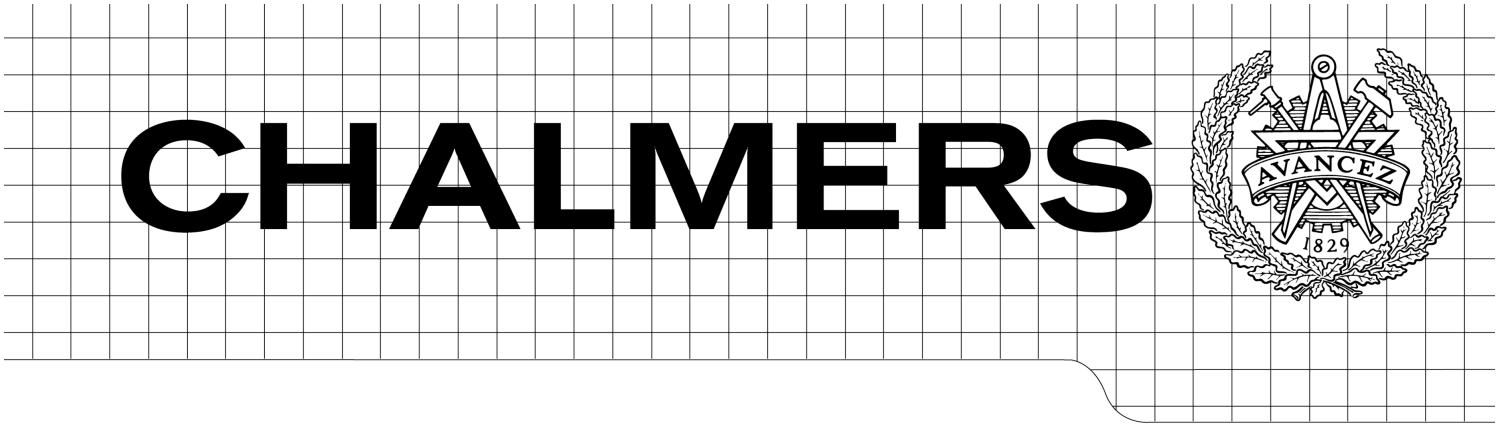


Skill Test

En Webbaserad Platform för Rekryteringstester

Kandidatarbete inom data- och informationsteknik

ANDERS HALLGREN
CHRISTIAN MEIJNER
MARKUS ANDERSSON NORÉN
PHILIP EKMAN
PONTUS DOVERSTAV
SIMON WIDLUND

Chalmers tekniska högskola
Göteborgs universitet
Institutionen för Data- och Informationsteknik
Göteborg, Sverige, Juni 2015

Skill Test
En Webbaserad Platform för Rekryteringstester

Anders Hallgren
Christan Meijner
Markus Andersson Norén
Philip Ekman
Pontus Doverstav
Simon Widlund

Examiner: Jan Skansholm

Chalmers University of Technology
University of Gothenburg
Department of Computer Science and Engineering
SE-412 96 Göteborg
Sweden
Telephone + 46 (0)31-772 1000
Department of Computer Science and Engineering
Göteborg, Sweden June 2015

Förord

Detta kandidatarbete har gjorts vid Institutionen för data- och informationsteknik på Chalmers Tekniska Högskola i Göteborg, av sex studenter från de femåriga civilingenjörsprogrammen i Data- respektive IT-teknik.

Projektet blev till tack vare företaget Software Skills, med ägare Henrik Enström i spetsen. Vi vill tacka Henrik för att han trodde på oss, och för all hjälp under projektets gång. Vi vill också tacka vår handledare, Ramona Enache, för hennes många goda råd och feedback genom hela projektet.

Sammanfattning

IT-branschen är en mycket rörlig och snabbt växande bransch, med ständigt stora behov av nyrekrytering. Den genomsnittliga utbildningsnivån i samhället ökar, men utbildning är inte det enda som gör en kandidat lämplig för ett jobb. Software Skills, ett göteborgsbaserat rekryteringsföretag med inriktning på IT-personal, jobbar med en mer gedigen process, där flertalet intervjuer och även tester ingår.

Syftet med detta projekt är att konstruera en webb-baserad plattform med färdighets- och personlighetstester för Software Skills räkning. Produkten består av två färdighetstester och ett personlighetstest. De förmågor som testas är simultanförmåga och korttidsminne. Personlighetstestet baseras på självskattning, där användaren ska rangordna tre påståenden, sedan används en algoritm som tar hänsyn till hur konsekventa användarens svar är.

För att testa simultanförmåga utför användaren två mindre, enkla uppgifter parallellt, och bedöms enligt en algoritm som ger fördel till den som lyckats fördela uppmärksamheten jämnt över båda uppgifterna.

Minnestestet ger användaren ett rutnät, varpå användaren måste imitera ett mönster av gradvis ökande längd. Utöver ökande längd finns ett distraktionsmoment i form av att mönstret slumpmässigt bryts av med en ruta i fel färg.

Produkten kallas kort och gott Skill Test, och är skapad med hjälp av en samling mycket populära ramverk för webbutveckling kallad MEAN - en akronym för de fyra ingående ramverken Mongo, Express, Angular och Node. All källkod är skriven i Javascript, HTML och CSS.

Gruppen har arbetat med en anpassad variant av en så kallad agil utvecklingsprocess kallad Scrum. Detta innebär att produkten inte har en komplett specifikation från början, utan att gruppen har en kontinuerlig dialog med kunden, som regelbundet får se den ofärdiga produkten och ge synpunkter. Denna process försäkrar att slutresultatet blir enligt kundens vision. Scrum är ett sätt att organisera ett utvecklingsteam där arbetsuppgifter konkretiseras i en produktlogg, och sedan fördelas mellan utvecklarna, som arbetar iterativt i så kallade spurter.

Abstract

The IT sector is volatile and rapidly growing, with a constant need for skilled employees. Although the average level of education in society is increasing, education is not the only thing that makes a candidate fit for a certain job. Gothenburg-based recruitment company Software Skills offers recruitment services for the IT sector. Their recruitment process includes several interviews, but also practical tests.

The purpose of this project is to design a web-based platform with skill- and personality tests to be used by Software Skills in their recruitment process. The product consists of two skill tests and one personality test. The skills tested are multitasking and short-term memory. The personality test is based on self-evaluation, where the user is supposed to grade three statements in the order they apply him/her. An algorithm that takes consistency into account is then used to calculate a result.

To test multitasking capacity, the user performs two simple tasks simultaneously, and is scored with an algorithm that gives an edge to a user that succeeds in giving equal attention to both tasks.

The memory test presents the user with a grid, and the user is tasked with imitating a pattern of gradually increasing length. There is also a distraction in the form of a different color breaking the pattern.

The product as a whole is aptly named Skill Test, and is created with a collection of very popular web-development frameworks called MEAN - an acronym for Mongo, Express, Angular and Node. All source code is written in Javascript, HTML and CSS.

The group has worked according to a slightly modified version of an agile development process called Scrum. This means that the product did not have a complete specification at the beginning in the project. Instead, the group has had regular contact with the customer, who has been able to continuously supply feedback. The Scrum process ensures that the product complies with the vision of the customer. In Scrum, tasks are distributed to the development team from a so-called product backlog. The team then works iteratively in what is commonly referred to as sprints.

Innehåll

1 Inledning

Arbetsgivare ställs ständigt inför utmaningen att anställa rätt person för rätt jobb. Den genomsnittliga utbildningsnivån ökar [1] i takt med att lågkvalificerade jobb i allt större utsträckning ersätts av maskiner [2], alternativt flyttas utomlands. Högre utbildning är dock ingen garant för att någon är rätt för ett visst jobb. Det göteborgsbaserade rekryteringsföretaget Software Skills gör gällande att färdighets- och personlighetstester är mycket användbara i en rekryteringsprocess. Att endast intervjua kandidater är i de flesta fall inte tillräckligt för att hitta rätt person [3]. Tester är synnerligen användbara i processen att försöka skilja på kandidater med snarlik erfarenhet och intervjuprestation. De kan dessutom, i ett inledande skede, användas för att välja ut vilka som bör kallas till intervju. Software Skills arbetar med rekrytering av mjukvaruutvecklare och annan IT-personal. I nuläget använder de sig av intervjuer och programmeringstester, men vill nu expandera verktygsportföljen ytterligare.

Förmågan att snabbt växla mellan olika uppgifter, vanligen kallat simultanförmåga, är något som i dagens samhälle blir allt viktigare, oavsett yrkesgrupp. Med modern teknik finns större möjligheter än någonsin att vara effektiv, men utan simultanförmåga riskerar istället tekniken enligt modern forskning att bli ett hinder för effektivitet [4]. Att kunna hitta personer som har denna förmåga är därför önskvärt, och något som Software Skills vill kunna erbjuda sina kunder.

Något annat som kan öka effektivitet och arbetsprestation är gott minne. God minnesförmåga har bevisade kopplingar till allmän begåvning [5], och kan vara till nytta i alla typer av jobb.

Ett väl utformat personlighetstest kan enligt oberoende akademiska studier förutsäga med upp till 40 procents träffsäkerhet vilket jobb en person får, och kan även till viss del förutsäga bland annat arbetsprestation och ledarskapsförmåga [6]. Det är därför ett eftertraktat verktyg i en rekryteringsprocess.

1.1 Syfte

Projektgruppens uppdrag är att, för Software Skills räkning, skapa en samling tester som kan användas i företagets rekryteringsprocess. Projektet ska mynna ut i en produkt som underlättar för Software Skills att hitta rätt anställda till sina kunder. Från kunden fanns en inledande önskan om en produkt som testar förmågan "att hantera mycket information och fatta snabba beslut".

1.2 Mål

Projektet fick en annorlunda inledning, då gruppen kort innan startdatum nåddes av beskedet att projektet blivit dubblerat på grund av ett administrativt misstag av kursledningen. Software Skills ställde dock upp på att vara kund för ytterligare ett projekt, vilket efter diskussion med examinatorn accepterades. Till gruppens nackdel saknades därför en projektbeskrivning.

Kunden var från början tydlig med ett antal kriterier. Testerna ska ge ett professionellt intryck och fick inte likna dataspel. De skulle vara lätta att komma igång med utan längre instruktionstexter. Testerna skulle vara säkra och konstruerade så att användare med färdigheter inom programmering inte ska kunna fuska. Varje deltest ska dessutom valideras genom väl dokumenterade användartester. Produkten skulle inte vara en prototyp, utan en färdig webbapplikation.

Användargränssnittet skulle vara rent, enkelt och fritt från distraktioner. Typsnitt, grafik och färger skulle vara enhetliga för att skapa ett professionellt intryck.

Avsaknaden av en projektbeskrivning i kombination med att gruppen valde en agil utvecklingsprocess medförde att mål för varje deltest tillkommit efter hand, i samråd med kund och handledare. Målet för slutprodukten blev till slut tre deltester, där varje del är fristående från de övriga, och ska testa olika förmågor eller egenskaper. Dessa tre är simultanförmåga, minne samt personlighetsdrag.

Utvecklingsnamnet för det första testet är "Multi Test". Målet är att testa simultanförmåga, det vill säga förmågan att växla mellan flera uppgifter snabbt.

Det andra testet är ett minnestest. Målet är här att testa korttidsminne genom att lösa en uppgift som gradvis blir svårare, och som är lätt att poängsätta och utvärdera.

Det tredje skiljer sig från de andra, då målet inte är att testa en viss förmåga, utan att utvärdera testpersonens personlighet. Kunden ville kunna, helst på kortare tid än existerande personlighetstester, få en analys av någons personlighet baserad på självskattning. Målet med detta test är att skapa bakomliggande algoritmer och ett ramverk som sedan kan användas med frågor som kunden själv bestämmer.

1.3 Relaterat arbete

Det finns många exempel på liknande tester, inte minst tester som syftar till att utvärdera just personligheten. Ett test som varit populärt under hela 1900-talet är MBTI (Myers-Briggs Typ-Indikator), som mäter personlighet [7].

Försvarsmakten har utvecklat en applikation som ska testa flera olika förmågor så som stresstålighet och minneskapacitet. Denna applikation syftar till att hitta rätt personer att rekrytera, och dessutom uppmuntra de personer som presterar väl att söka till försvaret. Skill Test är av liknande karaktär, men är mer avskalat. Varje test utvärderar en tydligt definerad förmåga, till skillnad från försvarsmaktens test där många förmågor utvärderas samtidigt och användaren inte är medveten om exakt vad som ska testas [8].

Av de minnestester som finns på webben är det speciellt ett som liknar Skill Test - Human Benchmarks [9]. Uppgiften är där att memorera de brickor som blinkar vitt för att sedan kunna klicka på dem. Till skillnad mot det test som utvecklas i detta projekt är att Human Benchmarks test har ett dynamiskt rutnät och ett poängsystem som ger mer poäng ju fler korrekta musklick som görs i rad.

