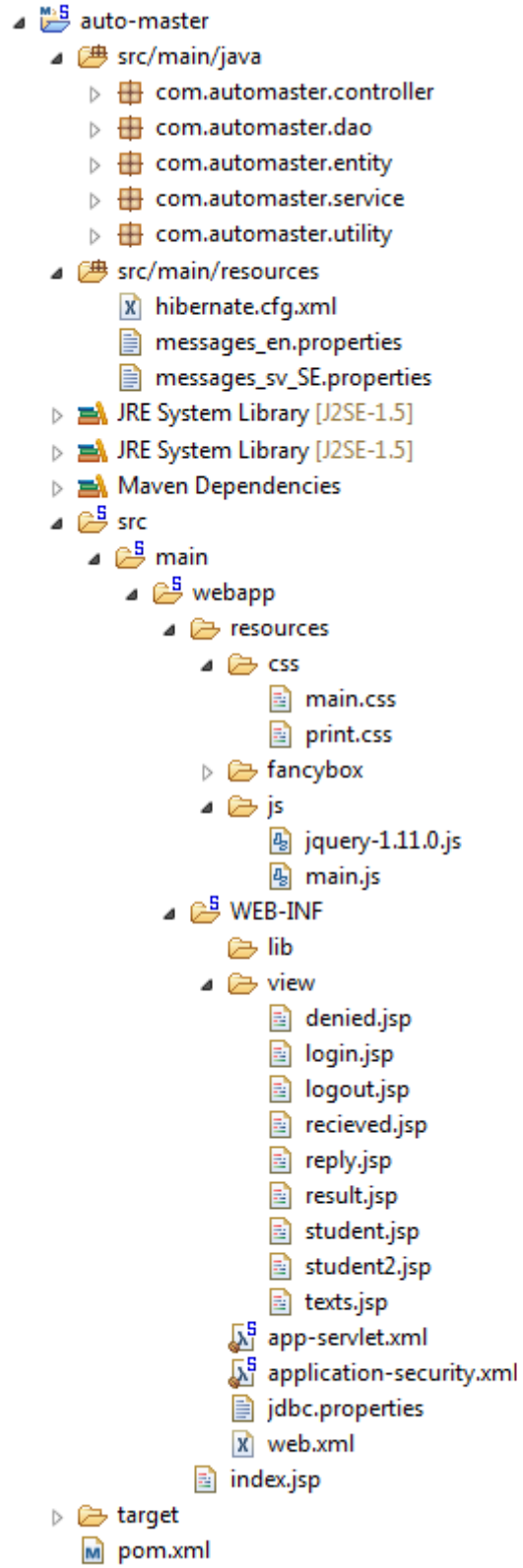
Figurer på pappersprototyp, systemets upplägg samt databastabeller.



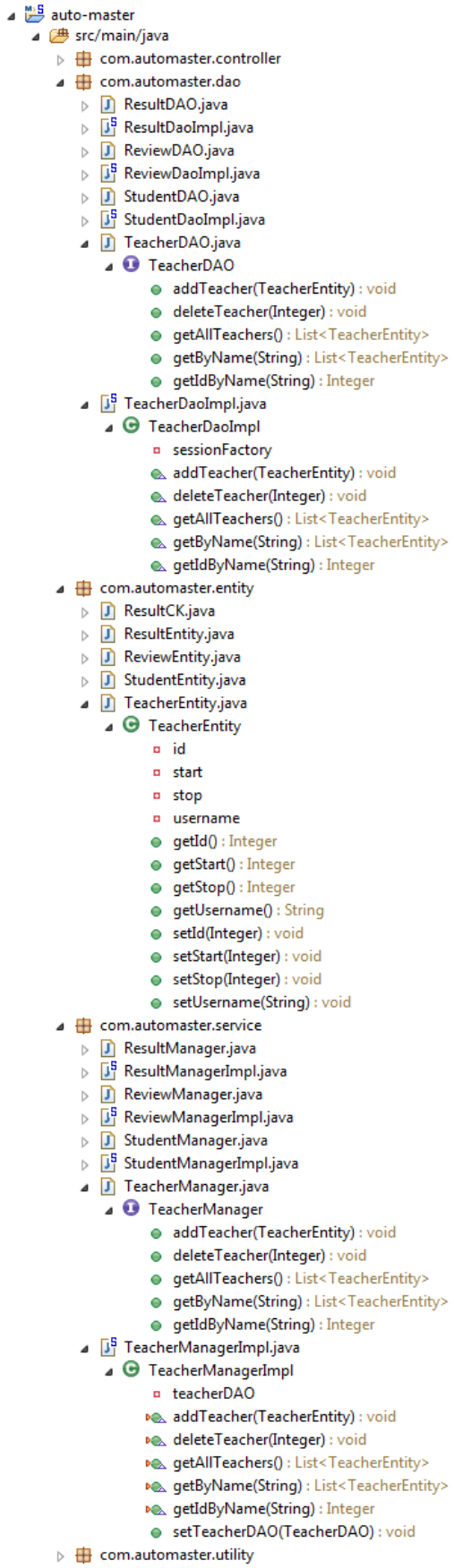
Figur A.1. Exempelvy från pappersprototypen



