Tidslogg

Datum	Timmar	Arbetsbeskrivning
2014-12-16	2.5	Läste och strukturerade uppgiften
2014-12-17	9.5	Uppgift 3,4,5
2014-12-18	9	Uppgift 5,6,7

Uppgift 1- Planera

Steg	Plan. tid	Beskrivning	Verkl. tid	Problem på vägen
1	40 m	**Uppgift 1 - Planera** Förstå alla uppgifter och planera dem.	35 m	Svårt att förstå exakt vad det är som ska göras i uppgifterna.
2	15 m	**Skapa reposition på github**	5 m	Inga problem
3	1 t	**Uppgift 2 – Testplan** • Läs på om testplan.	1 t	Svårt att skiva tester och plan innan man skriver klassen?! Mycket att information att själv hitta.
4	1 t	• Skriv en testplan som beskriver mitt testprojekt.	45 m	Skriven, omständigt, kommer att behövas redigeras senare.
5	15 m	**Uppgift 3 – Design och implementation** • Sätt upp kodningsmiljön	30 m	Konfigurering av en webbutvecklingsmiljö med jasmine
6	2 t	• Designa och implementera ett användningsfall, OBS. minst två klasser.	2 t 35 m	Gick ganska smärtfritt, hoppas att jag gjort rätt.
7	30 m	• Dokumentera klasserna	45 m	Klasserna dokumenterade på ett logiskt sätt. Med diagram, hade svårigheter att hitta verktyg för diagram.
7	15 m	• Sammanställ användningsfallmodell, användningsfallsbeskrivningar, klasser med beskrivningar samt kod för inlämning.	2 m	Inte svårt
8	30 m	**Uppgift 4 – Enhetstestning** • Läs på om xUnit, testsviter, testfixturer och testfall.	2 t	Här fick jag läsa på väldigt mycket och konfigurera Jasmin och dess plugins. Mycket pill.
9	2 t	• Gör analys av två klasser och specificera testsvit som innehåller testfixturer och testfall. • Beskriv testdata och test mot förväntat resultat, och hur jag kommit fram till detta.	2 t	Här fick jag också leta på information, tog tid att förstå exakt hur jag skulle göra.
10	30 m	**Uppgift 5 Implementera testsviten och kör** • Läs på om hur man bygger testklasser för min kod.	0	Det här steget gjorde jag redan i steg 8.
11	2 t	• Bygg testklasser för mina två klasser. Exekvera testklasserna och samla in testdata.	4 t	Mycket jobb att förstå Jasmine, och bygga upp tester, tog tid.
12	30 m	• Analysera och föreslå förbättringar	40 m	Gick relativt smidigt

13	30 m	**Uppgift 6 – Integrationstestning** • Läs på om integrationstestning	20 m	Gick lättare att förstå nu när man förstått enhetstestning
14	3 t	• Ta fram testbeskrivningar för integrationsfall	3 t 30 m	Detta tog tid.
15	30 m	**Uppgift 7 – Reflektion** • Reflektera kring svårigheter i att planera och genomföra uppgiften.	15 m	Inga konstigheter.

Beräknad tidsåtgång 925 minuter / 15.4 timmar.
Verklig tidsåtgång 1260 minuter / 21 timmar.

Uppgift 2- Testplan

Produkt

Gymnastiktävlingssystem för gymnastikligan. Version 1.

Referenser

1. Gymnasietävlingssystemet; Användningsfall 2 ”Skapa tävlingstillfälle för säsong”, version 1.
2. Gymnasietävlingssystemet; Supplementary specification, version 2.

Introduktion och Bakgrund

Den här testplanen är skapad för gymnastiktävlingssystemet ämnad att användas av gymnastikligan och alla inblandade såsom; gymnaster, domare, administratörer och administratörer.

För att kunna bygga ett bättre system och att slippa att ett högt antal fel i systemet når slutanvändaren så skriver vi här en plan för hur vi skapar tester för systemets programmerade klasser. Detta minskar risken för fel i det slutgiltiga systemet och ger en bättre användarupplevelse.

Den här testplanen berör användningsfall 2 ”Skapa tävlingstillfälle för säsong” (se referens 1) där användaren skapar ett tävlingstillfälle för en säsong.

Testade artefakter

Bokningssystemet i gymnastiktävlingssystemet.

