LINKÖPINGS UNIVERSITET

TDP003 Projekt: Egna datormiljön

Systemspecifikation av portfoliosystemet

Kursmaterial till kursen TDP003

Höstterminen 2017
Version 2.2

2017-06-30

Innehåll

1 Revisionshistorik

Ver.	Revisionsbeskrivning	Datum
1.1	Smärre språkliga korrigeringar. Numreringen av datakrav och icke-funktionella kraven korrigerades. Nytt krav 3.7. Utvidgning av kapitel 7 med en beskrivning av hur utvärderingen av systemen ska göras.	2013-08-26
1.2	Smärre språkliga korrigeringar. Tagit bort delar om KID.	2013-08-26
1.3	Ändrat överlämningsförförandet, numera enbart till assistent.	2014-08-26
2.0	Ändrat CSV till JSON som lagringsformat.	2014-09-03
2.1	Fortsatt uppdaterat layout. Renskrivning. Uppdelning av krav samt borttagning av detaljbeskrivning i dokumentationskrav.	2014-09-08
2.2	Översyn av krav och övergripande formattering.	2017-06-30
2.3	Förtydligande av skillnaden mellan *systemdokumentation* och *testdokumentation*	2017-10-09

2 Översikt

I portfolioprojektet ska du arbeta utifrån en given specifikation av systemet. Dokumentationen följer gängse traditionell standard för hur man specificerar ett system, om än i liten skala. Du kommer att komplettera denna dokumentation med detaljer som rör implementation och användning.

2.1 Deldokument

Kravspecifikation: Här beskrivs de krav som finns på hur systemet ska fungera sett ur ett användar- och sjösättningsperspektiv.

Arbetsprocess: Här står vilken arbetsordning som ska följas och vilka *leverabler*.

Arkitektur: Beskriver arkitekturen för portfoliosystemet.

Systemdesign: Preciserar detaljer om programgränssnitt till delsystemet *data*.

Testning: Kort om hur systemtestet ska utföras.

3 Kravspecifikation

Med en kravspecifikation menar vi en sammanställning av de krav som finns på hur systemet ska fungera (funktionella krav) samt övriga krav på system och projekt (icke-funktionella krav). Kravspecifikationen för portfoliosystemet är givet av kursledningen. Att omsätta denna kravlista till ett fungerande system är studentens arbete. I de fall kraven inte säger uttryckligen vad som gäller finns en frihet för studenterna att själva välja design/teknik/implementationsstil. Detta gäller till exempel exakt hur användargränssnittet ska se ut. Bara vissa grundläggande krav är ställda.

Ett av målen för projektkursen är att bygga ert eget portfoliosystem där ni skapar en webbplats som ni under utbildningens gång kan lägga in projekt ni arbetar med, både inom och utanför utbildningens ramar. På vägen lär ni er om användbara webbtekniker så som HTML5 och CSS. Programmeringsmässigt handlar det främst om att lära sig Python.

Portfoliosystemet är en webbplats med en statisk och tre dynamiskt skapade webbsidor, enligt följande lista:

- `index`: statisk eller dynamisk förstasida
- `list`: dynamisk sida som listar projekten

- `project:` dynamisk sida som visar information för ett specifikt projekt
- `techniques:` dynamisk sida som visar en sammaställning över de tekniker som används i projekten

Nedan följer en lista med specifika krav som ställs på portfoliosystemet. Kraven är numrerade i `x.y` där `x` är siffran på underkapitlet och `y` är ett löpnummer för kraven under rubrik `x`.

3.1 Presentation

ID	Krav	Tillagd	Ändrad
1.1	Förstasida med bilder. URL: `/`	2013-08-26	2014-09-08
1.2	Söksida som visar en lista över projekt med kort information om varje projekt och som gör det möjligt att sortera dessa, samt söka bland dem genom ett formulär på sidan. Url: `/list`	2013-08-26	
1.3	Projektsida som visar fullständig information om ett projekt. GET-variabel för att ange projekt-id: `id` URL: `/project/id` - där *id* är projektets nummer.	2013-08-26	2014-09-08
1.4	Tekniksida som visar information om alla projekt utifrån använda tekniker. URL: `/techniques`	2013-08-26	2014-09-08
1.5	För varje projekt ska en liten bild visas på söksidan och en stor på projektsidan. Det behöver inte vara samma bild. Bildtext för varje bild skall finnas.	2013-08-26	2014-09-08
1.6	Vid fel ska systemet skriva ut informativa meddelanden till användaren på en lämplig nivå för en slutanvändare. (Det vill säga, systemet ska fånga och omvandla felkoder och statuskoder till begripliga meddelanden.)	2013-08-26	
1.7	När en användare försöker visa ett projekt som inte finns, ska korrekt statuskod returneras (dvs. 404).	2017-06-30	