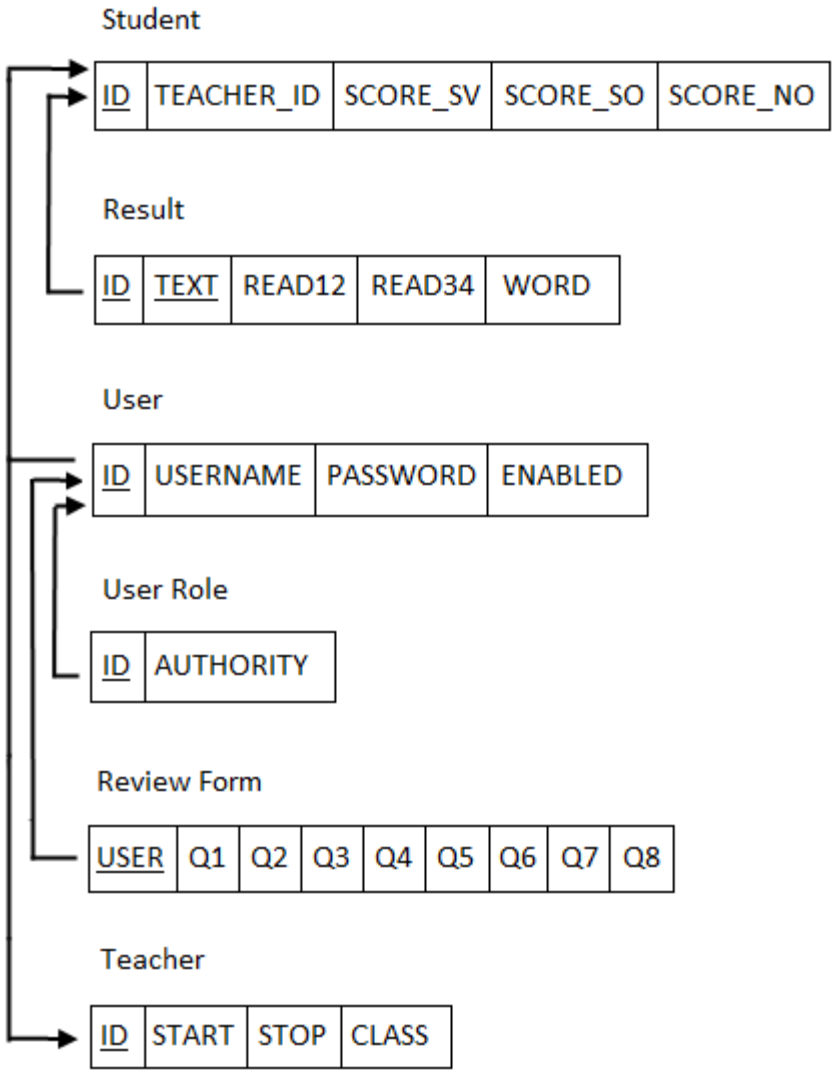
Figur A.2. Översikt över Java-paket och klasser.



