Laboration 1 1DV404

Valt programmeringsspråk

Javascript

Tidslog

Datum	Antal timmar	Arbetsbeskrivning
14-11-20	13.00-17.00 (4 timmar)	Uppgift 1a, 1b, 1c
14-11-23	14.00-19.00 (5 timmar)	Uppgift 2a, 2b, 3a, 3b
14-11-24	17.00-22.00 (5 timmar)	Uppgift 4a, 4b

Uppgift 1 – Tre “enkla” programmeringsuppgifter.

Uppgift 1a

Strategi: Jag delar upp uppgiften i dessa rimliga steg:

1. Skapa projektmapp, skapa dokumenten som jag kommer att jobba i.
2. Skriva en funktion som läser ifrån användarens input och kolla så att den fungerar.
3. Skriva kod för att räkna antalet a:n.
4. Presentera resultatet för användaren.

Steg	Planerad tidsåtgång	Beskrivning	Fel, gjorda av mig och problem på vägen	Verklig tidsåtgång
1	15 min	Skapa projektmapp för uppgiften, skapa html dokument och js dokument. Skriva dokumentet där användaren skriver sin input	Hade inte räknat med stegen involverade i ett skapa ett nytt projekt i Cloud9 var så många. (Skapade ett eget repository på Github för laborationen i kursen och inporterade den därifrån.)	10 min
2	15 min	Påbörja funktionen som läser användarens input och gör eventuell felhantering.	Hade ej tänkt på alla steg som omfattades. Formuläret ville ha false returnerat, förstod senare att detta skulle göras på två ställen. Hade inte heller räknat med att jag redan här var tvungen att använda reguljära uttryck för att se att om sökströngen har a:n.	15 min
3	30 min	Skriva en loop eller tillämpa färdig sökfunktion för att räkna antalet a, respektive A:n som finns i den inmatade textströngen.	Eftersom sökfunktionen redan av skriven i en valideringsfunktion i det tidigare steget var det bara att använda den sig av den koden för detta steg.	7 min
4	15 min	Presentera resultatet för användaren.	Det här steget var inte så svårt, och lade inte ner så mycket tid jag tänkt från början på design.	6 min

Reflektioner kring uppgift 1a

När jag planerar ett programmeringsjobb är det svårt för mig att tänka ut alla saker som kan gå fel eller alla de små stegen som ingår i programmet för att få det att fungera. Jag har inga problem med att fundera ut de största momenten, men när började med uppgiften hade jag till exempel inte tänkt på vilket jobb det låg i att få igång själv utvecklingsmiljön såsom; github, skapa filer och mappar, lägga till standardkod i html dokumenten för att få dem att fungera. Utan nej, jag hade mer kodmässigt tänkt ut hur jag skulle lösa de stora problemen, men inte på de små stegen man på något sätt förutsätter att de nästan fixar sig själva på något sätt, fast det inte alls är så.

Förutom att se förbi de så arbetsuppgifter som jag helt glömde bort så hade jag en annan tanke med i planeringsstadiet som jag hade svårt att besluta mig för: Hur mycket tid ska man reservera för allt som kan gå fel? Jag tänkte nog som så att det är bra om lägger på lite tid eftersom det säkert är någonting som kommer att krångla, vilket det gjorde. Men detta är väldigt svårt att uppskatta.

En annan reflektion jag gjorde under kodprocessen var den att i början så hade jag en klar bild för hur planeringen skulle vara och vad jag tyckte var

viktigt att lägga energi och tid på. Men väl inne i programmeringsprocessen så upptäckte jag att min bild av vad som måste göras ändrades. När man väl har koden framför sig så ser man klarare och enklare vad det är som är viktigt, något som inte är så lätt när man satt där och planerade uppgiften.

Uppgift 1b

Strategi: Då jag har en hel del kod jag kan återanvända så blir antalet steg mindre i denna uppgift:

1. Kopiera och anpassa kod från förgående uppgift.
2. Skriv kod som räknar ut antal udda och jämna siffror
3. Presentera resultatet, samt justering av kod i efterhand för bättre resultat.