2 Teori

Detta avsnitt presenterar tekniker, verktyg och processer som använts i skapandet av Skill Test.

2.1 Webbapplikationer

Moderna webbapplikationer som exempelvis Gmail och Facebook är de flesta bekanta med. Webbapplikationer har de senaste åren exploderat i popularitet allt eftersom fler insett fördelarna jämfört med traditionella plattformsbundna applikationer. Plattformsoberoende och enkel distribution brukar anses vara några av de största fördelarna [10].

Skillnaden mellan en webbsida och en webbapplikation är inte uppenbar, i synnerhet inte för mindre tekniskt kunniga användare. Många experter är överens om att en webbapplikation har vissa utmärkande drag [11]:

- Självständig - det vill säga oberoende av extern programvara utöver webbläsaren.
- Interaktivt gränssnitt.
- Använder ofta enhetens sensorer eller annan specialhårdvara såsom kamera.
- Använder sig inte av webbläsarens navigationsmöjligheter.
- Går att använda även utan internetanslutning

En webbapplikation uppfyller ofta flera av dessa kriterier, men sällan samtliga.

2.1.1 HTML och CSS

HTML och CSS är två standardiserade tekniker/språk som används vid utveckling av webbsidor och webbapplikationer. HTML definierar strukturen på webbsidan, medan CSS används för att beskriva hur den ska se ut, till exempel genom att bestämma färger, typsnitt och animationer [12].

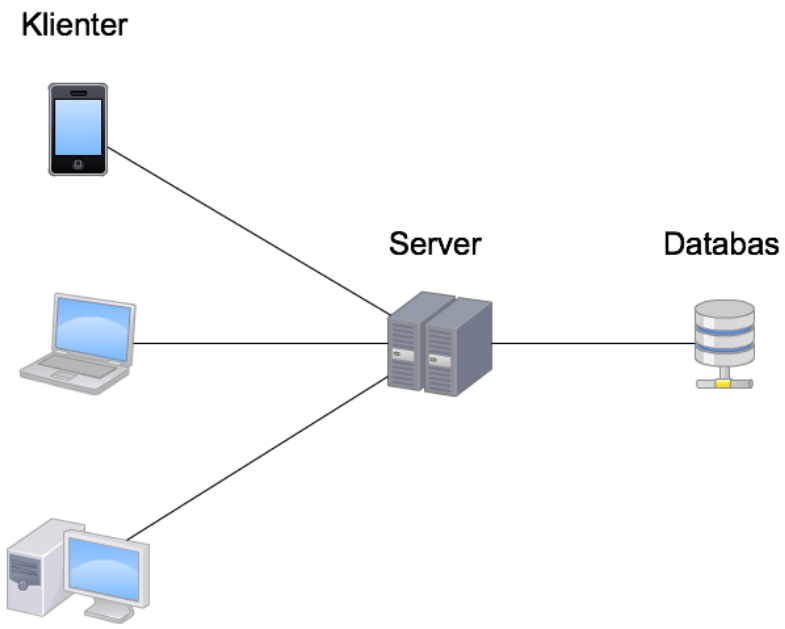
2.1.2 Javascript och JSON

Javascript är ett programmeringsspråk som introducerades 1995 som en del av webbläsaren Netscape Navigator [13]. Det har blivit en standard använd av alla moderna webbläsare för att tillhandahålla dynamik och interaktivitet på webbsidor. Javascript har namnet till trots inget med programmeringsspråket Java att göra.

Javascript Object Notation (JSON) är ett format för datautbyte. Det använder syntaxkonventioner från C-liknande språk och är helt textbaserat, vilket gör det användbart oavsett programmeringsspråk. JSON baseras på två grundläggande datastrukturer [14]: En samling av nyckel/värdepar och en sorterad lista av värden. Det finns väl utvecklade JSON-verktyg till i princip alla moderna programmeringsspråk.

2.2 Klient/server-modell

Många applikationer idag, såväl webb- som mobilapplikationer, är små noder i ett större nätverk bestående av flertalet klienter (se figur 1), där alla är uppkopplade mot en och samma server. Denna arkitektur är fördelaktig då många klienter behöver åtkomst till samma data. I de flesta fall behöver det finnas restriktioner, till exempel då en användare måste kunna identifieras unikt. Klient/server-modellen är i den typen av tillämpningar det bästa alternativet, då servern tillhandahåller gemensam data för hela applikationen som alla klienter kan nå. En nackdel med denna modell är att all data samlas på samma ställe. Detta gör systemet känsligt för angrepp utifrån, till exempel överbelastningsattacker och sårbarheter som innebär läckor av personlig data och/eller lösenord.



Figur 1: Klient/server-modell

2.3 HTTP och sockets

HTTP är ett grundläggande kommunikationsprotokoll på applikationsnivå som i mycket stor utsträckning används för att skicka data på internet. Protokollet har använts på internet sedan 1990 [15].

I vissa typer av webbapplikationer behövs möjligheten att kunna skicka data från server till klient utan att klienten först gör en begäran. Ett tillvägagångssätt för att simulera detta är att låta klienten periodiskt fråga servern om ny data finns tillgänglig. Detta är dock ineffektivt och långsamt, då det kan medföra redundanta förfrågningar och därmed skapa onödig belastning på servern. Som alternativ finns sockets. Denna teknik bygger istället på att en beständig anslutning görs mellan server och klient [16]. Båda kan sedan lyssna efter och skicka meddelanden till varandra utan att någon begäran görs [17].

2.4 REST

Representational State Transfer (REST) är en arkitekturstil som används för att bygga webbtjänster. För att en webbtjänst ska klassas som REST-tjänst behöver den uppfylla sex kriterier [18]:

Klient/server
Klient/server-modellen innebär en uppdelning av ansvarsområden. Genom att ge servern ansvar för datalagring och klienten ansvar för användargränssnitt ökas skalbarheten hos servern och det blir enklare att utveckla klienter för olika platformar.

Tillståndslös
Tillståndslös innebär att all kommunikation mellan server och klient kan förstås utan vidare kontext. All nödvändig information måste alltså skickas i klientens begäran till servern.

Cache
Kriteriet cache innebär att ett svar från servern ska vara märkt som cachebart eller icke cachebart. Om ett svar är cachebart innebär det att klienten kan spara och återanvända det om en identisk begäran görs i framtiden.

Enhetligt gränssnitt
Oavsett vad en webbtjänst gör eller tillhandahåller ska den använda ett enhetligt gränssnitt för att öka generalitet och simplicitet. Dock kan effektivitet förloras då ett standardiserat format används istället för ett som är skräddarsytt för en specifik uppgift.

Hierakiskt system
Hierarkiskt system innebär lager av komponenter som inte kan se bortom det lager det själv ligger i. Detta ger minskad komplexitet samt ökad frikoppling mellan komponenter. Dessa komponenter kan fungera som mellanhänder som skyddar äldre system mot nya klienter och vice versa. De kan dessutom fördela belastningen över systemet.

Kod på begäran
Kod på begäran är ett kriterie som *inte* måste uppfyllas för att en tjänst ska nå REST-standard. Kriteriet innebär att klienters funktionalitet kan utökas genom att exekverbar kod laddas ner från servern.

2.5 MEAN

MEAN är en samling av fyra ramverk som tillsammans används för att utveckla webbapplikationer. Dessa fyra är MongoDB, ExpressJS, AngularJS och Node.js (hädanefter Mongo, Express, Angular och Node). Mongo är en databas, Express står tillsammans med Node för serverlogiken, Angular tillsammans med HTML och CSS sköter det som visas för användaren. Då Express, Node och Angular alla är ramverk som använder Javascript, kan hela applikationen skrivas i samma språk, vilket underlättar utvecklingen [19].

2.5.1 MongoDB

Mongo är en NoSQL-databas, vilket innebär att den inte sparar data i relationer. Istället använder Mongo dokument, som är JSON-liknande objekt. Dessa dokument är rena textsträngar uppbyggda av par av nycklar och värden. Dessa värden kan i sin tur vara andra dokument, listor av värden, rena textsträngar eller tal [20].

Fördelarna med denna dokumentstruktur är bland annat att lagrade värden kan matcha inbyggda datatyper i många programspråk och det dynamiska schemat gör att formen på datan lätt kan ändras. Denna flexibilitet ger i många fall mycket bra prestanda, då databasens data kan matcha applikationens datastrukturer [21].

2.5.2 AngularJS

Angular används för att skriva logik för klienten. Denna blir fristående och skickar HTTP-anrop för att kommunicera med servern. Angular utökar webbläsarens syntax genom att skapa nya HTML-direktiv [22]. Ett direktiv har ett specifikt beteende kopplat till sig. HTML-element binds till ett direktiv genom attribut i koden, och när Angulars HTML-kompilator går igenom koden vet den då vilket beteende som ska kopplas till vilket element [23].

Ett viktigt koncept i Angular är databindningar. Angular användar sig av så kallad två-vägs databindning, vilket innebär att Javascript- och HTML-koden, har delade variabler. Ändringar i den ena uppdaterar den andra, vilket underlättar konstruktionen av applikationen [24].

2.5.3 Node.js

Node används främst för att skriva serverkod till applikationer som behöver ett applikationsprogrammeringsgränssnitt (API) att kommunicera mot. Anrop till klienten eller databasen sker asynkront, vilket innebär att exekveringen aldrig blockeras [25]. Då inga trådar används behövs inga lås, vilket i sin tur betyder att det inte finns någon risk för att processen fastnar i ett dödläge. Det finns heller inget behov av direkt in- eller utmatning i Node, så inte heller där kommer exekveringen blockera.

2.5.4 ExpressJS

Express är ett webbramverk för Node. Det är ett verktyg för utvecklare som vill göra en webbapplikation i Node, då Express utökar Nodes API samt bidrar med nya funktioner [26]. Express används bland annat ofta för att sätta upp funktionalitet på vissa webbadresser *eng. URL*. Att sätta upp funktionalitet på vissa webbadresser är vad som utgör ett REST-API.

2.6 Bootstrap

Bootstrap är ett ramverk som använder HTML, CSS och Javascript för att göra webbsidor responsiva [27]. Detta innebär att layout och/eller design ändras beroende på skärmstorlek på enheten som används. Bootstrap erbjuder dessutom stora mängder färdiga designelement.

2.7 MVC

Model-View-Controller (MVC) är ett designmönster som används för att utveckla applikationer. Modellen beskriver den data som applikationen ska behandla. Vyn visar information för användaren och kontrollern bearbetar användarinteraktion och gör lämpliga anrop till modellen. Fördelarna med MVC är att logiken blir separerad från vyn, så att de kan ändras eller bytas ut utan att man behöver göra ändringar i den andra komponenten [28].

2.8 Versionshantering

Versionshantering innebär användning av ett system som sparar ändringar av filer. Detta är mycket användbart inom programmering, exempelvis då utvecklaren omedvetet introducerat fel i koden och det därför finns ett behov av att gå tillbaka till en tidigare version. Det underlättar även när en grupp samarbetar kring utveckling av mjukvara.

2.8.1 Git

Git är ett kraftfullt versionshanteringssystem gjort för icke-linjär utveckling. Det innebär att alternativa kodgrenar kan skapas och användas utan att påverka den kod som andra i utvecklingsteamet arbetar med [29].

Varje operation i Git görs i första hand på utvecklarens egen dator. Systemet är därför oerhört snabbt jämfört med konkurrerande lösningar där allt synkroniseras mot en server. Alla ändringar kan dock laddas upp till en så kallad centralkatalog närhelst användaren vill. En incheckad ändring är nästintill omöjlig att förlora, eftersom Git generellt sett aldrig tar bort data ur den interna databasen. För att ta bort något måste användaren vara mycket medveten om vad denne gör.

2.8.2 Github

Github är en populär tjänst som tillhandahåller centralkataloger. De erbjuder gratis centralkataloger så länge dessa är publika, och kan erbjuda privata sådana mot en avgift.

Utöver centralkatalog finns verktyg för att spåra problem och buggar i koden. Dessa kan ges etiketter så att det enkelt går att se var felet är, eller av vilken natur problemet eller buggen är. Det finns funktioner för att diskutera och dokumentera, och även för att se statistik för ett projekt [30]. Utöver detta finns inbyggt stöd för att gruppera problem och buggar i milstolpar, vilket gör det enklare att erhålla en gemensam vision för när en milstolpe ses som uppnådd. När de grupperade buggarna och problemen är avslutade ses milstolpen som avklarad.

2.9 Agil systemutveckling

Mjukvaror är ofta komplexa, och kunden är inte alltid medveten om eller kunnig nog att beskriva exakt vad denne behöver. Det kan då bli svårt för utvecklarna att förstå vilka behov kunden har. Till skillnad från många andra ingenjörsområden finns därför ett stort behov för kunden att kunna ändra kraven på produkten under utvecklingsprocessens gång. Inom mjukvaruutveckling är ändringar i kravspecifikationen snarare norm än undantag.