Testprocess Testmål (objectives)

- Enhetstester i systemet
    - Alla uppgifter måste vara korrekta för ett **tävlingstillfälle**
        - Ansvarig: Johnny
        - Testtyp: Whitebox testning (Villkor)
        - Stoppregler:
            - Följande uppgifter i tävlingstillfället måste vara av rätt typ.
                - Starttid, sluttid
    - Alla uppgifter måste vara korrekta för en **deltävling**
        - Ansvarig: Johnny
        - Testtyp: Whitebox testning (Villkor)
        - Stoppregler:

        - Följande uppifter i tävlingstillfället måste vara av rätt typ.
            - Starttid, sluttid, gymnastikgren, junior eller seniortävling, män eller kvinnor, allround eller individuell och domare.
    - Måste ha domare registrerad efter att detta registreras.
        - Ansvarig: Johnny
        - Testtyp: Whitebox testning
        - Stoppregler:
            - Om inte domaren finns registrerad efter registrering.
            - Om någon domarinformation saknas vid registrering.
    - Information om deltävling de ska döma på skickas till domare
        - Ansvarig: Johnny
        - Testtyp: Whitebox testning
        - Stoppregler:
            - Om ingen information skickas till Domare trotts att uppgifter finns.

Rutiner för dokumentation av testresultat

Varje aktuell testkörning loggas i en textfil information om:

- När testet exekverades
- Testets utfall
- Vem som exekverat testet

Testomgivning

Testsystemets omgivning består av följande hårdvara och mjukvara. En webbserver krävs. Klasserna är byggda med javascript men kan inte köras utan webbserver på grund av att testfixturer i form av JSON objekt hämtas med hjälp av ajax.

Tänk på att funktionalitet i mjukvaran kan ändras i och med nya (eller äldre) versioner av mjukvaran.

Server

System:	Virtuell maskin i VMware ESXi
Minne:	1 GB
Hårddiskutrymme:	17 GB
Operativsystem:	Ubuntu 14.04 Server
Webbserver mjukvara:	nginx 1.4.6
Javascript testmjukvara:	Jasmine 2.1.3 pluginet jasmine-jquery 2.0.5 jquery 2.1.1

Klient

Operativsystem:	Windows 7
Webbläsare:	Chrome

Uppgift 3 - Design och implementation

Objektet Competition

Det här objektet rymmer själva tävlingstillfället som användaren skapar. Detta objekt rymmer sedan i sin tur deltävlingar i en arrayen ”eventsArray” som finns som en egenskap i objektet.

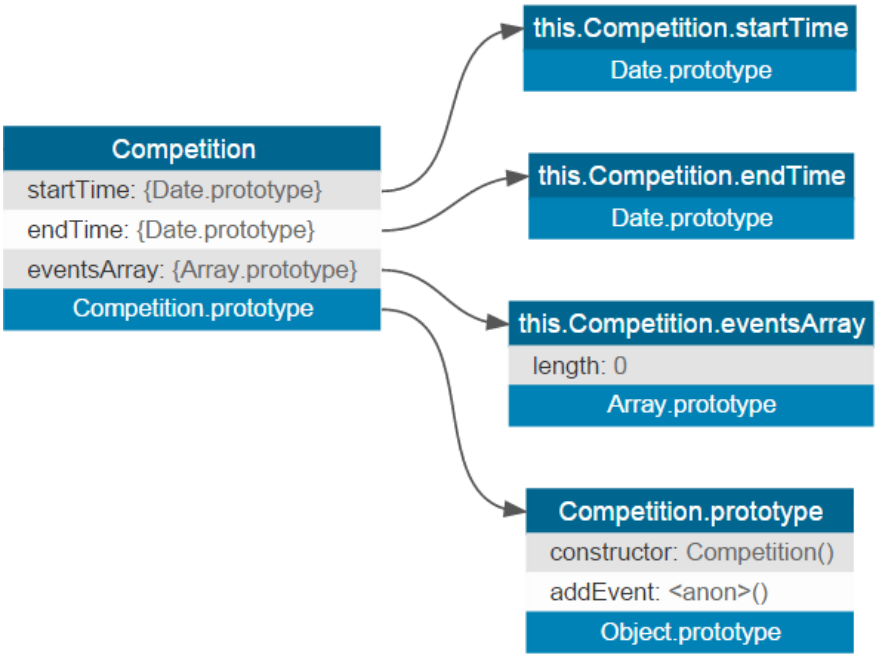
Egenskaper med getters och setters:

- startTime Starttiden för tävlingen i form av ett Date Objekt.
- endTime Sluttiden för tävlingen i form av ett Date Objekt.
- eventsArray Array med Event objekt (deltävlings objekt).