Steg	**Planerad tidsåtgång**	**Beskrivning**	**Fel, gjorda av mig och problem på vägen**	**Verklig tidsåtgång**
1	<15 min	Kopiera HTML och JS dokument ifrån förra uppgiften. Ta bort överflödig kod.	Allt gick ganska bra faktiskt. Tror jag. Nej, glömde länka till nya JS filen. + 1 minut på uppgiften	6 min
2	15 min	Skriv koden som räknar antalet udda och jämna siffror i from av loop eller reguljärt tryck.	Det blev en loop med ifsatser som testade siffrorna. Fick skriva om valideringsfunktionen för att säkerställa att användaren bara skriver in siffror. Skrev också klart koden där värdet visas för användaren pga felsökning av värdet.	17 min
3	15 min	Presentera resultatet för användaren plus finjusteringar i koden. Samt Git push.	Presentationen för användaren slutfördes i föregående steg. Kom på förbättringar som jag kunde göra i den föregående uppgiften. Fick problem med att Cloud9 inte ville pusha till Github, fick felsöka och fick det till slut att fungera med ”git push -u origin master”	20 min

Reflektioner kring uppgift 1b

Något som verkligen jag tyckte under denna uppgift var att tänkta ut i vilka steg jag skulle göra saker. Jag hade färdig kod ifrån den förra uppgiften som jag kunde återanvända. Och den minsta angivna tiden var 15 minuter. Jag kände att det hela handlade om att komma på steg som kunde vara upp till 15 minuter långa, så då använde jag mig av den strategin. På så sätt reserverade jag även tid ifall något skulle krångla, vilket det också gjorde i steg 2 och 3.

Här började få problem med Cloud 9. Jag upptäckte att utvecklingsmiljön inte pushade koden till github som det skulle. Jag kände mig stressad och kände att nu blev jag tvungen att fixa detta, vilket tog tid ifrån mitt steg jag hade planerat. Jag fick lite mer erfarenhet som gör att jag kan planera framtida javascript projekt bättre: Jag upptäckte att validering av användardata är ganska krångligt. Det finns inga riktigt bra funktioner för att försäkra sig om att användaren har angett rätt data, utan man får forska mycket för att hitta sätt som verkligen fungerar på all sorts input.

I slutet av uppgiften kom jag också på några kodmässiga förbättringar som vi lärt oss i webbteknik 1 och tillämpade dem även i uppgift 1a, det tyckte jag förtjänade lite övertid ifrån det tredje steget ifrån denna uppgift.

Uppgift 1c

Strategi: Även här kan jag återanvända en del kod, men fokus kommer att ligga på själva uträkningskoden, så därför delar jag in det momentet i två steg.

1. Kopiera och anpassa kod från förgående uppgift.
2. Skriv kod som validerar användarens data och sedan delar upp användarens sträng för analys.
3. Skriv kod som räknar ut det näst största talet.
4. Presentera resultatet, samt justering av kod i efterhand för bättre resultat.

Steg	Planerad tidsåtgång	Beskrivning	Fel, gjorda av mig och problem på vägen	Verklig tidsåtgång
1	<15 min	Kopiera HTML och JS dokument ifrån förra uppgiften. Ta bort överflödig kod.	Allt gick bra. Fick tänka till lite för att komma ihåg att länka den omdöpta js filen i html dokumentet.	6 min
2	15 min	Skriv kod som validerar användarens data och sedan delar upp användarens sträng för analys i en array.	Valideringsfunktionen tog längre tid än väntat på grund av att minustecknet är tvungen att vara på rätt sida av siffrorna.	19 min
3	30 min	Skriv en loop eller använd den statiska klassen Array:s metoder för att räkna ut det näst största värdet.	Detta gick mycket snabbare jag trodde eftersom det ej var så vårt att sortera arrayen och skriva ut det näst högsta värdet. Var tvungen att komplettera felsökningen med en kontroll att användaren har angett minst två värden.	12 min
4	15 min	Presentera resultatet för användaren plus finjusteringar i koden. Samt Git push.	Några måsvingar fixades, git push gick felfritt. Upptäckte att 10 heltal var tvungna att anges samt inga Arrays. Omskrivning av en hel del kod. Lade till en css fil för storleken på textarea. C9 började gå segt och krångla. Pushade nya versionen till slut till github.	1 timme 15 min

Reflektioner Kring uppgift 1c

Här översteg tiden återigen på grund av validering av användardata i steg 2. Av någon anledning så är jag tidsoptimistisk på den punkten.
Jag missbedömde tiden stort på steg 3, där mycket av koden blev klar i steget före med just valideringsfunktionen. Men i steg fyra, efter jag i princip var klar med uppgiften så upptäckte jag meningen *”Du får inte använda arrayer (eller andra datastrukturer) i den här uppgiften.”*.
Jag suckade och insåg att jag nu får skiva om stora delar av koden för uppgiften. Min tidsplan sprack därmed direkt. Mina egna misstag kan jag bara ta till mig och försöka att inte upprepa tills nästa gång. Så nästa gång ska jag helt enkelt se att läsa instruktionerna väldigt noga. Jag pratade tidigare i 1a om att ”ens interna planering i huvudet” ändras när man väl är inne i kodprocessen och upptäcker saker om man inte tänkte på i planeringen och tidsplanen inte kommer att hålla. Vad är viktigt då? Att hålla planeringen och försöka leva upp till den? Eller ett stanna upp och ändra planeringen så att den passar till nuläget, då man sett uppgiften med nya ögon, med mer förståelse för det verkliga problemet.
Ett viktigt steg som jag också lärde mig efter uppgift 1a var att lägga till ett steg och reservera tid där jag går igenom koden en extra gång för att rätta till eventuella programmeringsmisstag och försöka förbättra koden genom nya smarta lösningar som man kanske inte såg tidigare. Denna reserverade tid visade sig vara väldigt bra att sätta och jag gjorde programmets kod mycket bättre och effektivare.