Klassiska ingenjörsmetoder där design och implementation tydligt kan separeras är alltså inte särskilt effektiva inom mjukvaruutveckling. Därför har flertalet agila arbetsmetoder blivit populära. Dessa metoder bygger på iterativ utveckling, vilket innebär att ett antal funktioner implementeras, varpå kunden får lämna önskemål om eventuella ändringar. Kunden blir på så sätt delaktig i utvecklingprocessen, och chansen att den färdiga applikationen uppfyller dennes behov ökar markant [31]. Förändring blir en naturlig del av processen istället för ett orosmoln.

2.10 Scrum

En agil utvecklingsmetod som blivit mycket populär är Scrum [32], en väl dokumenterad och mycket utbredd metod inom agil utveckling. Alla krav på produktens funktionalitet specificeras i början av projektet i en produktlogg. Där ska det finnas uppgifter, krav eller programfixar som ska implementeras. Vad som finns i produktloggen bestäms gemensamt av kund och utvecklare och ska förbättras samt ses över under projektets gång. För att hela tiden ha en lista över aktuella uppgifter som teamet kan ta del av kan så mycket som 10-20 procent av den totala utvecklingstiden läggas på att specificera och konkretisera uppgifter i produktloggen.

Projektet delas in i delar där varje ingående del kallas för spurt. En spurt är en period av projektet där man har ett antal mål som ska uppfyllas. Varje sådan period inleds med en spurt-planering, där gruppen bestämmer vad i produktloggen som ska göras nästkommande spurt.

Tanken är att att det ska utvecklas färdig funktionalitet varje spurt, som sedan visas upp för kunden när spurten är avslutad, under en spurtöversyn. Efter spurtöversynen sker en tillbakablick där gruppen diskuterar hur arbetet har gått och hur det kan förbättras till nästa spurt.

Inom Scrum finns tre olika roller. Det finns en produktägare som är en person i gruppen som har ansvar för produktbackloggen. Denne ska aktivt granska och uppdatera backloggen så att alla i gruppen förstår vad som finns att göra, samt se till att det finns en tydlig prioritet bland de arbetsuppgifterna, så att värdet på det som produceras maximeras.

Det ska även finnas en Scrum-mästare som är en person i gruppen vars uppgift är att se till att Scrum används på rätt sätt, samt att avlägsna hinder så att gruppen kan ägna sig åt utveckling i så stor mån som möjligt.

Den tredje rollen som finns i Scrum är utvecklare. Utvecklingsgruppen är en självorganiserande grupp som har som uppgift att realisera punkterna i produktbackloggen till "färdig" mjukvara, där "färdig" är en tydligt fördefinierad gräns bestämd gemensamt av kunden och projektgruppen.

2.11 Användartester

Under utvecklingen av en applikation, är ofta de personer som testar mjukvaran gruppmedlemmarna själva. Gruppmedlemmarna är utvecklare, ofta duktiga på att hantera mjukvara, och är oftast inte representativa för slutanvändaren. Detta ger därför ingen indikation på hur lätt applikationen är att förstå, och det kan vara svårt att hitta vissa typer av tekniska problem om testaren använder mjukvaran på ett förutsägbart sätt.

Användartester eller användarexperiment [33] är ett vanligt sätt att utvärdera teknisk kvalitet och användarvänlighet hos mjukvara. Dessa kan bestå av att personer som ger en bra representation av målgruppen för applikationen använder mjukvaran i en naturlig miljö. I början av ett projekt kan det vara svårt att dra några slutsatser om själva applikationens funktionalitet, då ofta tekniska problem eller knapphändiga användargränssnitt kan stå i vägen för att användaren ska kunna hantera applikationen på ett naturligt sätt, då kan mindre tester vara bra för att tidigt upptäcka sådana problem.

Senare i processen kan mer utförliga tester göras för att ytterligare utvärdera applikationens funktionalitet och användarvänlighet. Det som ofta vill undersökas är hur en viss variabel påverkar mjukvaran. En variabel kan vara en knapps position eller färg, men också mer komplexa saker, så som olika algoritmer. Ett vanligt sätt att mäta hur en variabel påverkar detta är genom så kallad A/B-testning. I A/B-tester ges olika versioner av en applikation till två testgrupper, sedan observeras eventuella skillnader i resultat eller användning hos de två testgrupperna [34].

2.12 Elo-rating

För att mäta personlighet behövs ett tillvägagångsätt för att utvärdera och ställa olika personlighetsattribut mot varandra. I praktiken är det dessutom testdeltagarens egen uppfattning om sig själv som utvärderas. Denne väger olika attribut, representerade av påståenden, mot varandra. Det som görs är alltså en relativ skattning av attributen.

För att tekniskt kunna mäta denna skattning behövs en algoritm som tar hänsyn till de olika attributens relativa styrkor. En välbeprövad metod för att åstadkomma en sådan bedömning är så kallad Elo-rating [35]. Hur denna metod har använts beskrivs i detalj i avsnitt 5.3.

2.12.1 Metoden

Elo-rating, framtagen av fysikprofessorn Arpad Elo, användes ursprungligen för att rangordna shackspelare men används idag dessutom inom många lagsporter, för att matcha spelare inom tv/data-spel och för rankning inom e-sport.

Matchdeltagarnas rating används för att förutspå resultatet i en match. Spelare med samma rating kommer att ha samma förväntade resultat. När skillnaden i rating mellan spelare ökar, ökar även skillnaden mellan deras förväntade resultat. Tillsammans med det faktiska resultatet används sedan det förväntade resultatet till att beräkna spelarnas nya rating.

Syftet med detta system är att skapa en mer rättvis rating som inte är direkt baserad på antalet vinster och förluster. Istället tar den hänsyn till motståndarens rating i en match. Om exempelvis en spelare med hög rating vinner över en spelare med låg rating har det en liten inverkan på deras rating, då endast ett fåtal poäng kommer att tas från spelaren med låg rating. I fallet där spelaren med låg rating vinner blir utfallet raka motsatsen. Det vill säga att en större mängd poäng kommer att överföras från spelaren med hög rating med resultatet att båda spelares rating påverkas mer. Detta innebär att systemet blir självbalanserande. Med tiden kommer en spelares rating justeras så att den motsvarar spelarens faktiska skicklighetnivå.

2.12.2 Bakomliggande matematik

Skalan på ratingen R kan variera beroende på implementation. United states chess federation använder sig av en skala som sträcker sig från 100 och uppåtт. Nybörjare startar med $R = 800$.

Resultatet S i en match kan antingen vara 1, 0.5 eller 0. Vinst ger 1 poäng, oavgjort ger 0.5 poäng och en förlust ger 0 poäng.

Om en spelare A har en rating R_A och en spelare B har en rating R_B kan deras förväntade resultat E beräknas enligt följande formel.

$$E_A = \frac{1}{1 + 10^{(R_B - R_A)/400}} \tag{1}$$

$$E_B = \frac{1}{1 + 10^{(R_A - R_B)/400}} \tag{2}$$

Efter en match beräknas en spelares nya rating R_N baserat på det förväntade resultatet E_A, det faktiska resultatet S_A och dess gamla rating R_A. Följande formel används för beräkningen:

$$R_N = R_A + K(S_A - E_A) \tag{3}$$

K i ekvation 3 ovan är det maximala antal poäng som kan överföras mellan spelare efter en match. Värdet på K är inte skrivet i sten och kan justeras beroende på hur man vill balansera ratingen. I den ursprungliga algoritmen är $K = 32$ för spelare med låg rating och $K = 16$ för spelare med hög rating.

3 Metod

Alla typer av projekt behöver en fast struktur. Ju större grupp, desto högre krav på organisation. Genom att kombinera delar av en väl etablerad metod för mjukvaruutveckling med egna metoder och handledning har gruppen utvecklat ett sätt att arbeta som beskrivs närmare i denna del.

3.1 Intern kommunikation

För kommunikation inom gruppen har framför allt Slack och Google Drive använts.

Slack är en tjänst skapad specifikt för gruppkommunikation. Möjligheten att skapa separata kanaler, dela filer och ha privata konversationer gör Slack till ett bra verktyg för denna typ av projekt.

Google Drive har använts till att lagra dokument såsom tidsloggar, dagböcker och mötesprotokoll. Det användes även för att samarbeta med skrivandet av planeringsrapporten.

3.2 Scrum

Eftersom samtliga gruppmedlemmar har haft andra kurser parallellt med kandidatarbetet valde gruppen att endast anamma valda delar av Scrum. En lätt modifierad version av Scrum har tillämpats, som inneburit ett större veckomöte istället för ett litet Scrum-möte varje dag. Bortsett från detta har Scrum använts som beskrivet i avsnitt 2.10.

3.3 Roller

På examinatorns och handledarens inrådan gavs varje gruppmedlem ett specifikt ansvarsområde. Nedan följer en kort beskrivning av dessa roller.

Produktägare
Produktägarrollen är tagen från SCRUM, och dennes huvudsakliga uppgift

var att se till att visionen för produkten följs samt att se till att produktkatalogen hålls uppdaterad. Produktägaren agerar också länk mellan gruppen och kunden.

SCRUM-mästare
SCRUM-mästarens huvudsakliga uppgift var att se över de administrativa delarna av varje spurt och styra över vad som ska göras, bedöma vad som kommer hinnas med, och att prioritera uppgifterna. Ansvarar även för produktloggen.

Git-ansvarig
Den Git-ansvariges uppgift var att skaffa sig en mer ingående kunskap om Git, då detta var ett område som det saknades spetskompetens inom, men som är mycket viktigt för att utvecklingen ska gå smidigt.

Dokumentationsansvarig
Den dokumentationsansvarige hade huvudansvar för dokumentation av möten, samt övergripande ansvar för innehåll, struktur, stil och språk i planerings- och slutrapporterna.

Teknikchef
Teknikchefens uppgift var att vara påläst om de språk och ramverk som används, och att efter bästa förmåga hjälpa resten av gruppen med tekniska problem.

Projektledare
Projektledarens uppgift var att ha koll på deadlines samt ha en övergripande koll på gruppens arbete och projektets status. Gruppens trivsel var också projektledarens ansvarsområde.

3.4 Kundmöten

Möten med Henrik Enström från Software Skills har ägt rum på deras kontor ungefär varannan vecka - detta då två veckor har varit den normala längden på en spurt. På dessa möten har gruppen visat vad som gjorts under den senaste spurten samt diskuterat vad som ska göras under kommande spurt. Det har även varit under dessa möten som idéer till nya tester har tagit form. På så sätt har den inledningsvis vaga idén kunnat mejslas ut till en allt tydligare sådan. Dessa möten har alltså fungerat som ett tillfälle där kunden fått möjlighet att dels reflektera över produkten i dess nuvarande stadie, och dels givits möjlighet att säga sitt om vad som ska finnas med i den kommande spurten.

3.5 Versionshantering

Möjligheten att arbeta i separata grenar kan effektivisera och underlätta utvecklingen. Backup av koden blir också simpel, då både lokal och molnbaserad backup blir en naturlig del i arbetsflödet. Versionshantering är i stort sett ett krav i alla större projekt då bland annat delning och backup av kod är nödvändigt och har på så sätt varit avgörande för gruppens prestation.

Git och Github har gett möjligheten att arbeta i separata grenar, vilket har effektiviserat och underlättat utveckligen. De underlättar även delning av kod vilket är viktigt då flera utvecklare arbetar på samma projekt. Backup av koden blir också simpel, då både lokal och molnbaserad backup blir en naturlig del i arbetsflödet.

Gruppen har använt två huvudgrenar, kallade "Master" och "Develop". Dessa grenar ska finnas kvar under hela projektet. Kod i Master ska alltid vara i ett stadie där den är redo att köras i produktion, alltså så stabil som möjligt sett till projektets status. Develop representerar den senaste färdiga ändringen i koden som någon gång i framtiden ska sammanfogas med koden i Master.

Utöver dessa två huvudgrenar användes stödgrenar med prefixet "Feature". Dessa används för att utveckla nya funktioner som sedan sammanfogas med Develop, varpå de tas bort. Feature-grenar får endast grenas av från Develop och sammanfogas likaså endast med Develop[36].

3.6 Användartester

Efter två spurter var produkten i ett skick där enklare användartester kunde utföras för att upptäcka tekniska problem och se hur enkel applikationen var att använda. Så tidigt i utvecklingen upptäcks ofta problem med produkten snabbt, därför behövde inte användartesterna ha några speciella krav gällande omfattning och målgrupp, utan enklare tester utfördes på kurskamrater.