Figur A.3. Översikt över vyer, konfigurationsfiler och övriga resurser



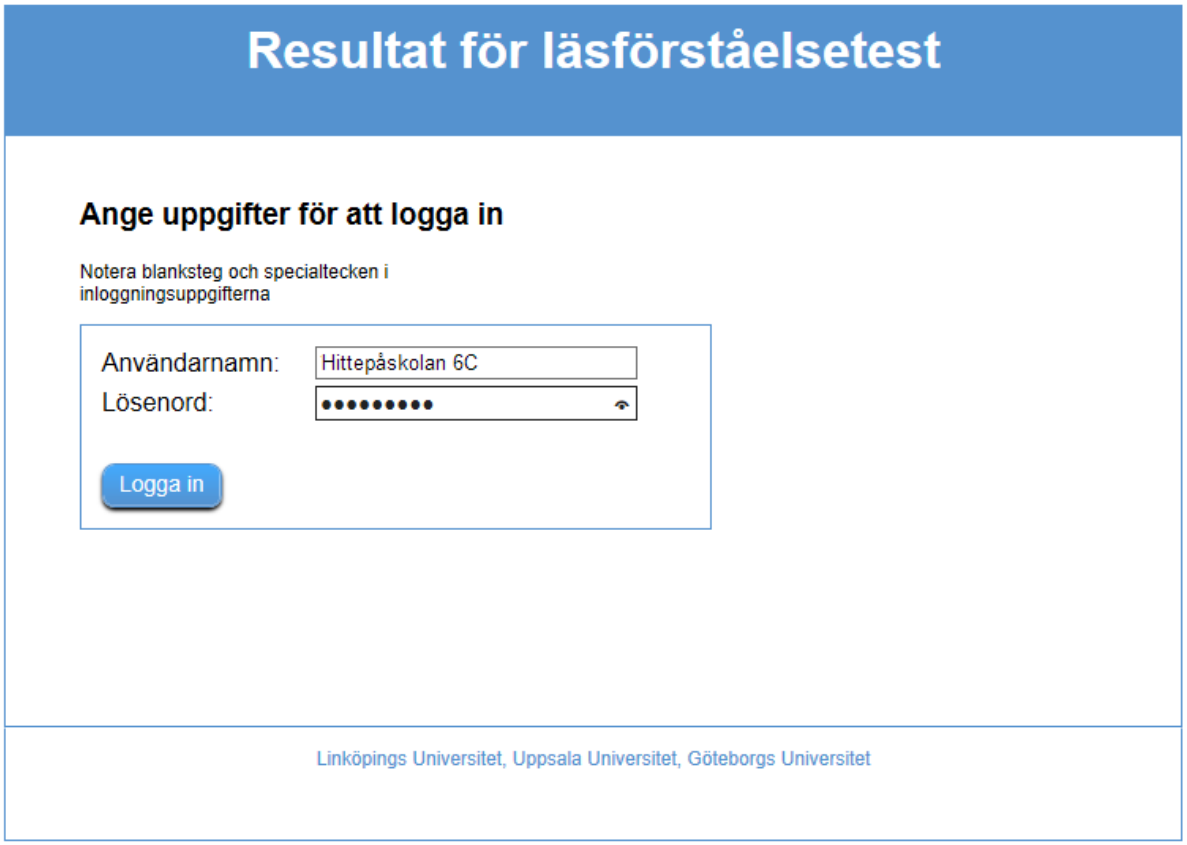
Figur A.4. Funktioner i klasser relaterade till en lärare.



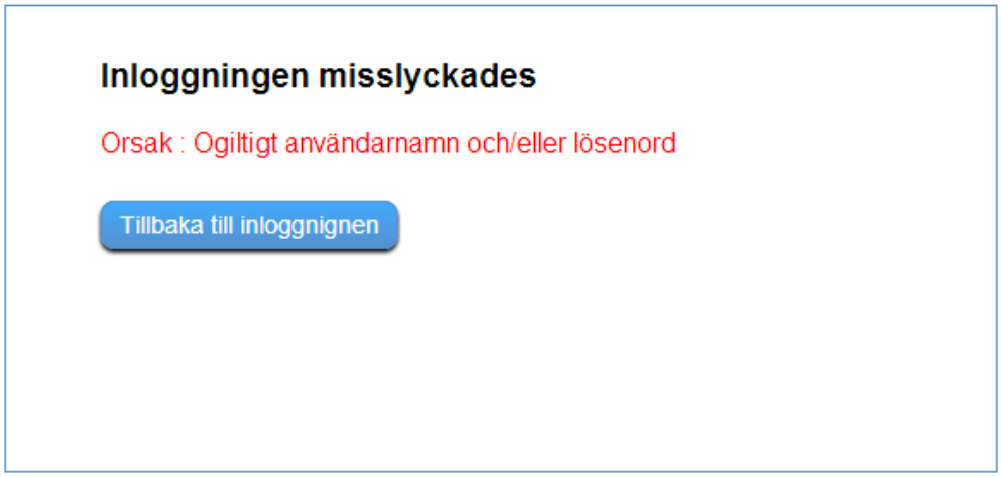
Figur A.4. Schema över databastabeller.