Uppgift 2a

Fler planeringsstadier

En viktig del i det hela vore att ha fler ”planeringsstadier” i processen, då man stannar upp och analyserar allt på nytt och eventuellt gör upp en ny planering och strategi. Ifall man då kanske upptäckte att man missbedömt tidsåtgången till hela uppgiften i den första planeringen och att tiden inte räcker till det är en sak. Men genom stanna upp och göra upp en ny plan för att vara så effektivt som möjligt använda tiden som finns kvar. När man väl är inne i processen har man också större insikt om vad som verkligen behöver göras och vilka delar som är viktiga att lägga tid på.

Försök att förutse så mycket som möjligt i planeringsstadiet.

Innan man väl sätter igång och tar på sig uppgiften så underlättar det ju en hel del om man förutsett så mycket som möjligt. Att man tänkt ut vilka verktyg man ska använda. Vilken utvecklingsmiljö man ska använda och hur man ska sätta upp den. Att man verkligen förstått uppgiften som ska göras innan man börjar. Man skulle kunna rita en mindmap med olika steg om hur man skulle bära sig åt, med små kodexempel med lösningar på problemet som ska lösas och att de passar ihop med uppgiftens krav.

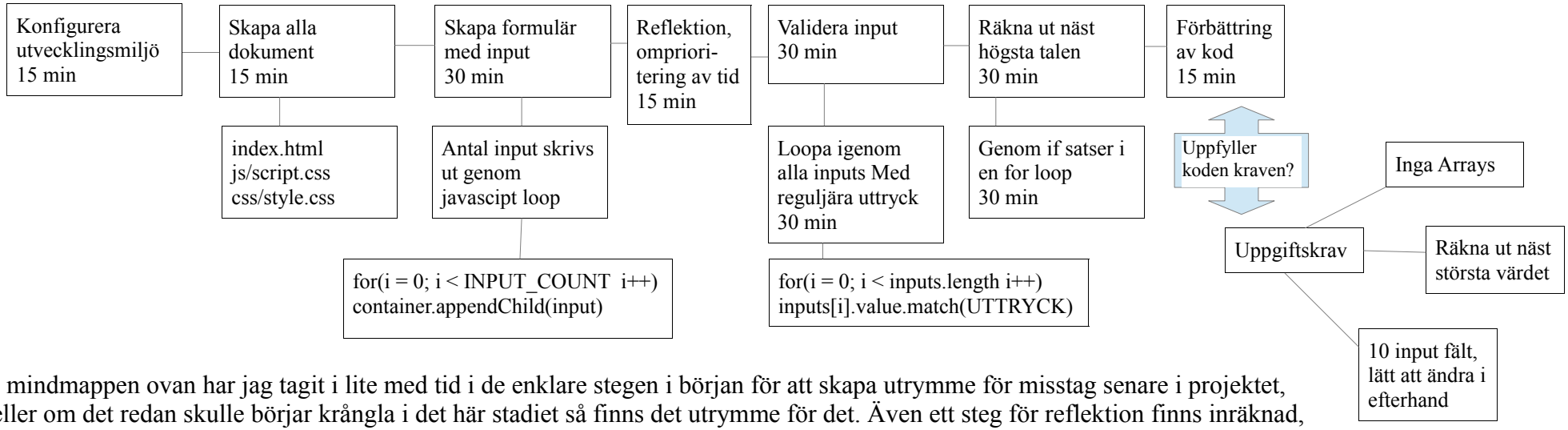
Uppgift 2b

Exempel på konkreta åtgärder vore att:

1. Att ha fler planeringsstadier i processen, att stanna upp och reflektera över arbetet, tiden och eventuellt prioritera om för att hinna med ändå.
2. Försöka förutse allt i planeringsstadiet. Rita en mindmap över över alla stegen med små kodexempel, lösningar och se om de passar ihop med uppgiftens krav.
3. Ett ytterst viktigt moment, som speciellt jag tycker gäller mig är att räkna med att det kommer att gå åt skogen någonstans, och att ge sig själv tid till det. Det är så lätt att vara för lösningsfokuserad på ett problem och därför inte se alla risker som finns på vägen. Därför vore det bra att räkna med att vissa saker kan ta lite längre tid än normalt, och på så sätt så får man utrymme när det väl krånglar. Så länge inte allt krånglar.