Metoder:

Metoderna ligger som en prototype, för att spara resurser ifall det någon gång i framtiden skapas många Competition objekt.

- addEvent(eventObj) Lägger till Event objekt i eventsArray:en.



Objektet Event

Det här objektet rymmer själva deltävlingen som, när den är fylld med tävlingens information i sina egenskaper ska läggas in i Competition objektets array.

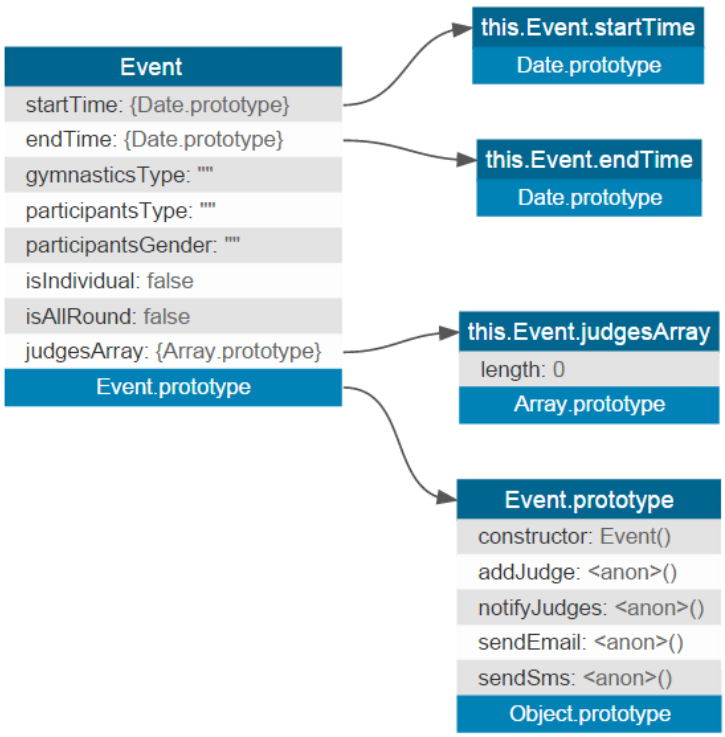
Egenskaper med getters och setters:

- startTime Starttiden för tävlingen i form av ett Date Objekt.
- endTime Sluttiden för tävlingen i form av ett Date Objekt.
- gymnasticsType Sträng som representerar tävlingsgren.
- participantsType Sträng som representerar om deltävlingen är för juniorer eller seniorer.
- paticipantsGender Sträng som representerar om deltävlingen är för män eller kvinnor.
- isIndividual Boolean som representerar om deltävlingen är individuell.
- isAllRound Boolean som representerar om deltävlingen är allround.
- judgesArray Array med domare i objektform som ska döma i deltävlingen.

Metoder:

Metoderna ligger som en prototype, för att spara resurser ifall det någon gång i framtiden skapas många Event objekt.

- addJudge(judgeObj) Lägger till ett Domar objekt i judgesArray:en.
- notifyJudges() Informerar domarna i judgesArray:en via e-post eller sms om att de ska döma i deltävlingen.
- sendEmail(emailAdress) Skickar e-post till e-postadress (ej implementerad)
- sendSms(phoneNumber) Skickar sms till telefonnummer (ej implementerad)



Uppgift 4 - Enhetstestning

Testsvit för skapande av tävlingar

Beskrivning av testfixturer

- */spec/fixtures/bad_values.json*
  - Json-fil formaterad fixtur-fil med felaktiga värden för skapande och manipulation av alla berörda objekt.
- */spec/fixtures/competition_good_values.json*
  - Json-fil formaterad fixtur-fil med korrekta värden för skapande och manipulation av ett Competition objekt .
- */spec/fixtures/event_good_values.json*
  - Json-fil formaterad fixtur-fil med korrekta värden för skapande och manipulation av ett Event objekt .
- */spec/fixtures/integration_values.json*
  - Json-fil formaterad fixtur-fil med korrekta värden för skapande och manipulation av både Event och Competition objekt för integrationstestning.

Enhetstest för ”klassen” Competition.

ID	Scenario	Skapa tävlingstillfälle Kommando	Angiven Starttid	Angiven Sluttid	Förväntat resultat
1	Scenario 1	Ja	Giltig	Giltig	Tävlingstillfället skapas.
2	Scenario 2	Ja	Giltig	Ogiltig	Användaren får felmeddelande
3	Scenario 3	Ja	Ogiltig		Användaren får felmeddelande

Enhetstest för ”klassen” Event.