Appendix B

Figurer på de vyer som implementerats, detta är alltså systemet så som det ser ut i användningen för de första användarna.



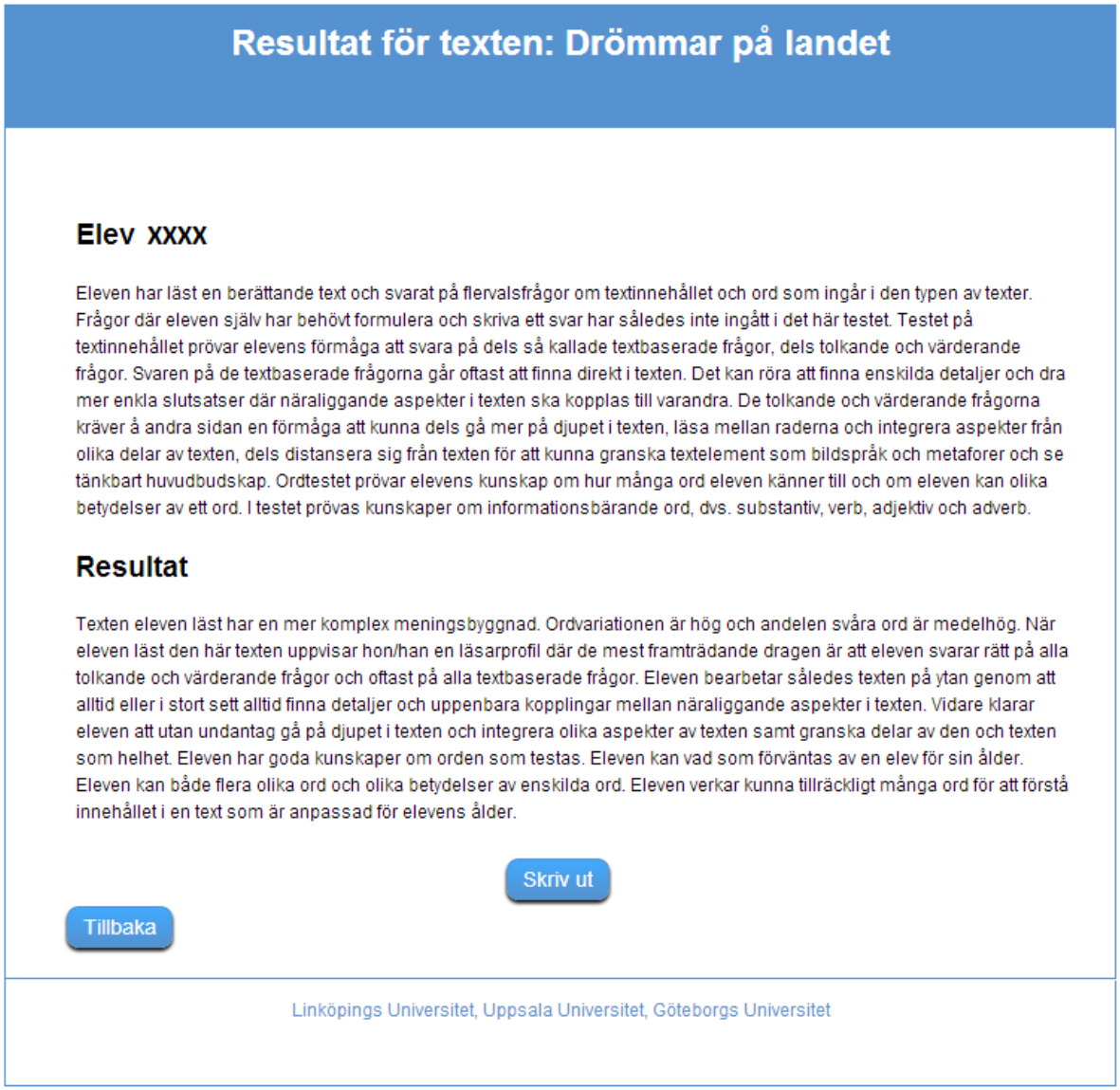
Figur B.1. Startsida med inloggningsformulär.