Uppgift 2c

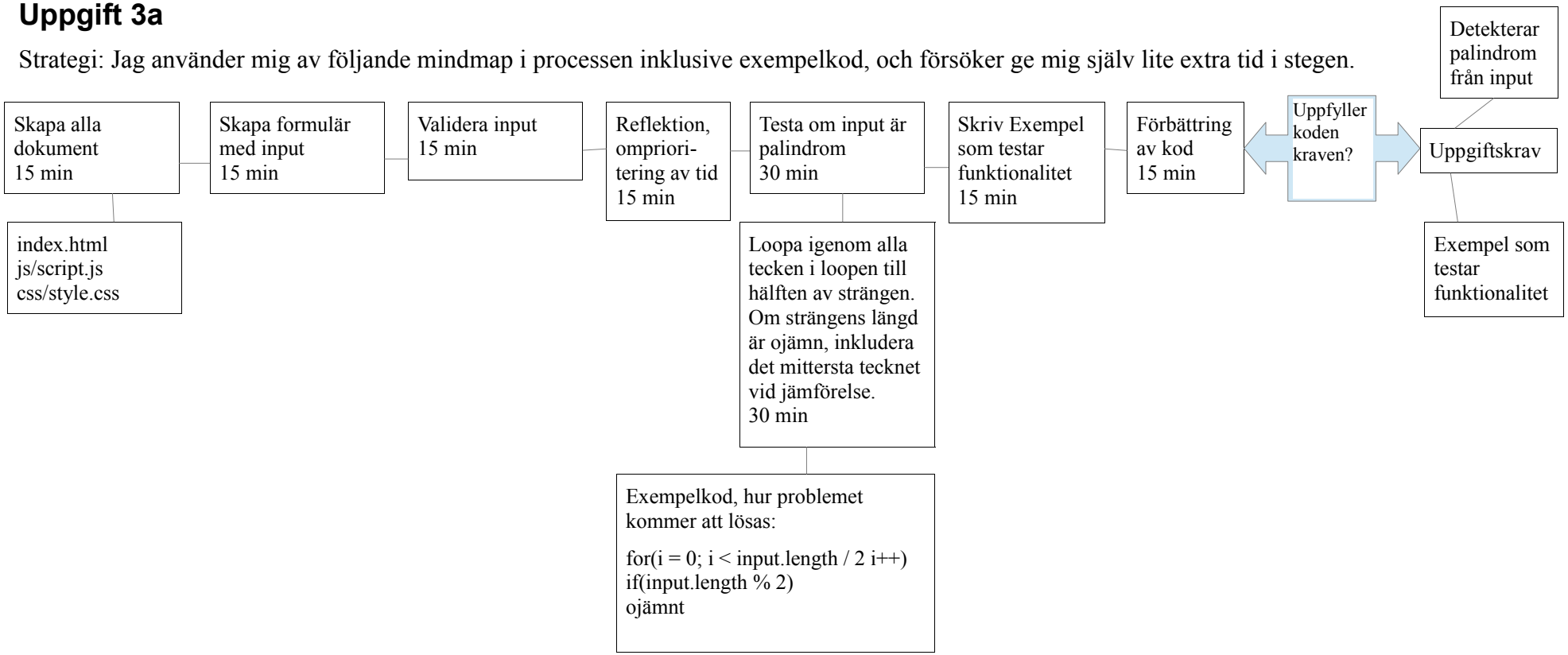
En förbättring i mitt tidigare arbete exempelvis uppgift 1c vore att rita en mindmap, som skulle kunna se ut såhär:



I mindmappen ovan har jag tagit i lite med tid i de enklare stegen i början för att skapa utrymme för misstag senare i projektet, eller om det redan skulle börjar krångla i det här stadiet så finns det utrymme för det. Även ett steg för reflektion finns inräknad, det här som ett steg för att andas ut och reflektera över vad som är viktigt nu när den bättre insyn i uppgiften finns. Även ett sista steg för förbättring av kod finns inlagd, för att just förbättra koden eller om tiden är knapp; en buffert för andra steg som drar ut över tiden.

Uppgift 3a

Strategi: Jag använder mig av följande mindmap i processen inklusive exempelkod, och försöker ge mig själv lite extra tid i stegen.