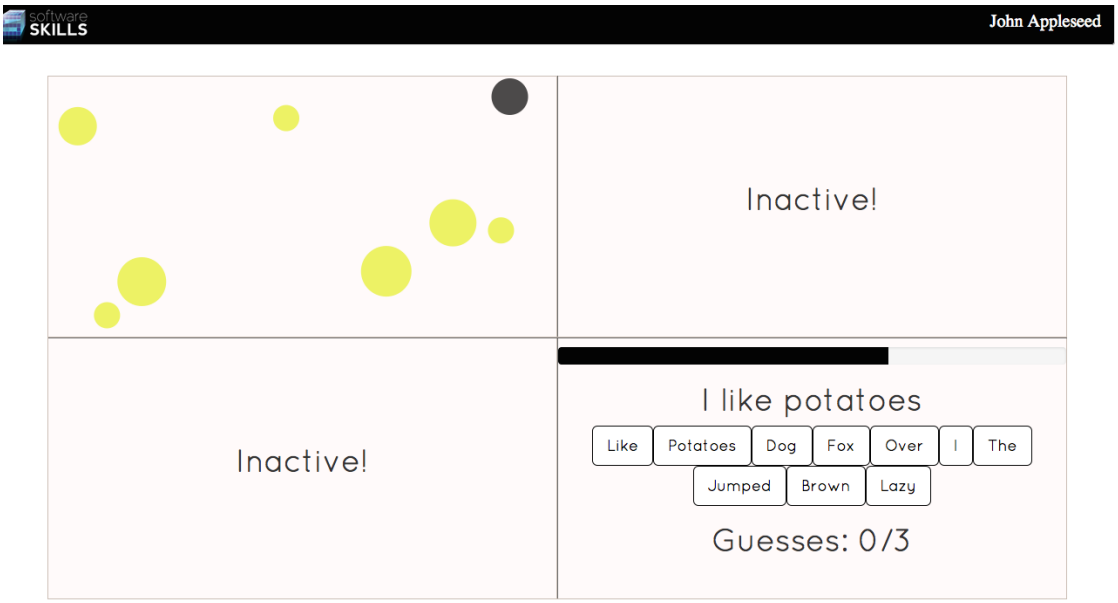
4 Resultat

Produkten som skapats kallas "Skill Test", och består av tre delar. Två av dem fokuserar på simultanförmåga respektive korttidsminne, och en testar personlighet. Till detta hör en kontrollpanel för administratören, för att denne ska kunna ändra och lägga till data i testerna. Såväl arkitektur och tekniska lösningar som hur produkten används beskrivs i detta avsnitt. I början av avsnittet förklaras det hur testerna fungerar för att sedan handla om hur systemet är implementerat.

4.1 Multi Test

Färdighetstestet Multi Test sätter användarens simultanförmånga på prov genom att under en eller flera rundor låta användaren utföra två små, enkla uppgifter samtidigt under tidspress. Uppgifterna är antingen aktiva eller passiva, och det finns två av varje typ. En runda består av ett passivt och två aktiva test, där det passiva pågår dubbelt så länge som de aktiva testen. Ett passivt test behöver användaren bara hålla under uppsikt, och kräver minimal interaktion, medan de aktiva kräver fler inmatningar, detta för att användaren inte ska få orimliga krav på sig att behöva klicka på flera ställen samtidigt. Användaren behöver alltså ha uppmärksamheten riktad på två tester samtidigt, men det är bara ett av dem som kräver mer omfattande interaktion. Administratören kan genom en kontrollpanel välja hur länge varje uppgift är aktiv, och även antal rundor, uppgifternas ordning slumpas fram tills sista rundan är färdig.

Användargränssnittet syns i figur 2 nedan.

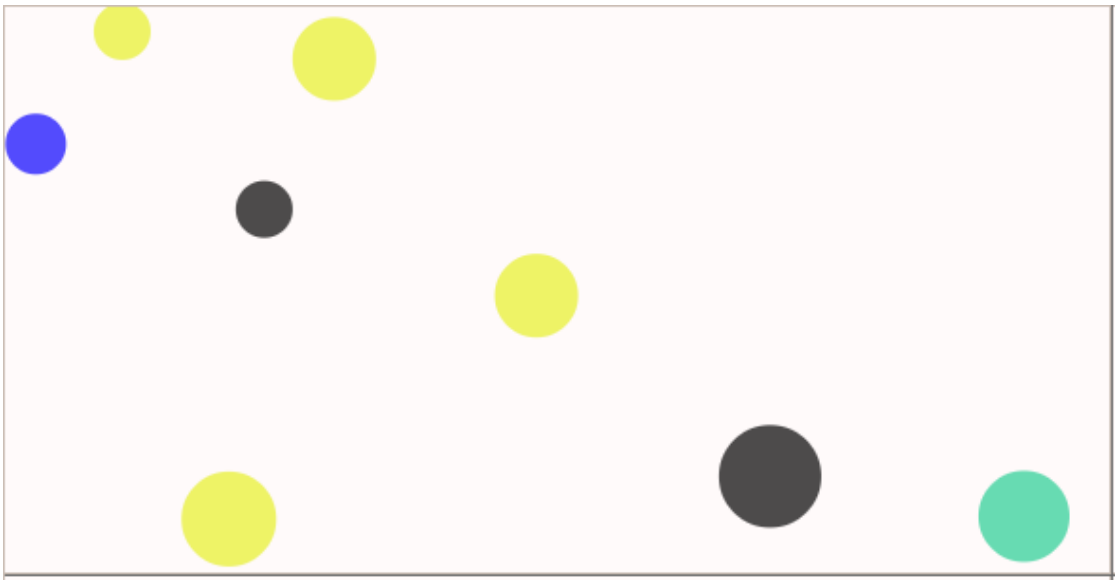


Figur 2: Skärmdump från Multi Test

4.1.1 Balls

Detta är en av de två passiva uppgifterna. Uppgiften är att hålla koll på hur många av de studsande bollarna som har en viss färg. De byter färg slumpmässigt, och utan förvarning försvinner alla bollar och användaren ska då komma ihåg hur många som hade den efterfrågade färgen ögonblicket innan de försvann. För att undvika att färgen ändras precis i ögonblicket de tas bort slutar bollarna byta färg två sekunder innan sluttid. Varje rätt svar ger 3 poäng. Vid rätt svar på alla tre försök i en omgång erhålls en bonuspoäng.

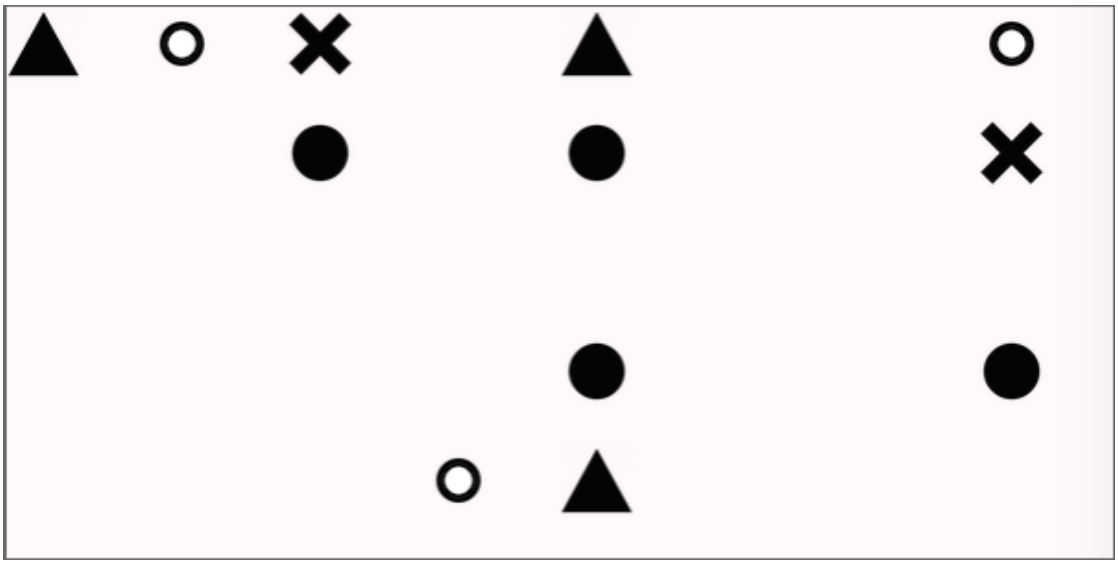
Figur 3 visar hur Balls ser ut.



Figur 3: Skärmdump från Balls-testet

4.1.2 Shapes

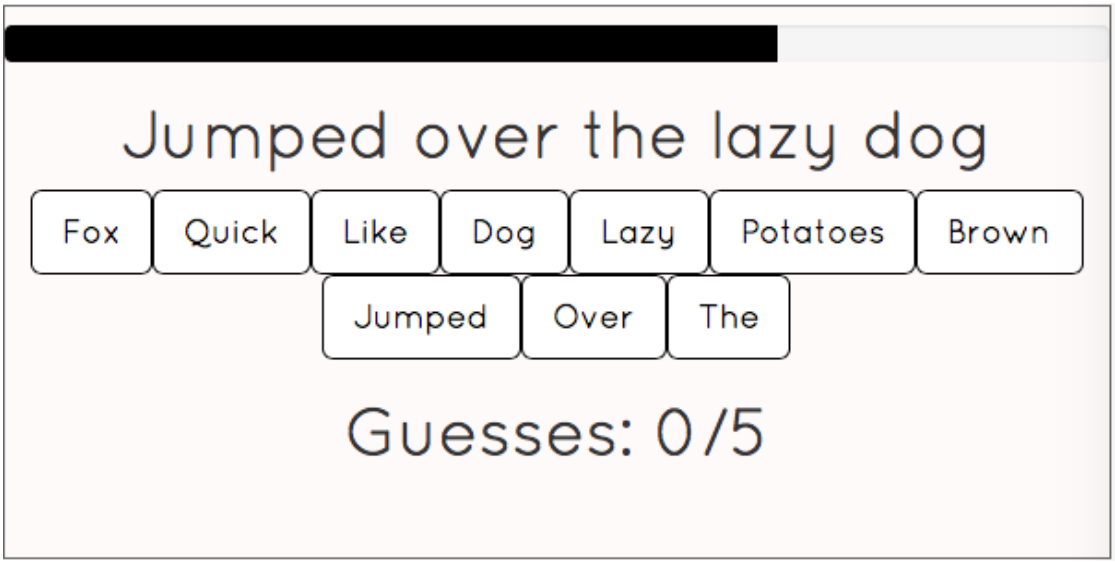
Ett passivt test där man ser ett antal geometriska former placerade på skärmen, med ett godtyckligt antal figurer per geometrisk form, se figur 4.1.2. Dessa försvinner sedan successivt. Användarens uppgift är identifiera den första figuren som blir ensam av sitt slag. Poängen baseras på hur lång tid det tar för användaren, från att den första formen blir unik till att ett svar ges.



Figur 4: Skärmdump från Shapes-testet

4.1.3 Words

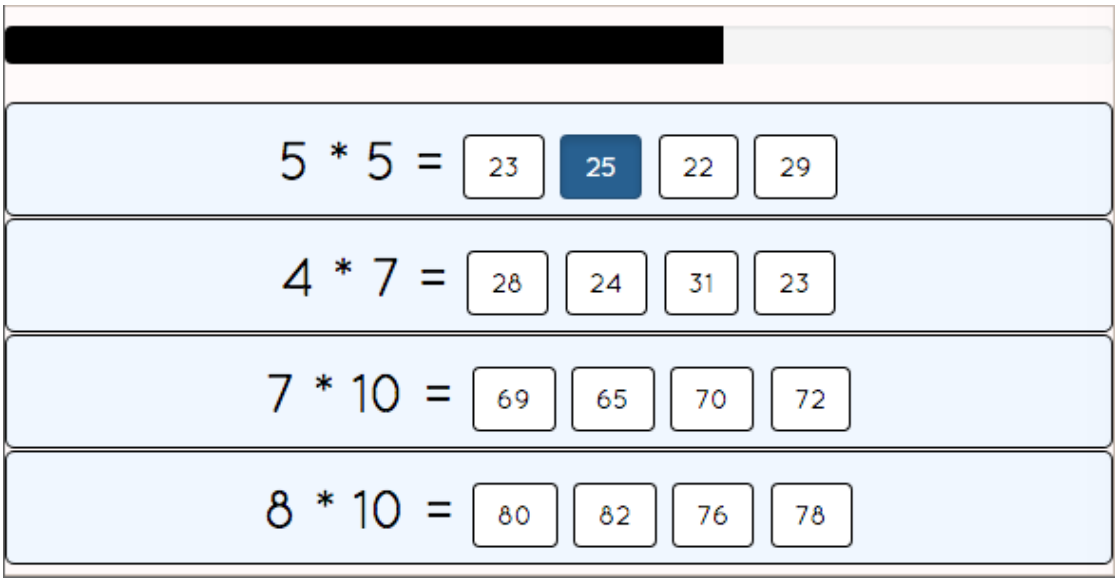
En aktiv uppgift som innebär att användaren så snabbt som möjligt ska klicka på ett antal ord i rätt följd så att de bildar en korrekt mening. Meningarna läggs in manuellt av administratören, och de färdigskrivna meningarna slumpas sedan fram av servern. De skickas till klienten tillsammans med en lista av ord. Dessa ord är slumpvis valda och visas i slumpmässig ordning, se figur 5. Användarens svar skickas till servern för validering. Ett poäng ges för varje korrekt valt ord, inga avdrag görs för felaktiga val.