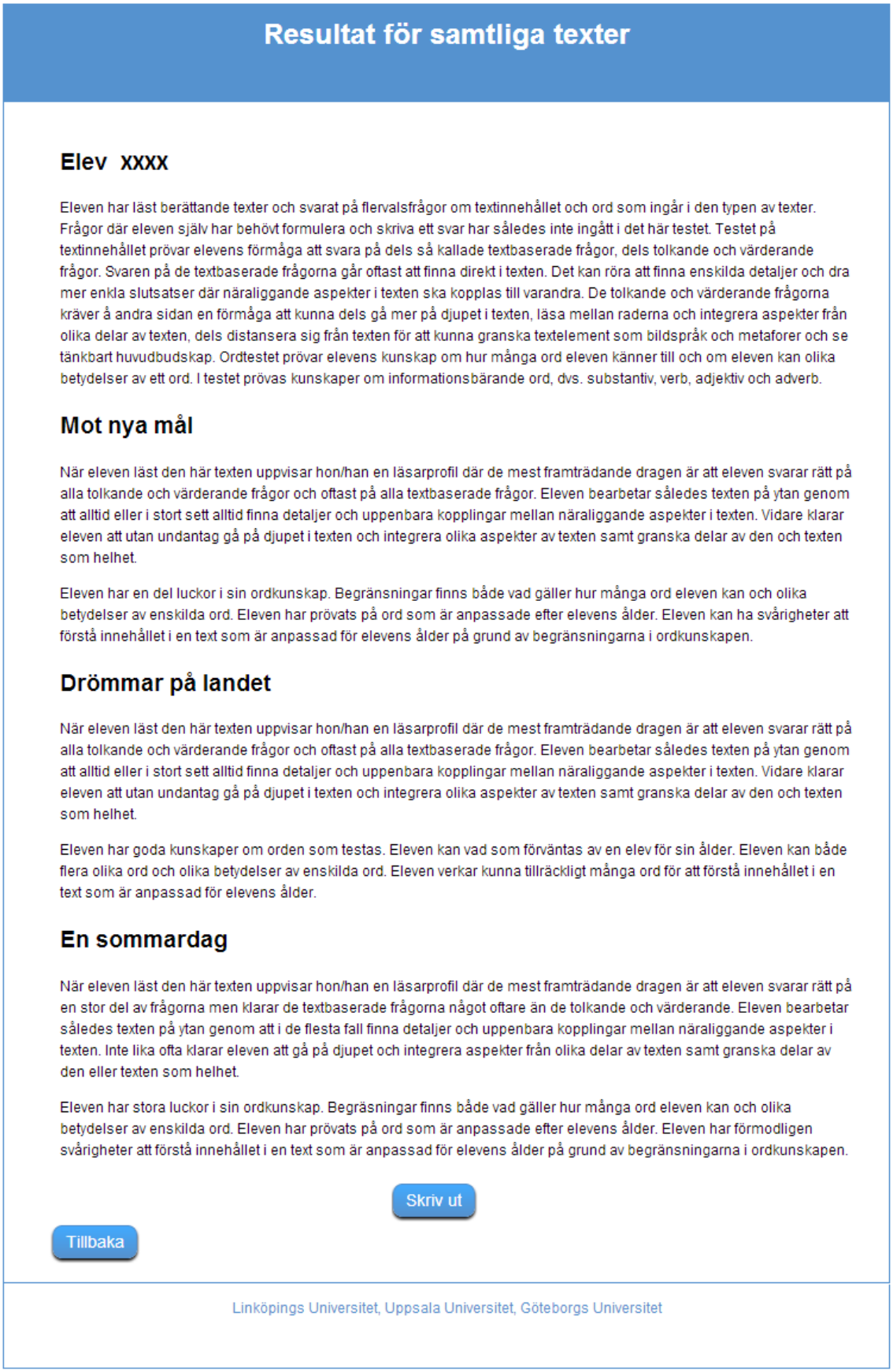
Figur B.2. Sida som visas vid misslyckad inloggning.

Figur B.3. Lista över elever i den inloggade lärarens klass.