Steg	Planerad tidsåtgång	Beskrivning	Fel, gjorda av mig och problem på vägen	Verklig tidsåtgång
1	15 min	Skapa alla dokument	All gick smärtfritt	4 min
2	15 min	Skapa formulär med input	Inga problem	2 min
3	15 min	Validera input	Inte mycket validering behövdes.	2 min
4	15 min	Reflektion, omprioritering av tid.	Jag har missbedömt att steg 1-3 tagit så kort tid, koden har har i stort sett klart sedan tidigare uppgifter. Steg 5 kommer att vara det svåraste steget och ta mest tid. Den förutfattade metoden för att testa palindromen kommer antagligen att fungera. Jag har mycket tid över från föregående steg.	5 min
5	30 min	Testa om input är palindrom	Jag kom på att det fanns ett mycket lättare sätt än att loopa igenom strängen. Om man istället vände på strängen med array funktionalitet och jämförde med orginalsträngen fick man enkelt reda på om strängen var en palindrom.	14 min
6	15 min	Skriv exempel som testar funktionalitet	Gick bra att använda sig av funktionen isPalindrom, så därför tog jag ”Förbättring av kod” steget före detta steg, om man ska vara petig.	8 min
7	15 min	Förbättring av kod	Skrev en funktion isPalindrom för återanvändning av kod.	8 min

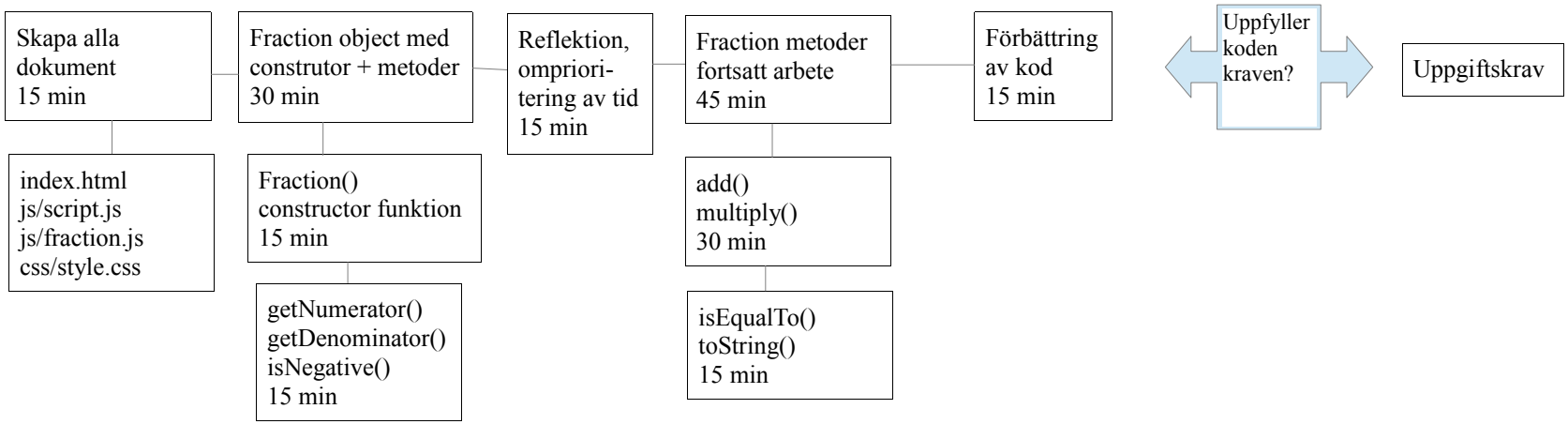
Reflektioner kring uppgift 3a

Denna gång tog steg 1-3 mycket mindre tid än jag räknat med, hela uppgiften tog mindre tid än jag räknat med, vilket är bättre än att allt tar mycket mer tid än man räknat med. Anledningen till detta är nog att jag inte räknade med att jag kunde kopiera koden ifrån de tidigare uppgifterna av någon anledning. Validering krävde också mycket mindre tid än i de andra uppgifterna då jag bara behövde kontrollera så att den angivna textsträngen inte är tom.

Jag krånglade även till det i planeringen hur jag skulle gå till väga gällande steg 5, då jag skulle testa om inputen ifrån användaren var en palindrom. Här hittade jag en mycket bättre lösning som inte krävde mycket kod alls, vilket bidrog till att jag sparade tid. Efter detta så hoppade jag till det sista steget för att skriva en funktion; isPalindrom() som kontrollerar detta, som jag sedan kunde återanvända i steg 6. Att det skulle bli såhär blev nog svårt att förutse. Men ibland så känner jag att man ser förbättringar man måste göra, men som inte passar in i något angivet steg och då kan det ju vara svårt att veta vart man ska lägga tiden. Men i ett generellt steg såsom ”Förbättring av kod” omfattar ju mycket av det arbetet.