Figur 5: Skärmdump från Words-testet

4.1.4 Math

En aktiv uppgift där användaren ska lösa ett antal matematikuppgifter. Uppgifterna slumpas fram av servern och skickas till klienten med fyra svarsalternativ, se figur 6. De tre alternativ utöver det korrekta är slumpade med ett intervall av fem runt det korrekta för att de ska anses rimliga. När tiden är slut skickar klienten användarens svar till servern för validering. För varje rätt svar ges ett poäng.



Figur 6: Skärmdump från Math-testet

4.1.5 Testomgång

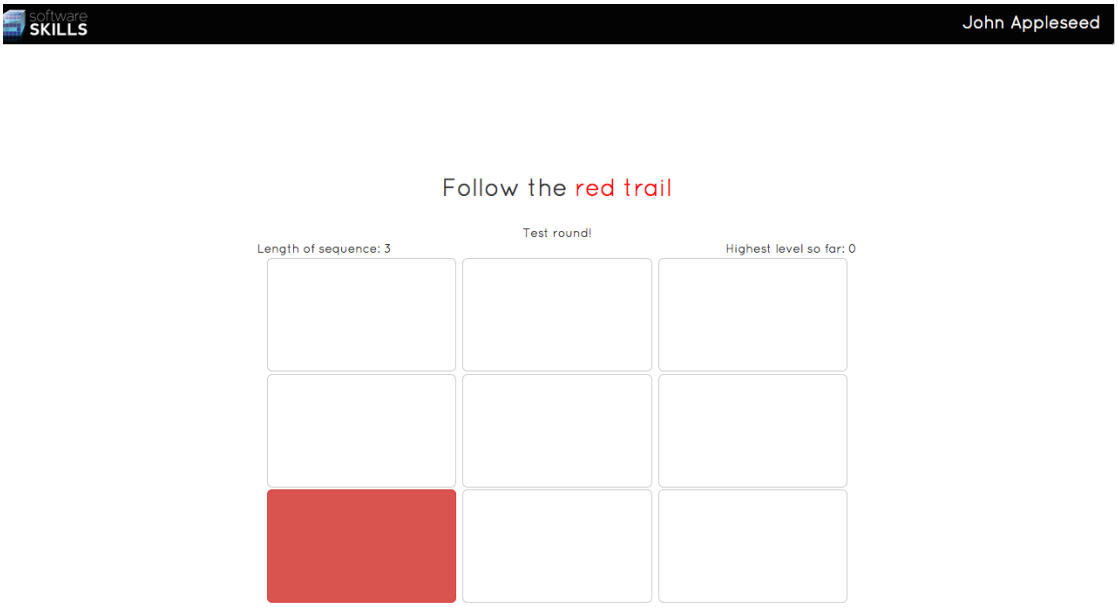
Då testet syftar till att testa förmågan att göra flera saker samtidigt är det viktigt att användaren först har full koll på hur de individuella uppgifterna fungerar. Därför finns möjligheten att få instruktioner samt att prova samtliga uppgifter individuellt innan själva Multi Test börjar. Dessa kan provas upprepande gånger tills användaren känner sig bekväm med hur uppgifterna genomförs. På så sätt kan det, förutsatt att användaren följer instruktionerna, garanteras att det är förmågan att lösa de olika uppgifterna samtidigt som testas i själva Multi Test.

4.2 Memory Test

Fokus för detta test är att utvärdera användarens korttidsminne genom att låta användaren iakta en sekvens av rutor som lyses upp i en bestämd ordning. Användarens uppgift är att följa denna sekvens med hjälp av musklick. Om användaren klarar av att upprepa sekvensen kommer det en ny, längre sekvens i nästa runda. Emellanåt kommer störande moment visas för användaren i form av rutor som lyses upp i en annan färg än den som ska följas. Detta för att testa hur väl användaren filtrerar ut den information som är relevant. Resultatet baseras på den längsta sekvens användaren klarade av att upprepa.

4.2.1 Gränssnitt

Det grafiska gränssnittet är baserat på ett rutnät om tre gånger tre, se figur 7. Användaren kan se hur lång den nuvarande sekvensen är, hur många försök det är kvar, och den hittills längsta sekvens användaren lyckats följa. Detta visas med hjälp av text som berättar för användaren när det är dennes tur att trycka in sekvensen, om det är en genomgång av testet eller en poängsättande runda, när användaren svarar rätt eller fel och så vidare. Färger används för att förstärka känslan av vad som händer.



Figur 7: Skärmdump från Memory Test

4.2.2 Testomgång

Som förberedelse inför Memory Test får användaren genomföra en testrunda. Testet påbörjas inte förrän användaren avklarat testrundan för att försäkra att uppgiften är fullt förstådd. Instruktioner visas för användaren i form av en förklarande text.

4.3 Adaptive Self-Assessment Test

Det tredje testet som utvecklades är ett personlighetstest kallat ASAT, vilket står för Adaptive Self-Assessment Test. Användaren får ranka tre påståenden mot varandra, baserat på hur de passar dennes personlighet. Attributen mäts på en skala som sträcker sig från ett attribut till dess motpol. Ett exempel på ett attribut är *konkret*, vars motpol således är *abstrakt*.

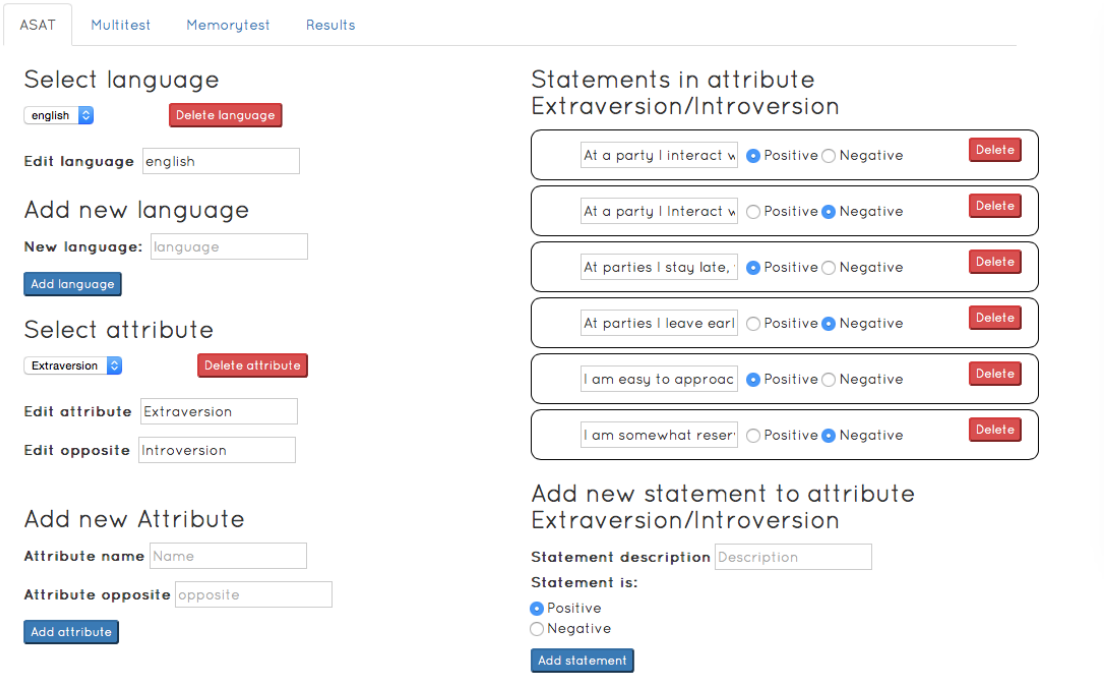
Tre påståenden presenteras för användaren. Dessa ska sedan rangordnas i fallande ordning, där överst är det som stämmer in bäst och underst det som stämmer in sämst. Detta görs genom att genom att klicka, dra och släppa påståendet på önskad position, se figur 8. När användaren har sorterat alla påståenden går denne vidare till nästa uppsättning påståenden.

Figur 8: Skärmdump från ASAT

4.4 Kontrollpanel

Varje test har olika inställningar som går att ändra för att justera svårighetsgraden. Då det inte är användarvänligt att ändra i källkoden för att byta inställningar har en enkel kontrollpanel utvecklats. Panelen kommunicerar med servern som i sin tur synkroniserar inställningarna med databasen.

Kontrollpanelen för Multi Test tillåter ändring av antal rundor ett test ska köras, hur lång tid hela testet ska ta, samt hur lång tid de passiva respektive de aktiva testerna ska ta. Inställningarna för Memory Test inkluderar antal försök, vilken sekvensnivå användaren börjar på samt storleken på ökningen efter varje avklarad nivå. För ASAT går det att lägga till, ta bort, och ändra attribut, språk, och påståenden. En skärmdump av kontrollpanelen för ASAT syns i figur 9.



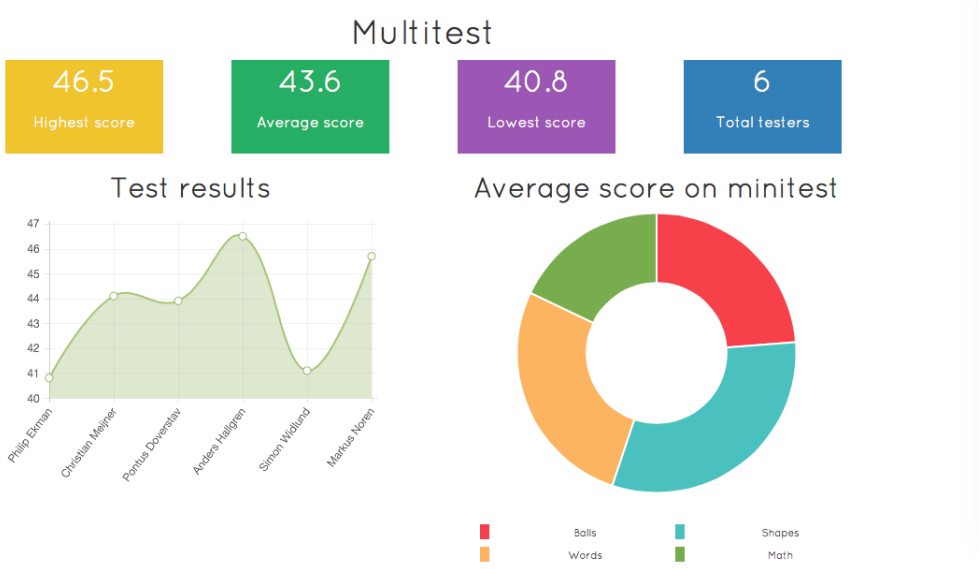
Figur 9: Kontrollpanelen

4.5 Presentation av testresultat

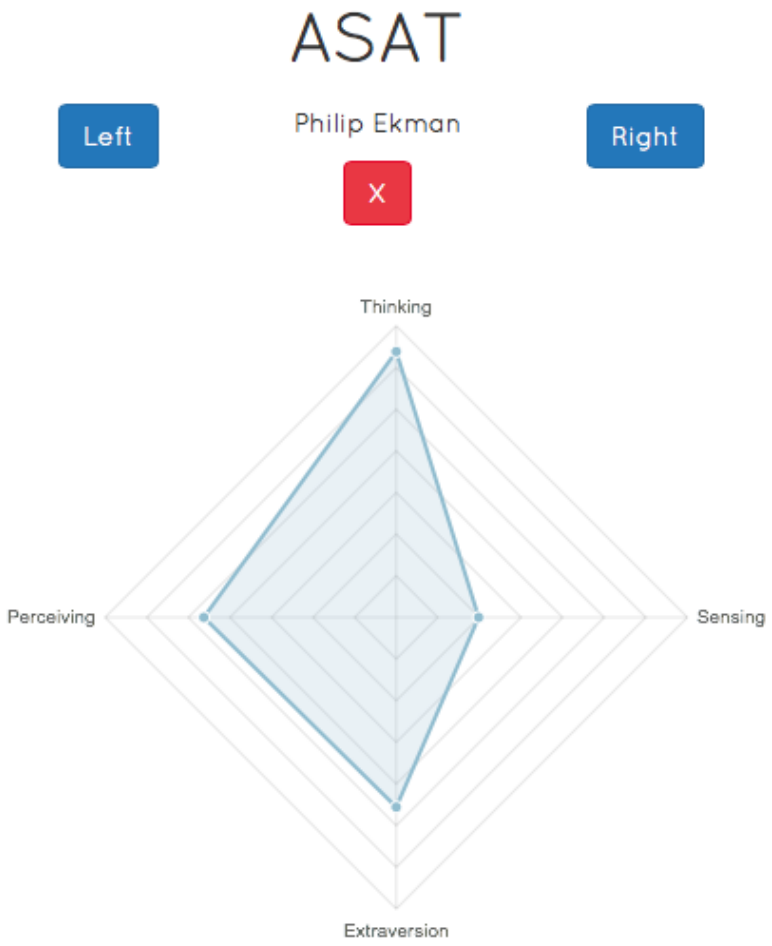
Graferna som syns på skärmdumpen i figur 10, visar upp användarnas resultat i Multi test, medan de fyra mindre panelerna representerar allmän statistik för testet, så som högsta poäng. Cirkeldiagrammet till höger visar upp fördelningen av poäng mellan de små testerna som Multi Test består av.