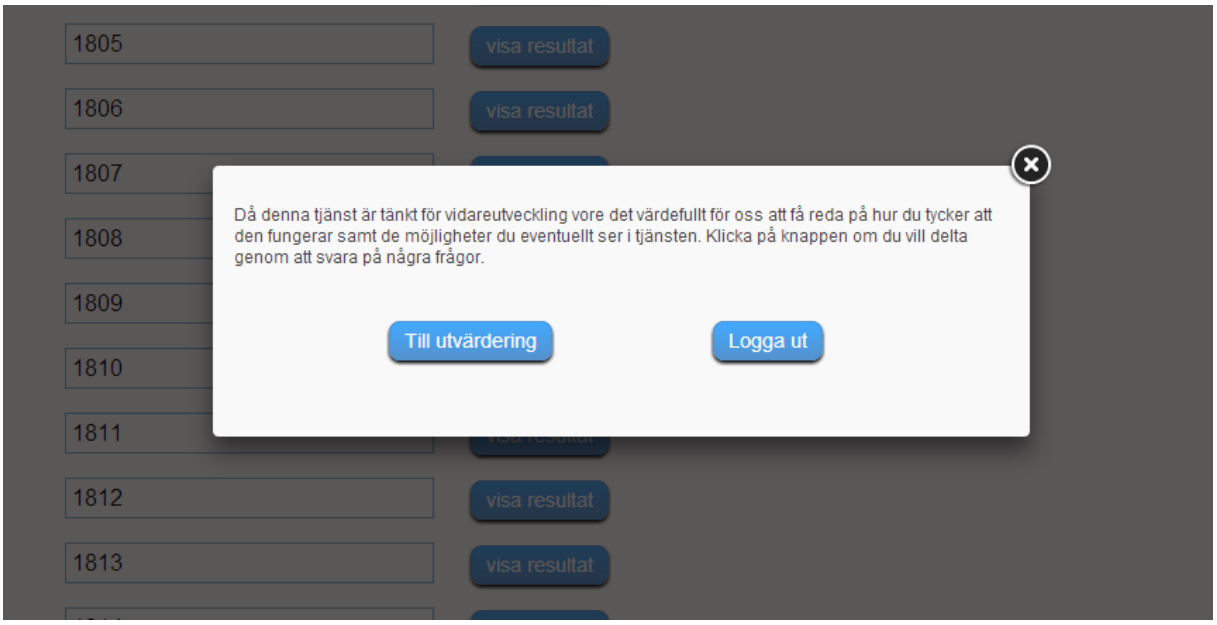
Figur B.4. Lista över de texter eleven har läst och testats för, denna nås genom knapparna ”visa resultat” i föregående figur.



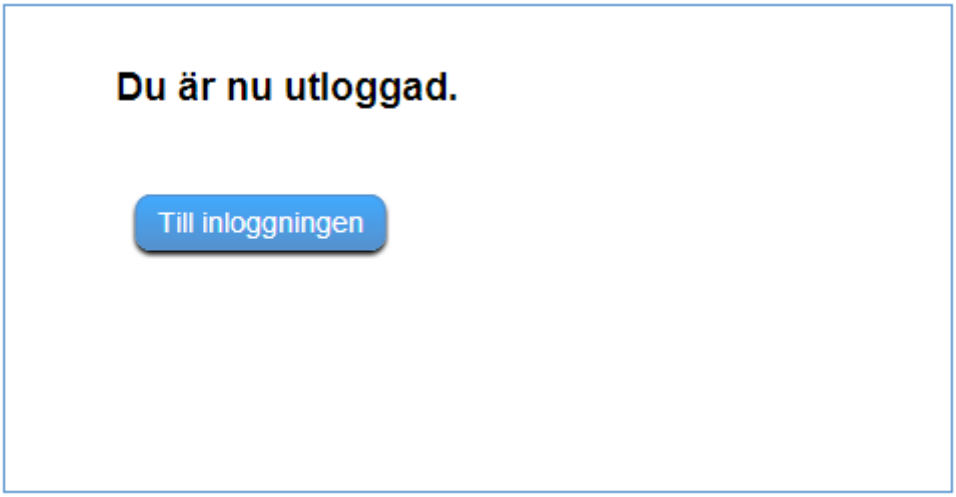
Figur B.5. Genererad resultattext för enskild text, visas då användaren klickar på ”visa resultat” i föregående figur.