Uppgift 3b

Strategi: Jag använder mindmap även här, och delar upp uppgiften i många steg bestående av 15 minuter för att ge mig själv utrymme för fel.



Steg	Planerad tidsåtgång	Beskrivning	Fel, gjorda av mig och problem på vägen	Verklig tidsåtgång
1	15 min	Skapa alla dokument	All gick smärtfritt	10 min
2	15 min	Fraction objekt: constructor funktion	All gick bra, några småsaker jag inte tänkt på innan, såsom en ny funktion för window.onload som ska skapa nya Fraction objekt.	12 min
3	15 min	Metoderna getNumerator(), getDenominator(), isNegative()	Det gick rätt så snabbt att definiera metoderna. Lade lite tid på viss felsökning av konstruktor parametrar på fraction objektet som jag glömt.	10 min
4	15 min	Reflektion, omprioritering av tid.	Kontrollerade att koden fungerar som den skulle, valideringsfunktionerna krånglade samt kod för att skapa ett nytt objekt av formuläret behövde kompletteras.	13 min
5	30 min	Metoderna: add(), multiply(),	Upptäcke problem att get prototype metoderna var tvugna att ligga som instansierbara metoder i fraction constuktor funktionen. Vilket jag fixade här. Fick klura lite på vad jag skulle i metoderna add() och multiply(), men tror att jag tolkade uppgiften rätt.	20 min
6	15 min	Metoderna: isEqualTo(), toString()	Gick Bra. Metoderna fungerade vid test.	8 min
7	15 min	Förbättring av kod	Förbättrade metoderna add() och multiply(), de behöver inte ha två stycken argument bara ett då this används. Kontrollerade kod, allt verkar fungera som det ska. Lade till några test som demonstrerar att metoderna fungerar. Vid ytterligare kontroll om hur man räknar ut bråk så upptäckte jag att jag gjort fel i metoderna add(). Kompletterade metoden.	53 min

Reflektioner kring uppgift 3b

Denna gång upplevde jag att tidsramarna var ganska bra ända tills jag upptäckte att bråk inte var så lätt att räkna ut som jag trodde i det sista ”Förbättring av kod steget”. Jag hade utrymmet att fixa till kodmässiga saker som behövde fixas allt eftersom enda tills jag upptäckte kallduschen i slutet. När jag för första gången skrev metoderna add() och multiply() blev jag lite fundersam, men stressade förbi eftersom jag kände att jag hade tidspress, därav missade jag att det var svårare att addera bråktal än jag trodde. En annan mindre sak jag missade att planera för var själva grundfunktionen som skapar Fraction objekt. Jag hade inte heller jag något dedikerat steg för att testa om alla metoder fungerade, vilket kändes viktigare i den här uppgiften. Men jag lyckades baka in testningen i de befintliga stegen.

Jag gjorde också en miss med att skapa get funktionerna som prototypes men insåg sedan att dessa måste ligga definierade i konstruktorfuktionen för att kunna komma åt de privata variablerna i Fraction objektet.

Men det som klart spräckte min tidsplan denna gång var missen med additionen av bråk. Jag gick 6 minuter över total planerad tid.

Uppgift 4a

Vi säger att arbetsgruppen består av följande programmerare: Kalle, Sara, Johan, Sven och Niklas. Det första steget vore att göra är att bestämma gemensamma ramar som ska finnas, eftersom gruppen kommer att jobba med olika saker.

Steg1:

Gruppen börjar med att gemensamt skriva klassinterfaces eller liknande (beroende på språk) för projektet så att de kan dela upp arbetet på ett vettigt sätt och att slutprodukten av koden som de separat kommer att skriver passar ihop och fungerar i slutändan.

Tidsåtgången för detta steg är **en vecka**.

Steg2:

Efter att arbetsgruppen har bestämt hur slutresultatet kommer att se ut genom interfaces så får nu får alla i gruppen separata arbetsuppgifter. En kritisk sak att lösa och ha insikt i är vilken molntjänst som kommer att användas för hela produkten, eller om detta drivs som en molntjänst på egna servrar. I mindmappen om två sidor ser vi hur arbetsuppgifterna är uppdelade:

- Kalle får ansvara och ta reda på vilken molntjänst som ska användas och eventuellt anpassa den så att den kommer att fungera med slutprodukten CloudPortifolio. Kalle får också ansvaret för att filsystemet fungerar; att dokumenten och mapparna fungerar på molnservern som de ska.