ID	Scenario	Skapa del-tävling körs.	Angiven Starttid	Angiven Sluttid	Angiven gymn. gren	Junior eller Senior Angiven	Man eller kvinna angiven	Invidivi-duell angiven	Allround angiven	Domare angiven	Förväntat resultat
1	Scenario 1	Ja	Giltig	Giltig	Giltig	Giltig	Giltig	Giltig	Giltig	Giltig	Deltävlingen skapas och lagras i Competition objektet. Domare underrättas.
2	Scenario 2	Ja	Giltig	Giltig	Giltig	Giltig	Giltig	Giltig	Giltig	Ogiltig	Användaren får felmeddelande
3	Scenario 3	Ja	Giltig	Giltig	Giltig	Giltig	Giltig	Giltig	Ogiltig		Användaren får felmeddelande
4	Scenario 4	Ja	Giltig	Giltig	Giltig	Giltig	Giltig	Ogiltig			Användaren får felmeddelande
5	Scenario 5	Ja	Giltig	Giltig	Giltig	Giltig	Ogiltig				Användaren får felmeddelande
6	Scenario 6	Ja	Giltig	Giltig	Giltig	Ogiltig					Användaren får felmeddelande
7	Scenario 7	Ja	Giltig	Giltig	Ogiltig						Användaren får felmeddelande
8	Scenario 8	Ja	Giltig	Ogiltig							Användaren får felmeddelande
9	Scenario 9	Ja	Ogiltig								Användaren får felmeddelande

Beskrivning

Eftersom objekten validerar argumenten när de skapas så är det här de flesta fel ska fångas upp. Event och Competition objekten har inte så många andra funktioner förutom att bli skapade, informera domare och bli sparade. Bra att veta är att Event objektet efter att det skapas ska läggas in i Competitions objektets array (för event objekt).

Uppgift 5 – Implementera testsviten och kör

Beskrivning av de utförda testerna

I testerna provades det främst om objekten skapades som de skulle utifrån sina konstuktorvärden.

I den första delen av testerna för både ”Competition” och ”Event” objekten så skapades de med ett antal korrekta förutbestämda värden och sedan kontrollerades det att objektets egenskaper överensstämde med konstruktorvärdena.

I den andra delen av testerna så testades det om objekten reagerade som de skulle på att skapas med felaktiga värden.

I ”Event” objektet testades också om metoderna fungerade som de skulle. Men i ”Competition” testades inte den enda testbara metoden ”addEvent” då detta kan ses mer som ett integrationstestfall.

Kommenterad kod för tesklasserna

Testspecifikationsfilen för objektet ”Competition” hittas under sökvägen:
spec/CompetitionSpec.js

Testspecifikationsfilen för objektet ”Events” hittas under sökvägen:
spec/EventSpec.js

Competition ”Klassen”

Konstuktor:

ID	Scenario	Skapa tävlingstillfälle Kommando	Angiven Starttid	Angiven Sluttid	Förväntat resultat	Verkligt resultat
1	1	Ja	1418821210000	1418839210000	Tävlingstillfället skapas.	Tävlingstillfället skapas.
2	2	Ja	1418821210000	t23osjkaw	Användaren får felmeddelande	Användaren får felmeddelande
3	3	Ja	244		Användaren får felmeddelande	Användaren får felmeddelande

Event ”Klassen”

Konstruktor:

ID	Scenario	Skapa del-tävling körs.	Angiven Starttid	Angiven Sluttid	Angiven gymn. gren	Junior eller Senior Angiven	Man eller kvinna angiven	Invidivi-duell angiven	Allround angiven	Domare angiven	Förväntad och verkligt resultat
1	1	Ja	1418821 210000	1418839 210000	trampett	junior	man	true	true	Array med ett objekt med rätt värden.	Deltävlingen skapas och lagras i Competition objektet. Domare underrättas.
2	2	Ja	1418821 210000	1418839 210000	trampett	junior	man	true	true	ioerji20#	Användaren får felmeddelande
3	3	Ja	1418821 210000	1418839 210000	trampett	junior	man	true	t23osjka w		Användaren får felmeddelande
4	4	Ja	1418821 210000	1418839 210000	trampett	junior	man	2424			Användaren får felmeddelande
5	5	Ja	1418821 210000	1418839 210000	trampett	junior	1.343634 233				Användaren får felmeddelande
6	6	Ja	1418821 210000	1418839 210000	trampett	true					Användaren får felmeddelande
7	7	Ja	1418821 210000	1418839 210000	2424						Användaren får felmeddelande
8	8	Ja	1418821 210000	t23osjka w							Användaren får felmeddelande
9	9	Ja	1.343634 233								Användaren får felmeddelande