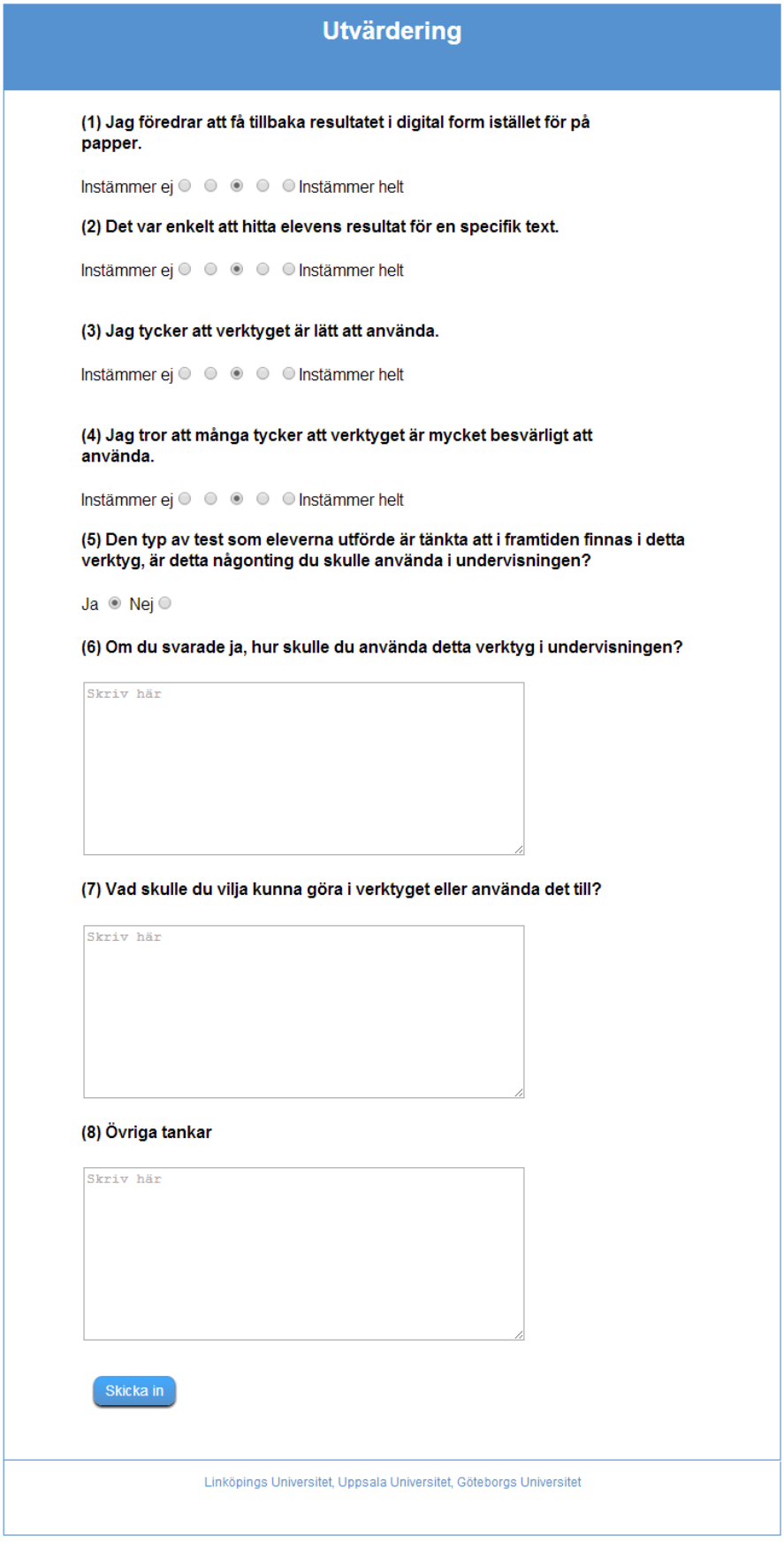
Figur B.6. Genererad resultattext för samtliga texter, visas då användaren klickar på ”visa alla” i föregående figur.



Figur B.7. Dialogruta med uppmaning att utvärdera tjänsten, denna visas vid utloggning.



Figur B.8. Bekräftelse som visas när användaren loggat ut.



Figur B.9. Utvärderingsformulär.