- Sara skriver klasserna för ändringshistoriken och färgkodningen för detta enligt de överenskomna klassinterfacen i det tidigare steget.
- Johan skriver klasserna för användarna och dess typer enligt klassinterfacen
- Sven skriver klasserna för verktygen som användarna kan använda sig av; ändra mappar eller dokument, radera mappar eller dokument, möjligheten att lämna CloudPortifolio och möjligheten att dela dokument med andra användare.
- Niklas skriver klasserna för funktionerna som verktygen använder sig av; kontroll för om alla användare är överens om fil- och mapp-borttagning samt inbjudan till delning av filer och mappar. Niklas skriver också klasser gällande rättigheterna för filer och mappar samt åtkomstkontrollista.

Detta steg genomförs med ett inplanerade möten varje vardagsmorgon för att se hur det går för respektive medarbetare och se om de behöver hjälp och råd av varandra och kan prata om problem som de stött på. Eftersom medarbetare kommer att bli klara med sina tilldelade arbetsuppgifter olika snabbt så får de som är klara med sitt hjälpa sina gruppmedlemmar att färdigställa sina uppgifter tills dess att alla är färdiga.

Tidsåtgången för detta steg är svårt att uppskatta, beroende på allt som händer. Men för att göra ett försök till en realistisk gissning så säger vi **en månad**. Hur lång tid saker och verkligen och för att hålla sig till tidsplanen är något som kan åtgärdas under mötena där medarbetarna kan prioritera om för att hinna klart i tid.

Steg3:

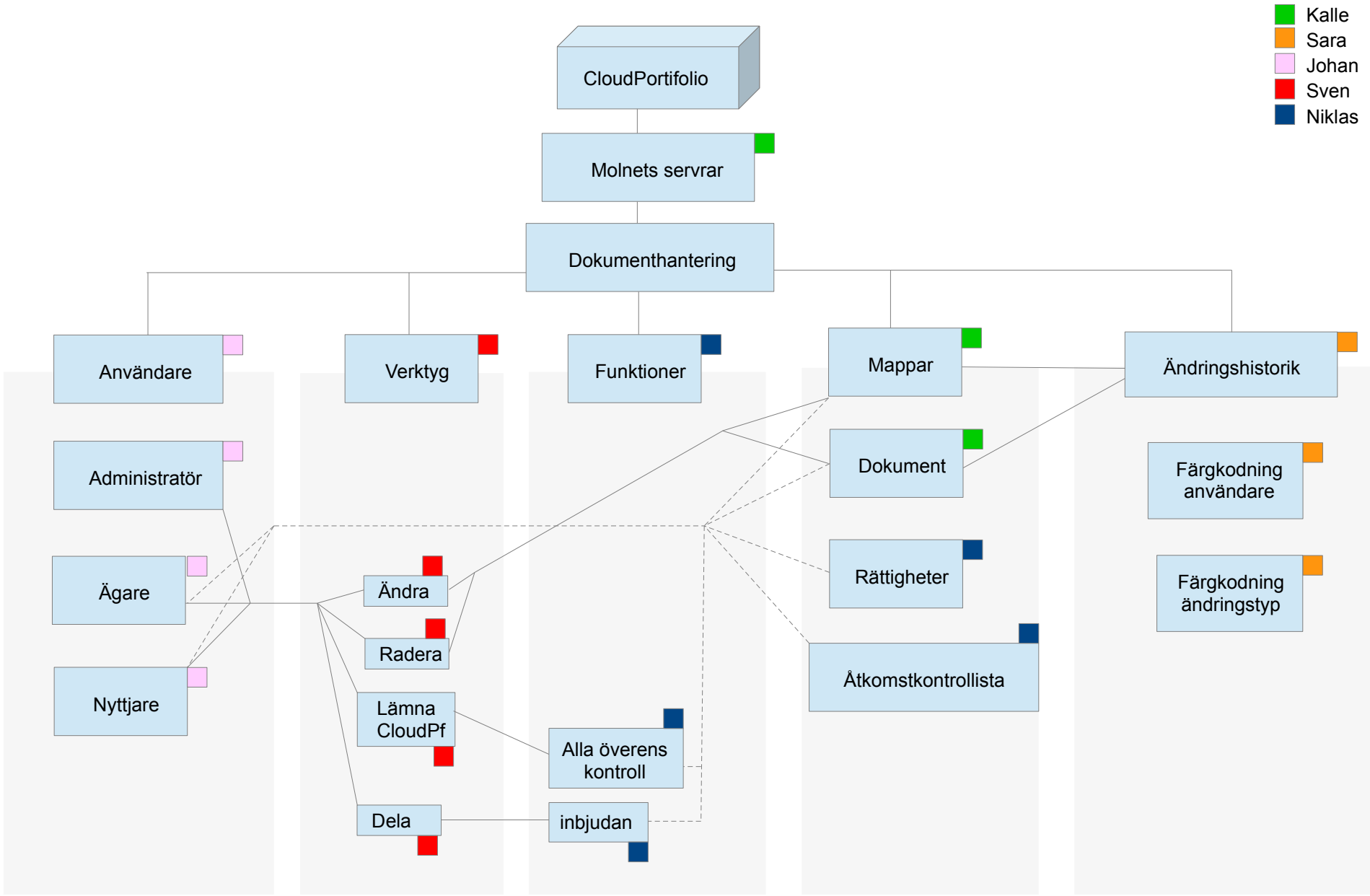
Efter att alla anser sig vara klara så är det viktigt att testa så att allt fungerar som det ska. Efter att problem upptäckts så måste dessa fixas. Det här är ett viktigt steg och är någonting som jag personligen tycker att det slarvas med i dagens it samhälle. Medarbetarna får prova sina egna klasser och funktioner så att de fungerar som de ska, varpå de får prova varandras för att till slut fokusera på hela produkten.