Resultatvyn för Memory Test innehåller statistik i form av totalpoäng, högst, lägst och genomsnittlig uppnådd sekvens.

I resultatet för ASAT presenteras de dominerande attributen med hjälp av en axel i diagrammet, se figur 11. De attribut som är motpoler till dessa visas inte alls, eftersom användaren då inte anses besitta dessa.



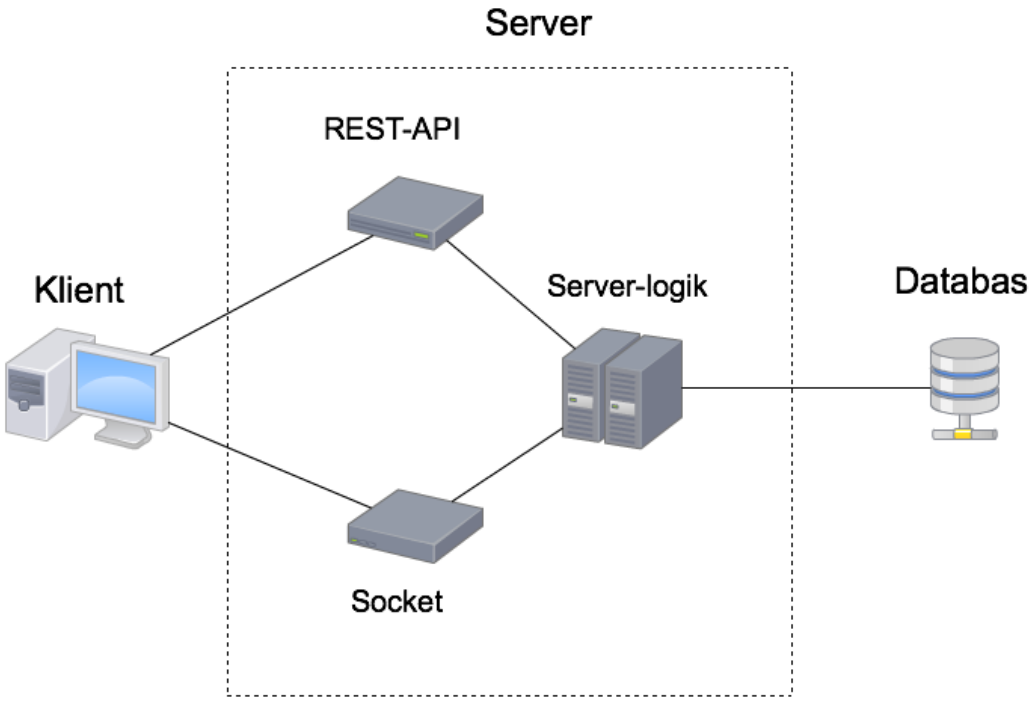
Figur 10: Resultatpanel för Multi test



Figur 11: Resultatpanel för ASAT

5 Implementation

Applikationen bygger på en klient/server-arkitektur. I de flesta webbapplikationer används servern endast för att lagra data som måste sparas under längre tid. Klienten brukar för denna traditionella typ av applikationer stå för logiken. Skill Test fungerar dock inte på detta sätt, då säkerhet legat i fokus, vilket beskrivs närmare i avsnitt 5.7.



Figur 12: Arkitektur

5.1 Designmönster - MVC

I figur 12 ges en överblick av arkitekturen. En MVC-struktur har använts där modellen har separetats till servern, så att den inte går att manipulera hos klienten. På klienten finns vyn och controllern som hanterar användarinput och kommunicerar med modellen på servern via ett REST-api eller med en socket-anslutning, beroende på vilka krav på kommunikation testet har. De tekniker som används

och de designval som gjorts vid konstruktionen av applikationen har resulterat i en bra uppdelning mellan modell, vy och controller.

5.1.1 Vy

Vyn är skriven i HTML och CSS. Bootstrap har använts för att få en bra layout, och för att bättre anpassa applikationen till olika skärmstorlekar. Stöd till smartphones och tablets har dock valts bort, eftersom alla användare bör ha samma premisser i testerna; att använda pekskärm kan ge helt andra resultat än att använda mus och tangentbord.

5.1.2 Controller

Innehållet i vyn kontrolleras med Angular, som hanterar användarinput och kommunicerar med modellen. Controllern och vyn delar minne och vissa funktioner genom ett så kallat scope. Angular håller variabler och funktioner som är definerade i scopet uppdaterade så att de hela tiden har samma data, och en ändring från controllern kommer direkt synas i vyn. Detta är ett centralt begrepp i Angular, och kallas två-vägsdatabindning. Detta gör att att controllern och vyn kan dela det som är nödvändigt, utan att ha tillgång till onödiga detaljer.

5.1.3 Modell

Controllern kommunicerar med modellen genom ett REST-API eller via sockets. Att använda sig av ett REST-API ger en lös koppling mellan controllern och modellen. Ändringar och förfrågningar i modellen sker genom HTTP-begäranden, som bara specifierar vilken begäran som ska göras, och skickar med eventuella parametrar som JSON-objekt. Således behöver inte klienten känna till några implemenationsdetaljer på servern. Detta ger ett säkert system, eftersom användaren inte får någon inblick i serverlogiken. Det gör det också möjligt att helt byta ut klienten mot en klient som inte nödvändigtvis behöver vara implementerad med Javascript, så länge den använder sig av HTTP-begäranden och JSON-objekt.

5.2 Poängalgoritm för Multi Test

Varje uppgift rapporterar användarens poäng efter att uppgiften avslutats, och när en runda avslutas berättar testet detta för algoritmen. Varje runda associeras med de poäng som uppnåddes av de avslutade uppgifterna och sparas undan tills den totala poängen ska beräknas. När testet är slut rapporteras även detta och poängalgoritmen räknar ut slutpoängen.

Poängalgoritmen för Multi Test har två mål - att i första hand belöna simultankapacitet och att i andra hand belöna uppnådda poäng. Den uppnår detta genom att jämföra skillnaden i procent av uppnådda maxpoäng mellan passiva och aktiva uppgifter. Beroende på hur nära dessa procentsatser är appliceras en multiplikator på poängen som är högre ju mindre skillnaden är. På så sätt belönas användare som delat sin uppmärksamhet lika mellan de olika uppgifterna och användare som fokuserat på en uppgift straffas.

Mängd uppnådda poäng belönas genom att algoritmen ger en viss mängd bonuspoäng baserat på andel av maxpoäng som användaren uppnådde. Denna andel adderas även till ovan nämnda multiplikator. Utan det andra målet skulle simultankapacitet värderas för högt. En användare som uppnåt en högre poäng men med större differens mellan de individuella uppgifterna skulle då värderas lägre än en användare med mindre poäng men bättre uppskattad simultanförmåga.

Denna process upprepas för varje samling av poäng från rundor vilket leder till att varje runda blir associerad med en poäng. Snittet av summan av dessa blir användarens slutpoäng på testet.

Anta att en runda består av två uppgifter, en passiv och en aktiv.

S_p = poäng på passiv uppgift
$S_{a1}...S_{ai}$ = poäng på aktiv uppgift
M_p = maximal poäng på passiv uppgift
$M_{a1}...M_{ai}$ = maximal poäng på aktiv uppgift
A = antal inrapporterade poäng

M är multiplikatorn baserad på aktiva och passiva uppgifters differens och beräknas:

$$M = \frac{\sum_i \left(1 - \left|\frac{S_p}{M_p} - \frac{S_{ai}}{M_{ai}}\right|\right)}{A - 1} + \frac{S_p + \sum_i S_{ai}}{M_p + \sum_i M_{ai}} \tag{4}$$

B är bonuspoäng baserat på procent av maximal poäng uppnådd och beräknas:

$$B = 100 * \frac{S_p + \sum_i S_{ai}}{M_p + \sum_i M_{ai}} \tag{5}$$

Slutpoängen S beräknas:

$$S = \frac{\frac{\left(S_p + \sum_i S_{ai}\right) * M}{A} + B}{A} \tag{6}$$

5.3 Elo-rating i ASAT

Att ställa olika attribut mot varandra och räkna hur många gånger de värderas högre än andra ger en indikation på hur starka alternativt svaga de olika attributen är. Denna metod kan dock vara bristfällig - ett svagt attribut som favoriseras över ett annat svagt attribut tar egentligen bara fram det bästa av två dåliga alternativ. Det behövs någon form av balansering som förtydligar när ett attribut är verkligt starkt istället för det bättre av två dåliga.

Då ett attribut som ständigt värderas som sämre än andra helt plötsligt favoriseras över ett riktigt högt värderat attribut visar det på en obalans. För att motverka detta behövs någon form av relativ poängfördelning. Därför har en egenutvecklad variant av Elo-rating använts.

Attributen agerar spelare och jämförelsen mellan attributen ses som en match. Den traditionella metoden för Elo-rating är anpassad för möten mellan två spelare, men i detta fall fanns ett behov av att kunna sätta fler än två attribut mot varandra. Därför har en modifierad version av "Judge Diplomacy Player"-rankning [37] använts, som i grunden är baserad på Elo-rating men är anpassad för att matcha flera spelare samtidigt.

Nedan följer en matematisk beskrivning av den modifierade Elo-rating-algoritmen.

R = attributets tidigare rating
M = antal attribut som deltar i jämförelsen
G = antal gånger ett attribut har jämförts
F = antal attribut i jämförelsen som anses ha en fullt utvärderad rating

Från början har alla attribut samma rating. Denna ska inte börja på 0 utan istället sättas i mitten på skalan som används. I detta fall, då ratingskalan går från 0-2000 ska ett attributs rating initialt sättas till 1000.

Då tre attribut ställs mot varandra sker poängfördelningen S efter en jämförelsen enligt följande. Det vinnande attributet får 3 poäng, tvåan erhåller 1 poäng och det förlorande får -1 poäng.

Ett attributs nya rating R_N baseras på tre variabler. Antalet vunna poäng i jämförelsen S, antalet poäng attributet förväntades att vinna i jämförelsen X samt attributets tidigare erfarenhet E. Förändringen i attributets rating beräknas enligt följande formel:

$$E * (1 + \frac{F}{M}) * (S - X) \tag{7}$$

Attributets nya rating R_N blir då:

$$R_N = R + E * (1 + \frac{F}{M}) * (S - X) \tag{8}$$

E är attributets tidigare erfarenhet och beräknas:

$$E = 1 + \frac{40}{10 + G}; \tag{9}$$

E kommer att konvergera mot 1. Detta innebär att förändringar till en början kommer att vara större och med tiden gradvis minska.

X baseras på attributets tidigare erfarenhet och styrka samt den totala erfarenheten och styrkan av alla attribut i jämförelsen:

$$X = M * \frac{e^{R/500}}{T} \tag{10}$$

T är summan av styrkan på samtliga deltagande attributs rating och beräknas med:

$$T = \sum_{i=1}^{M} e^{Ri/500} \tag{11}$$

5.4 Resultatpanel

Med ett anrop till servern hämtas resultaten av en resultatpanel som ligger på klienten. Servern svarar med ett JSON som innehåller poäng för alla användare som gjort något utav de tillgängliga testerna. För minskad belastning på servern hämtas alla resultat med ett anrop och det är sedan klientens uppgift att sortera och visa datan på ett tydligt sätt.

Presentationen av testresultaten sker med hjälp av ett Javascript-bibliotek som heter Chart.js (Chart). Biblioteket är till för att skapa olika typer av grafer. Datan som hämtas från servern sorteras och läggs in i listor. Sorteringen sammankopplar alla resultat till varje unik användare och slutligen skickas listorna in till Chart för bearbetning och presentation.