Metoden addJudge:

ID	Scenario	Deltävling är skapat. (Event objektet)	Lägg till domare: addjudge ()	Förväntat resultat	Verkligt resultat
1	1	Ja	Array med ett objekt med rätt värden.	Domare läggs till i domar arrayen i Event objektet.	Felmeddelande
2	2	Ja	Array med ett objekt med väden som saknas	Användaren får felmeddelande	Användaren får felmeddelande

Metoden notifyJudges:

ID	Scenario	Deltävling är skapat. (Event objektet)	Notifiera domare: addjudge ()	Förväntat resultat	Verkligt resultat
1	1	Ja	Array med två domarobjekt. En med bara e-post, och en annan med bara mobilnummer.	Domare notifieras. En får e-post, en annan får sms.	Felmeddelande

Utfall av tester

15 specs, 1 failure

- Competition
  - should be created with correct properties from good constructor values
  - should throw an error when created with bad constructor endTime argument
  - should throw an error when created with bad constructor startTime argument

- Event
  - should be created with correct properties from good constructor values
  - should throw an error when created with bad constructor startTime argument
  - should throw an error when created with bad constructor endTime argument
  - should throw an error when created with bad constructor gymnasticsType argument
  - should throw an error when created with bad constructor participantsType argument
  - should throw an error when created with bad constructor participantsGender argument
  - should throw an error when created with bad constructor isIndividual argument
  - should throw an error when created with bad constructor isAllRound argument
  - should throw an error when created with bad constructor judgesArray argument
  - should have correct judgesArray after using 'addJudge' method
  - should throw an error after using 'addJudge' with faulty argument.
  - should should run 'sendEmail' method or 'sendSms' method correct after using 'notifyJudges' method

Som vi ser så gick de flesta tester lyckligtvis igenom. Det här är nog tack vare att jag försökte vara ganska noga med valideringen av data när jag skrev koden. Vi kan se att valideringen av egenskaperna verkar fungerar som de ska. Men metoden ”addJudge” i ”Event” objektet fungerar inte önskvärt.

Förbättringar

Själva felet i addJudge metoden visade sig ligga i själva testet, där en hel array med domare från testfixturen skickades till metoden addJudge, men addJudge förväntar sig bara ett objekt. Och när detta åtgärdades så testades metoden korrekt.

Andra testmässiga förbättringar som jag kan se nu i efterhand är att metoden addJudge skulle behövas testas ytterligare för att säkerställa att följande if sats fungerar:

```
if(!(judgeObj.fullName) || !(judgeObj.email) && !(judgeObj.sms))
```

Bättre validering av participantsType (Junior eller Senior) behövs verkligen. Detta är ingenting som syns av testerna, utan testerna provar bara värdetypsvalidering. Här kan det bli fel i framtiden. Likaså är det med participantsGender som bara ska kunna vara man eller kvinna och inget annat.

Ytterligare validering av data skulle behövas läggas till i Event objektet för att säkerställa att alla värden är satta. Kanske en metod vid namn isComplete() går igenom objektets värden och ser till detta skulle vara bra.

Något ytterst viktigt att validera är telefonnumret (som används vid sms) och e-post adressen. Jag valde att inte bygga in denna funktionalitet, men validering av dessa uppgifter skulle verkligen behövas i metoderna sendEmail och sendSms.

Uppgift 6 – Integrationstestning

Eftersom jag här kommer att fokusera på integrationstestning, hur klasserna fungerar med varandra så kommer jag att utgå ifrån att Competition objektet och Event objekten skapades med värden för sig själva, men inte nödvändigtvis med korrekta värden för att kunna interagera med varandra.

Beskrivning av de utförda testerna

Jag utgick till en början utifrån användsningsfallsbeskrivningen, vilket är en bra grundmall. Men den saknar utförlig beskrivning för avancerad integration just emellan Competition klassen och Event klassen och därför utgick jag mer ifrån testdatat och den tidigare dokumentationen som återfinns på uppgift 4 och 5. Det här är därför en vidareutveckling av vad som förväntas hända med ifall objektens data innehåller fel. Testdata har angetts så långt det varit möjligt.