Det är viktigt att gruppen tidigt i detta steg utser några utomstående till betatestare för att på så sätt få in input på produkten.

Arbetsgruppen får **två veckor** på sig för detta steg, vilket kan kännas som en lång tid i förhållande till resten av projektet men det är viktigt att se till att allt fungerar som det ska.

Steg 4:

Gruppen lanserar en första version av CloudPortifolio.


Kalle
Sara
Johan
Sven
Niklas
CloudPortifolio
Molnets servrar
Dokumenthantering
Användare
Verktyg
Funktioner
Mappar
Ändringshistorik
Administratör
Dokument
Färgkodning användare
Ägare
Ändra
Rättigheter
Färgkodning ändringstyp
Radera
Nyttjare
Åtkomstkontrollista
Lämna CloudPf
Alla överens kontroll
Dela
inbjudan

Uppgift 4b

Att gå in på detaljnivå är ganska uteslutet vid planering av ett sådant här stort projekt. Man får helt enkelt planera tiden för gruppmedlemmarna att gå in på den nivån. Jag hade först tänkt att även kodmässigt bygga upp projektet med beskrivning av vilka klassnamn som ska användas, men det kändes som att projektet var för stort för att det skulle gå. En övergripande planering med ansvarsområden kändes mer aktuell.

Det svåraste att planera att sådant här projekt är att veta hur lång tid saker och ting tar, desto längre i framtiden de ligger. Det är ganska lätt att bestämma sig för att en rimligt tidsåtgång på det första steget är en vecka. Men vad som helst kan i princip hända i steg 2 som gör så att den första planeringen spricker. Ifall Kalle i sin upptäckt kommer på att den valda molnplattformen plötsligt inte fungerar så får alla tänka om och ta fram en ny sorts krisplan på ett möte eller i värsta fall lägga ner projektet. Av den anledningen så känns det viktigt att skapa utrymme. Nu har jag inte skapat så mycket utrymme och fria tyglar i min planbeskrivning, men Detta är något som skulle kunna ge lite mera andrum om det väl går fel utan att de som investerat pengar i projektet anser sig bli svikna. Det är bra om man så gått det går är en realist, gör en så realistisk plan som möjligt och ovanpå det har en buffert med tid.

En annan svårighet jag upptäckte i min plan jag gjort var att gruppmedlemmarna kommer att bli klara med sina uppgifter vid olika tidpunkter. Det skulle kunna bidra till att vissa medlemmar inte kan prova om deras kod fungerar eftersom de är beroende av annan kod för det som inte är färdig än, vilket i sin tur kan leda till buggar. Men med tanke på att det första som görs är att interfaces utarbetas, så borde det inte vara något problem. Så när en gruppmedlem är klar med sitt så är det nog bättre om denna hjälper resten av gruppen att bli klar med sitt. Det kan ju också vara så att en gruppmedlems arbetsuppgifter visar sig vara mycket större än någon annans och fördelningen blir orättvis. I sådana fall så får gruppen komma överens om omfördelning av arbete på ett morgonmöte.

En ytterligare risk i processen är om det visar sig dyka upp många fel under steg 3, kanske för att man i de tidigare stegen inte hunnit prova funktionaliteten hos de färdigställda klasserna. Detta kanske leder till att en hel del kod måste skrivas om och att projektet fördröjs.

Den största risken i hela projektplaneringen är dock ifall en eller flera medarbetare är frånvarande till exempel p.g.a. sjukdom. Eftersom det är så få inblandade i projektet så är konsekvenserna stora. Till detta kan jag inte komma på något bra försvar förutom att skapa en större tidsbuffert.