5.5 Kommunikation

Då mycket av applikationens logik ligger på servern finns höga krav på väl valda kommunikationsprotokoll för att skicka data mellan komponenterna. Memory Test och Balls använder sockets, då de har höga krav på realtidsprestanda för att ge en god användarupplevelse. Socket-implementationen använder sig av så kallade namnrymder (*eng. namespaces*) för att optimera bort redundanta anslutningar. I en namnrymd finns ett antal händelser (*eng. events*) för att koppla ihop klienten och servern på ett liknande sätt som REST-API:et.

Applikationen har ett omfattande REST-API som är till för att leverera HTTP-data då klienten skickar en förfrågan. Detta sker typiskt vid diskreta tillstånd som när ett nytt test startas eller när ett test avslutas.

5.6 Synkronisering med kunds system

De användare som gör tester i applikationen kommer från Software Skills system där varje användare har ett unikt id. Detta id skickas med till testet i webbadressen och systemet ser till att spara ner detta till databasen tillsammans med ett namn, även detta från Software Skills, och ett antal tomma variabler som skall representera poängen. Då användaren gjort klart ett test uppdateras poängvariabeln för motsvarande test.

5.7 Säkerhet

En nackdel med att köra Javascript i en webbläsare är att en teknisk användare fritt kan modifiera koden. I webbapplikationer laddas all klientkod in i webbläsaren, och därför kommer all kod som finns i frontenden vara exponerad för användaren, och det går att kommunicera med logiken med javascriptkonsollen som finns inbyggd i alla moderna webbläsare. Detta innebar en del extra arbete. Så stor del av applikationens logik som möjligt ska placeras på servern, och endast logik som absolut måste vara på klienten ska finnas där. Grundtanken blir att bara anrop som direkt motsvaras av något som kan göras med användarinmatning ska kunna göras från klientkoden; på så sätt blir det lika svårt att styra applikationen från javascriptkonsollen som från användargränssnittet.

6 Diskussion

Att projektet dubblerades av misstag präglade i synnerhet de första veckornas arbete med en viss osäkerhet hos gruppen. Detta då vi inte i samma utsträckning som andra grupper hade en väl definierad uppgift, utan var tvungna att formulera den själva i samråd med kunden. Arbetet med planeringen blev därför omfattande, vilket i sin tur gjorde att vi började med själva utvecklingsarbetet lite senare än vad som hade varit optimalt.

Det har dock inte endast varit till nackdel för gruppen att uppgiften var löst definierad från projektets start. Möjligheten gavs då att tillsammans med kunden utforma produkten utan att vara bunden av en redan existerande projektspecifikation.

6.1 Måluppfyllnad

Målen för projektet kan delas in i allmänna och deltest-specifika. De allmänna målen kan sammanfattas som följer, mer detaljerad beskrivning av målen återfinnes i avsnitt 1.2. Alla deltester ska vara:

- Validerade genom användartester.
- Tillräckligt skilda från dataspel för att kännas seriösa.
- Enkla att komma igång med.
- Skyddade mot fusk.
- Trevliga att använda, med enkla gränssnitt och enhetlig formgivning

Kunden har önskat att applikationen testas utförligt på utomstående, oinsatta personer. Genom användartester kan svårighetsgrad justeras, buggar upptäckas, och otydligheter i till exempel instruktioner komma fram. Tyvärr har några sådana tester inte kunnat utföras, då applikationen inte var mogen för användartester i tillräckligt god tid. För att användartester ska vara meningsfulla behöver produkten ha kommit så långt i utvecklingen att för utvecklarna uppenbara fel och brister inte förekommer. Att inga användartester gjorts betyder att produkten heller inte kan anses vara en färdig produkt, utan en prototyp. Målet var att skapa just en färdig produkt, således har detta mål inte uppfyllts.

Både gruppen och kunden är överens om att produkten inte känns oseriös eller dataspelsliknande. Det fanns vid projektets start en viss oro hos kunden att testerna skulle kännas oseriösa om de blev för spel-lika, men genom att undvika nivåer, överdriven poängräkning och relaterade koncept kunde detta undvikas.

6.1.1 Användartester

En brist för samtliga tester är avsaknaden av ordentlig validering, både av testresultaten som genereras men också av avändargränssnittet. Ett mål för projektet var att leverera en färdig produkt till Software Skills, men på grund av att mycket tid har lagts på planering och konkretisering av mycket löst definierade projektmål är inte testerna evaluerade i den utsträckning att de går att börja använda idag.

6.1.2 Multi Test

Målet med Multi Test var att testa simultanförmåga. Mindre användarstudier har utförts, främst för att hitta tekniska problem och för att se om användaren förstår vad denne ska göra. Ytterliggare validering krävs för att se vad som faktiskt testas. Detta kan göras genom att jämföra resultatet som Multi Test ger med resultatet från liknade tester som är väl ansedda.

6.1.3 Memory Test

Minnestestet, liksom Multi Test, har genomgått användartester för funktionalitet och för användarvänlighet. Det kan tyckas trivialt att detta test utvärderar korttidsminne, då testet går ut på att memorera en sekvens av händelser, men vi skulle gärna se ytterliggare validering för att göra testet mer trovärdigt. Att jämföra med etablerade minnestester skulle även här vara lämpligt.

6.1.4 Adaptive Self-Assessment Test

Personlighetstester är svåra att utvärdera av den enkla anledningen att de syftar till att förenkla något så komplext som en hel personlighet till ett antal värden på en förbestämd skala. I projektets slutskede användes frågor från MBTI för att internt testa och utvärdera ASAT. Frågorna har fått en del kritik just för

att förenkla bilden av människors personlighet, men det är ändå väl beprövat material som anses kunna mäta en persons personlighet. Ett problem med detta är att ASAT och MBTI inte har samma upplägg. I vårt test ska användaren sortera ett antal påståenden efter hur väl de stämmer in på dennes personlighet, medan användaren i MBTI ges två påståenden och ska ange vilket som stämmer bäst. Resultaten utvärderas också på helt olika sätt. I MBTI räknas resultatet ganska primitivt, där varje fråga ger en poäng mot det attribut som stämmer in på det valda svaret. I ASAT används istället tidigare nämnda Elo-rating, beskrivet i detalj i avsnitt 5.3.

Om mer tid funnits hade utförliga användartester gjorts för att evaluera ASAT. Ett bra alternativ hade varit att utföra A/B-tester där MBTI används för att evaluera resultatet för den ena testgruppen och ASAT för den andra. Frågor/påståenden från MBTI hade då använts för båda grupperna för att på så sätt kunna urskilja och utvärdera skillnaden i resultat.

ASAT kan potentiellt ge ett mer rättvist och korrekt resultat än MBTI. Detta tack vara den Elo-baserade algoritmen som på ett bättre sätt visar relativa skillnader mellan attributen. En uppenbar brist med MBTI:s linjära poängsättning är att ett attribut kan få poäng trots att det egentligen bara är det minst dåliga av två dåliga alternativ [38]. ASAT hanterar sådana situationer tack vare att poängen ett attribut erhåller baseras på styrkan av attributet det ställs mot.

6.2 Grafisk design

Grafisk design har varit en utmaning, då ingen i gruppen är van att arbeta med design. Målet med designen var att hålla den enhetlig, lättförstådd, och att ge användaren ett professionellt intryck. Trots begränsad erfarenhet kan målet anses vara uppfyllt, även om applikationen hade kunnat se mer lockande ut.

Designen är den del av projektet som ingen i gruppen velat ta tag i, då ingen känt sig bekväm med uppgiften. Detta har lett till att designutvecklingen konstant har legat steget bakom den tekniska utvecklingen. Sammanfattningsvis kan det konstateras att mer tid och tanke kring designen hade varit lämpligt, men att gruppen samtidigt är nöjd med resultatet utifrån de förutsättnignar som fanns.

6.3 Kodkvalitet

En mål med projektet var att utveckla en platform som har stöd för utbyggnad, så att testerna kunde dela mycket logik och implementation av nya tester kunde göras så enkel som möjligt. Visuellt ser testerna ganska enhetliga ut, och alla tester använder sig av samma ramverk och samma sätt att kommunicera mellan server och klient. Själva testlogiken följer dock inte en gemensam struktur, och applikationen hade därför kunnat vara mer modulär, utbyggbar och lättare att förstå om en sådan struktur bestämts på förhand.

6.4 Jämförelse med andra ramverk

Här diskuteras fördelar och nackdelar med tekniker som hade kunnat användas istället för MEAN.

6.4.1 Meteor

Likt MEAN använder sig Meteor av Mongo och Node, och ramverket det använder för att skapa en front-end liknar Angular. Det är dock inte lika skalbart som MEAN och det gör mycket jobb åt utvecklaren vilket hade minskar lärovärdet i projektet. Att det även fanns tidigare erfarenhet av MEAN inom gruppen gjorde valet av MEAN enklare.

6.4.2 LAMP

LAMP, ursprungligen Linux/Apache/MySql/PHP, är en samling backend-tekniker som lämpar sig bra för webbapplikationer [39]. Hade LAMP valts som backend för projektet hade serverkoden behövts skrivas i PHP eller Python. Två språk som gruppmedlemmarna på förhand inte hade någon större kunskap inom. Samma sak gällde även Javascript, men oavsett val av serverspråk hade klienten fortfarande behövts skrivas med hjälp av Javascript. Valet stod då mellan att lära sig Javascript samt något annat programspråk alternativt att välja en teknik där Javascript uteslutande kunde användas. För att hålla nere den mängd ny teknik som behövde läras föll valet på MEAN.

6.4.3 MySQL

MySQL är en så kallad relationsdatabas där det finns inbyggd stöd för relationer mellan objekt. NoSQL är en relationslös databasmodell, vilket betyder att utvecklaren själv implementerar relationer mellan data. Trots att det finns många fördelar med en MySQL-databas föll valet på att använda Mongo. Dels för att det är en del av MEAN-stacken, dels för att dokumenten i Mongo är uppbyggda med hjälp av BSON, vilket är en variant av JSON. Detta ger en stark koppling och potentiellt mycket god prestanda, då data från klienten skickas till servern i JSON-format[40].

6.5 Arkitektur och säkerhet

Att använda sockets var inte självklart i början av utvecklingen. Ingen i gruppen hade kunskap om tekniken sedan tidigare, så gedigen reasearch behövde göras för att i första hand hitta denna teknik, och sedan ta reda på hur den används och hur den bör användas i detta projekt. Biblioteket (Socket.IO) som användes för att implementera sockets var sparsamt dokumenterat, och krävde därför en del experimenterande innan det kunde implementeras fullt ut.

Beslutet att göra applikationen fusksäker togs redan innan utvecklingen började. Gruppen var överens om att detta skulle bli en lärorik och intressant teknisk utmaning, som också skulle ge mervärde till kunden, men insåg inte från början hur mycket detta skulle komma att påverka utvecklingsarbetet. Arkitekturmässigt är den resulterande produkten på många sätt lik ett online-spel med flerspelarstöd. Likheterna ligger i att kritisk logik körs på servern för att undvika fusk, och att servern håller koll på var flera spelare "befinner sig", vilket i detta fall innebär att ha separata variabler för varje användare.

Det går inte att hindra en klient från att manipulera det som exekveras i webbläsaren. Servern är emellertid utom räckhåll för klienten, förutom genom kommunikation med de HTTP-förfrågningar som vi har definierat att servern ska ta emot. Eftersom all kritisk logik lagts på servern och de HTTP-förfrågningar som kan göras motsvarar små ändringar så som till exempel användarinput, får en potentiell fuskare inte någon fördel av att skicka förfrågningarna genom javascriptkonsollen. Det är snarare fördelaktigt för användaren att använda sig av användargränssnittet eftersom det automatiserar de förfrågningarna.

6.6 Gruppens erfarenhet

Gruppens bristande erfarenhet av modern webbutveckling har varit en begränsande faktor i utvecklingsarbetet. Att teknikchefen arbetat med MEAN förut har dock varit till stor hjälp. För resten av gruppen har det varit en stor utmaning att uppskatta tidsåtgången för olika uppgifter, då det ofta ingick att lära sig något helt nytt genom att läsa dokumentation och kodexempel. Majoriteten av gruppen hade ingen eller minimal erfarenhet av Javascript sedan tidigare, vilket resulterade i många spenderade timmar på att lära sig språket och de ramverk som använts. Att lära sig ett programmeringsspråk samtidigt som flera mycket omfattande ramverk var för flera i gruppen en stor utmaning.

Samtliga i gruppen hade tidigare använt både Scrum och versionshantering i mindre projekt, vilket har underlättat arbetsprocessen. För projekt av denna storlek saknades dock erfarenhet.