Kommenterad kod för testklassen

Testspecifikationsfilen för objektet det här intregrationstestet emellan ”Competition” och ”Event” objektet hittas under sökvägen:
spec/IntegrationSpec.js

Integrationstest för ”klasserna” Competition och Event

Test Case Id	**Förvillkor**	**Event objektet läggs till i Competition objekt genom Competition.addEvent(EventObj)**	**Förväntat resultat**	**Verkligt resultat**
1	Objekten Competition och Event skapade.	Event.startTime (1418811210000) är längre än Competition.startTime (1418821210000)	Ett felmeddelande kastas.	Inget felmeddelande kastades
2	Objekten Competition och Event skapade.	Event.endTime (1418849210000) är längre än Competition.endTime (1418839210000)	Ett felmeddelande kastas.	Inget felmeddelande kastades
3	Objekten Competition och Event skapade	Event objektet saknar viktig information i sina egenskaper.	Ett felmeddelande kastas.	Inget felmeddelande kastades
4	Objekten Competition och Event skapade.	Competition objektet har redan ett identiskt Event objekt lagrat i sig.	Ett felmeddelande kastas.	Inget felmeddelande kastades
5	Objekten Competition och Event skapade.	Competition objeketet har redan Event objekt med registerade domare i sig. Det nya Event objektet äger rum under samma tid och har samma domare registrerad. Domaren blir alltså dubbelbokad.	Ett felmeddelande kastas.	Inget felmeddelande kastades

Utfall av tester

Eftersom klassen saknar checkar för detta så lyckades inte någon av testerna. Dessa inbyggda kontroller tänkte jag helt enkelt inte på förrän i teststadiet.

- Integration
  - should throw an error when Competition.addEvent(Event) and Event.startTime < Competition.startTime
  - should throw an error when Competition.addEvent(Event) and Event.endTime > Competition.endTime
  - should throw an error when Competition.addEvent(Event) and Event has missing properties
  - should throw an error when Competition.addEvent(Event) and Event is allready present in the Competition object.
  - should throw an error when Competition.eventsArray has an Event object with a judge assigned to it and a new event is added over the same period of time with the same judge.

Uppgift 7 – Reflektion

När man inte förstår vad det är man ska göra så är det fruktansvärt svårt att planera. Det är min största lärdom ifrån denna laboration. Jag kunde omöjligt föreställa mig hur svårt det var att greppa alla steg i uppgifterna. Jag fick leta information på nätet om det mesta, boken gav mig inte mycket hjälp. Mycket tid gick också åt att leta på, förstå och installera testningsverkyget jasmine för javascript. För att få fixturer att fungera så var jag tvungen att lägga till jquery och ett plugin till jasmin för att det skulle fungera. Men när jag väl fick upp miljön så fungerade den väldigt bra. Dock fick jag flytta allt från min egen klientdator till en server, jasmine läste in fixturer så gjordes detta över http vilket krävde en webbserver, vilket jag också specifierade i min testplan i uppgift 3.

Jag upplevde att stegen flöt ihop mer än någonsin i verkligheten. I början så hade jag en idé om att att skulle kunna göras stegvis, från steg 1 till 2 till 3 och så vidare. Men i verkligheten blev den hel del hopp emellan stegen och uppgifterna för att i nästa sekund krångla lite mer med filstrukturen eller inställningar i jasmine miljön. Det är helt enkelt omöjligt att få till precis allt rätt första gången man gjorde en uppgift, ett steg eller ställde in utvecklingsmiljön. Detta gjorde det också ganska svårt att bedöma exakt hur mycket tid det gick åt varje steg och uppgift. Men i stora drag så stämmer tiden per uppgift.

Kodmässigt så upplever jag det väldigt svårt att planera och tänka in integrationstestfunktionalitet i båda klasserna. Jag visste att de skulle interagera, men kunde inte tänka ut att Competition objektet faktiskt skulle validera Event objektet i samband med att den lades till i Competition objektet. Självklart kunde jag förstå att objekten i sig själva validerar sina egna värden vid skapande, men de också måste hålla koll på varandras värden vid integration. Detta var någonting jag upptäckte i integrationssteget, då jag i princip inte hade någon sorts integrationsvalidering av värden alls.

I det stora hela har det varit en relativt jobbig laboration. Att aldrig ha hållit på med testning av kod tidigare märks väldigt väl när man väl ska göra det. Men detta har också lärt mig en hel del om testning av kod, hur det fungerar och varför man gör det.