6.7 Versionshantering

I en grupp om sex personer är det oerhört viktigt med väl fungerande versionshantering, vilket gruppen var medvetna om från början. Det beslutades tidigt att använda en viss förgreningspolicy, som finns beskriven i avsnitt 3.5. Denna har över lag fungerat mycket bra. Några få gånger har det blivit problem att sammanfoga grenar som innehåller olika funktioner i samma deltest. Det är dock snarare kommunikationsbrist som orsakat dessa problem.

Utöver sporadiska sammanfogningsproblem har det blivit några mindre incidenter, men Git är gjort så att allting alltid går att ångra, så dessa problem har alltid gått att lösa.

6.8 Scrum

Utvecklingsprocessen har i stor utsträckning präglats av Scrum. Användandet av en agil utvecklingsprocess har varit till stor fördel för gruppen, då den ger flexibiliteten som var nödvändig för att skapa den produkt som kunden vill ha. Flexibilitet har varit mycket viktigt, då en tydlig produktspecifikation från kund inte stod att finna vid projektets start.

Användningen av en backlog är ett centralt koncept inom Scrum, men något som

gruppen till en början hade svårt för att genomföra på ett bra sätt. Uppgifterna i backlogen tenderade att vara vagt formulerade, eller alldeles för omfattande. Det förekom exempelvis att ingen i gruppen ens visste vad en uppgift betydde, som den stod formulerad i backlogen. Behovet av en bättre strukturerad backlog som kontinueligt uppdaterades upptäcktes tidigt, och varje spurt har inneburit en klar förbättring i kvaliteten på backlogen.

6.9 Kodtestning

I början av projektet var gruppen ivriga att komma igång med utvecklingen, då planeringsrapporten hade tagit mer tid än vad som var tänkt. Ungefär halvvägs in i projektet insågs att det hade varit bra att skriva enhetstester, då det ger möjlighet att snabbare hitta fel som kommer allt eftersom koden ändras. Utvecklingsarbetet hade vid denna tidpunkt nått så långt att det helt enkelt skulle ta för mycket tid att gå tillbaka och anpassa allt till en mer testdriven utveckling. Gruppen har under projektets gång stött på så pass många buggar att fördelarna med kodtester är uppenbara, och det har blivit en lärdom för framtiden.

7 Slutsats

Projektet saknade från början en klar projektbeskrivning, vilket har gett en viss särprägel jämfört med andra kandidatarbeten. Mycket tid har lagts på konkretisering av uppgiften i samverkan med Software Skills, men även till stor del på egen hand. Detta har dock gett oss mycket frihet att själva utveckla idéer och koncept.

Många kreativa idéer har kunnat implementeras. Kunden har bestämt att de ska vidareuteckla produkten. För att testerna ska kunna användas i rekryteringssyfte behövs ytterliggare utvärdering av testerna för att säkerställa rättvisa resultat och rimlig svårighetsgrad, då detta inte hunnits med inom projektets tidsram.

I ASAT har frågor och attribut lånats från MBTI för att kunna testas och felsökas. Software Skills vill istället använda eget material, och har en beteendevetare som ska arbeta vidare med det.

Multi Test och Memory Test behöver validering av själva testresultaten. Software Skills behöver göra egna experiment med användare där man på annat sätt undersökt dennes minnes- eller simultanförmåga, och göra jämförelser med resultatet som uppnåtts i Memory Test och i Multi Test.

Vi tror att den traditionella rekryteringsprocessen, där mycket avgörs av CV och på intervjuer, kan komma att bytas ut mot en mer pragmatisk process där tester kommer vara en naturlig del för att med större säkerhet kunna avgöra hur någon kommer prestera på arbetsplatsen. Vår förhoppning är att Skill Test ska hjälpa arbetsgivare att anställa rätt personer.

Referenser

[1] Statistiska Centralbyrån. (2014, April 23). Utbildningsnivån ökar i Sverige [Online]. Tillgänglig: http://www.scb.se/sv_/Hitta-statistik/Statistik-efter-amne/Utbildning-och-forskning/Befolkningens-utbildning/Befolkningens-utbildning/9568/9575/Behallare-for-Press/372838/ [Besökt: 2 juni 2015]

[2] Stiftelsen för Strategisk Forskning. (2014). Vartannat jobb automatiseras inom 20 år [Online]. Tillgänglig: http://www.stratresearch.se/documents/folder.pdf [Besökt: 2 juni 2015]

[3] Brad Remillard. (Okänt datum). *Most Company's Hiring Process Is Not A Process* [Online]. Tillgänglig: http://www.impacthiringsolutions.com/blog/most-companys-hiring-process-is-not-a-process/ [Besökt: 2 juni 2015]

[4] Daniel. J. Levitin. (2015, January 18). *Why the modern world is bad for your brain* [Online] Tillgänglig: http://www.theguardian.com/science/2015/jan/18/modern-world-bad-for-brain-daniel-j-levitin-organized-mind-information-overload [Besökt: 2 juni 2015]

[5] Brock Eide, Fernette Eide. (2014, December). *Brains on Fire: The Multinodality of Gifted Thinkers* [Online]. Tillgänglig: http://education.jhu.edu/PD/newhorizons/Neurosciences/articles/Brains%20on%20Fire/ [Besökt: 2 juni 2015]

[6] Tomas Chamorro-Premuzic. (2014, May 12). *Seven Common (But Irrational) Reasons For Hating Personality Tests* [Online]. Tillgänglig: http://www.forbes.com/sites/tomaspremuzic/2014/05/12/seven-common-but-irrational-reasons-for-hating-personality-tests/ [Besökt: 2 juni 2015]

[7] David J. Pittenger. (1993). *Measuring the MBTI... And Coming Up Short* [Online]. Tillgänglig: http://www.indiana.edu/~jobtalk/Articles/develop/mbti.pdf [Besökt: 2 juni 2015]

[8] Försvarsmakten. (Okänt datum). *Försvarsmakten - Samarbetstestet* [Online]. Tillgänglig: http://team.forsvarsmakten.se [Besökt: 2 juni 2015]

[9] Human Benchmark. (Okänt datum). *Human Benchmark* [Online]. Tillgänglig: http://www.humanbenchmark.com/tests/memory [Besökt: 2 juni 2015]

[10] Daniel Nations. (Okänt datum). *What is a Web Application?* [Online]. Tillgänglig: http://webtrends.about.com/od/webapplications/a/web_application.htm [Besökt: 2 juni 2015]

[11] Ciprian Borodescu. (2013, July 29). *Web Sites vs. Web Apps: What the experts think* [Online]. Tillgänglig: http://www.visionmobile.com/blog/2013/07/web-sites-vs-web-apps-what-the-experts-think/ [Besökt: 2 juni 2015]

[12] The World Wide Web Consortium. (2014, October 28). *HTML & CSS* [Online]. Tillgänglig: http://www.w3.org/standards/webdesign/htmlcss [Besökt: 2 juni 2015]

[13] M. Haverbeke, "What is Javascript?", in *Eloquent JavaScript: A Modern Introduction to Programming*, 2nd ed. San Francisco, CA: No Starch Press, 2014, pp. 7-8.

[14] T. Bray. The JavaScript Object Notation (JSON) Data Interchange Format. Internet RFC's, ISSN 2070-1721, RFC 6455, 2014.

[15] Jim Gettys. Hypertext Transfer Protocol - HTTP/1.1. Internet RFC's, ISSN 2070-1721, RFC 2616, 1999.

[16] Ian Hickson. The WebSocket Protocol. Internet RFC's, ISSN 2070-1721, RFC 6455, 2011.

[17] Various contributors (Github project). (Okänt datum). *Get Started: Chat application* [Online]. Tillgänglig: http://socket.io/get-started/chat/ [Besökt: 2 juni 2015]

[18] Thomas Roy Fielding, "Architectural Styles and the Design of Network-based Software Architectures", Ph.D. dissertation, University of California, Irvine, 2000. Tillgänglig: http://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm [Besökt: 2 juni 2015]

[19] Valeri Karpov. (2013, April 30). *The MEAN Stack: MongoDB, ExpressJS, AngularJS and Node.js* [Online]. Tillgänglig: http://blog.mongodb.org/post/49262866911/the-mean-stack-mongodb-expressjs-angularjs-and [Besökt: 2 juni 2015]

[20] MongoDB, Inc. (Okänt datum). *Introduction to MongoDB* [Online]. Tillgänglig: http://docs.mongodb.org/manual/core/introduction/ [Besökt: 2 juni 2015]

[21] Steve Hall. (2014, July 1). *MySQL vs MongoDB* [Online]. Tillgänglig: http://www.scriptrock.com/articles/mysql-vs-mongodb [Besökt: 2 juni 2015]

[22] Google, Inc. (Okänt datum). *Developer Guide - Introduction* [Online]. Tillgänglig: https://docs.angularjs.org/guide/introduction [Besökt: 2 juni 2015]

[23] Google, Inc. (Okänt datum). *Developer Guide - Directives* [Online]. Tillgänglig: https://docs.angularjs.org/guide/directive [Besökt: 2 juni 2015]

[24] Google, Inc. (Okänt datum). *Developer Guide - Data Binding* [Online]. Tillgänglig: https://docs.angularjs.org/guide/databinding [Besökt: 2 juni 2015]

[25] Various contributors (Github project). (Okänt år). *About Node.js* [Online]. Tillgänglig: https://nodejs.org/about/ [Besökt: 2 juni 2015]

[26] Evan M. Hahn, "What is Express?", in *Express.js in Action*, 2014 [Online PDF]. Tillgänglig: http://www.manning.com/hahn/Express_MEAP_CH01.pdf [Besökt: 2 juni 2015]

[27] Tutorials Point, "Bootstrap Overview", in *Bootstrap Tutorial*, okänt år [Online PDF]. Tillgänglig: http://www.tutorialspoint.com/bootstrap/bootstrap_tutorial.pdf [Besökt: 2 juni 2015]

[28] Martin Fowler. (2006, July 18). *GUI Architectures* [Online]. Tillgänglig: http://martinfowler.com/eaaDev/uiArchs.html [Besökt: 2 juni 2015]

[29] Various contributors (Github project). (Okänt datum). *Getting Started - A Short History of Git* [Online]. Tillgänglig: http://git-scm.com/book/en/v2/Getting-Started-A-Short-History-of-Git [Besökt: 2 juni 2015]

[30] GitHub, Inc. (Okänt datum) *Github.com features* [Online]. Tillgänglig: https://github.com/features/ [Besökt: 2 juni 2015]

[31] Martin Fowler and Jim Highsmith in *The Agile Manifesto*, 2001 [Online PDF]. Tillgänglig: http://www.pmp-projects.org/Agile-Manifesto.pdf [Besökt: 2 juni 2015]

[32] Jeff Sutherland in *The Scrum Papers: Nut, Bolts, and Origins of an Agile Framework*, 2012 [Online PDF]. Tillgänglig: http://jeffsutherland.com/ScrumPapers.pdf [Besökt: 2 juni 2015]

[33] Per Runeson och Martin Höst in *Guidelines for conducting and reporting case study research in software engineering*, 2008 [Online PDF] Tillgänglig: http://lup.lub.lu.se/luur/download?func=downloadFile&recordOId=1276781&fileOId=1276782 [Besökt: 2 juni 2015]

[34] Amazon.com, Inc. (Okänt datum). *How A/B Testing Works* [Online]. Tillgänglig: https://developer.amazon.com/public/apis/manage/ab-testing/doc/how-ab-testing-works [Besökt: 2 juni 2015]

[35] Lars Magnus Vattum and Halvard Arntzen. (2009). Using ELO ratings for match result prediction in association football [Online]. Tillgänglig: http://www.sciencedirect.com/science/article/pii/S0169207009001708 [Besökt: 2 juni 2015]

[36] Vincent Driessen. (2010, January 5). *A successful Git branching model* [Online]. Tillgänglig: http://nvie.com/posts/a-successful-git-branching-model/ [Besökt: 2 juni 2015]

[37] Ted Miller. (Okänt datum). *Judge Diplomacy Player Ratings – Description* [Online]. Tillgänglig: http://diplom.org/Email/Ratings/JDPR/describe.html [Besökt: 2 juni 2015]

[38] Development Edge Consulting Ltd. (Okänt datum) *Myers-Briggs Type Indicator (MBTI)* [Online]. Tillgänglig: http://www.dec.co.th/mbti_explanation.htm [Besökt: 2 juni 2015]

[39] TechTarget (2008, December) LAMP (Linux, Apache, MySQL, PHP) [Online]. Tillgänglig: http://searchenterpriselinux.techtarget.com/definition/LAMP [Besökt: 2 juni 2015]

[40] Luke P. Issac. (2014, January 14). *SQL vs NoSQL Database Differences Explained with few Example DB* [Online]. Tillgänglig: http://www.thegeekstuff.com/2014/01/sql-vs-nosql-db/ [Besökt: 2 juni 2015]