3.2 Data

ID	Krav	Tillagd	Ändrad
2.1	Systemet ska kunna hantera följande information om ett projekt: projektnamn, projekt-id-nummer, startdatum, slutdatum, kurskod, kursnamn, kurspoäng, använda tekniker, kort beskrivning, lång beskrivning, liten och stor bild, gruppstorlek och en länk projektsida. Projektnamn och projekt-id är obligatoriska, övriga fält kan lämnas tomma.	2013-08-26	
2.2	Projekt-id ska vara ett unikt heltal för varje projekt.	2013-08-26	
2.3	Varje projekt kan ha en sekvens av tekniker angivna.	2013-08-26	
2.4	Sökning ska kunna göras på godtycklig projektinformation. Sökning kan ske på flera fält samtidigt. Sortering ska kunna göras på ett fält, i stigande och fallande träffordning. Man ska kunna filtrera utifrån använda tekniker i sökningen. Observera att allt ska fungera tillsammans, så att man kan söka på ett sökord, filtrera till vissa tekniker och sortera söklistan i en viss ordning samtidigt.	2013-08-26	
2.5	~~Loggning av varje anrop ska ske till en loggfil. Loggen ska ange tidpunkt, vilken typ av anrop det är och relevant information till exempel data i sökning, vilket projekt som slås upp etc.~~ Utgått.	2013-08-26	2017-06-30
2.6	~~Systemet ska hantera statuskoder som skickas till presentationslagret och där vid behov översätts till lämpliga meddelanden. Tre statuskoder används: 0 = ok, 1 fel vid access av databasfil, 2=angivet projekt-id existerar inte.~~ Utgått.	2013-08-26	2017-06-30
2.7	Data lagras i *JavaScript Object Notation* (JSON) i filen `data.json`. Filen ska lagras med UTF-8 teckenkodning.	2013-08-26	2014-09-08
2.8	Data läggs till i JSON-filer manuellt (eller av andra verktyg) i systemet.	2013-08-26	2014-09-08
2.9	Förändring av `data.json` ska slå igenom direkt i systemet utan omstart av webbserver.	2013-08-26	2014-09-08
2.10	(Frivilligt) Utvidga systemet med en administrativ sida för redigering av data.	2013-08-26	2014-09-08

3.3 Icke-funktionella krav

ID	Krav	Tillagd	Ändrad
3.1	Pythonskriptens utdata ska formateras med en HTML-mall via Jinja2.	2013-08-26	2014-09-08
3.2	Presentationen ska implementeras med hjälp av HTML5 och CSS3.	2013-08-26	2014-09-08

3.3	Katalogstrukturen ska se ut som följer:
	`MyPortfolio/`
	`  doc/`
	`  static/`
	`  images/`
	`  *.png, *.jpg, *.gif`
	`  style/`
	`  *.css, *.png, *.jpg, *.gif`
	`  templates/`
	`  *html, *.xml, *.json`
	`  README`
	`  data.json`
	`  myFlaskProject.py`
	`  *.py`
	Viktigt: Katalogen `style/` är till för CSS-filer och bilder som refereras från dessa. Bilder som hör till innehållet/projekten läggs i `images/`.
		2013-08-26	2014-09-08

3.4	Versionshantering med Git ska användas.	2013-08-26	
3.5	Hela systemet ska testas av tredje person som får utföra de huvudsakliga uppgifterna som portfoliosystemet är tänkt för. Se vidare under avsnitt 7.	2013-08-26	2014-09-08
3.6	Källkoden ska kommenteras på engelska för varje modul, funktion och för varje global variabel. Ej självförklarande kodavsnitt ska även kommenteras löpande i koden.	2013-08-26	
3.7	Alla namn på filer, moduler, funktioner och variabler ska vara på engelska.	2013-08-26	
3.8	En systemdokumentation ska ingå i systemets dokumentation. En `README`-fil ska beskriva hur systemet är paketerat och peka ut övrig dokumentation. Tydliga installationsinstruktioner ska ingå, antingen som en del av `README`-filen eller i valfri annan fil utpekad av `README`-filen.	2013-08-26	2014-09-08 2017-10-09
3.9	En användarmanual ska ingå i systemets dokumentation.	2014-09-08	2017-10-09
3.10	Testerna ska dokumenteras och skall ingå i systemets dokumentation.	2014-09-08	2017-10-09

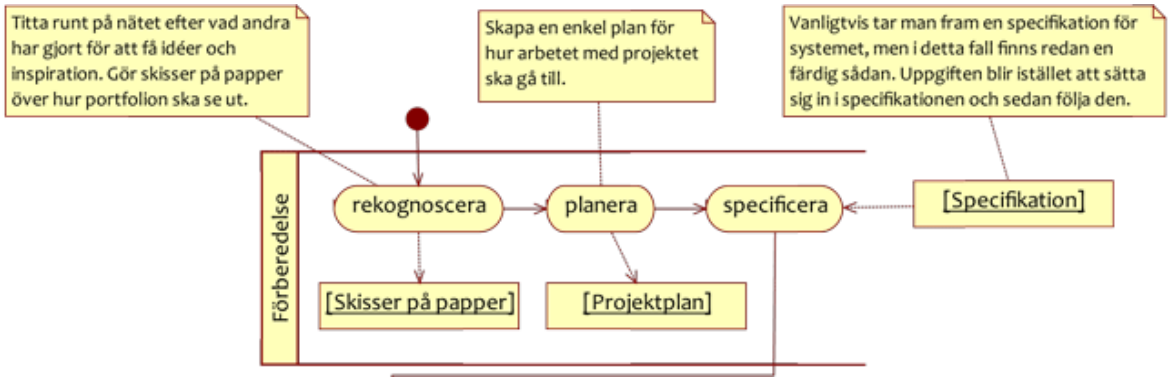
4 Arbetsprocess

Portfolioprojektet följer en på förhand fastställd arbetsgång med olika moment som beskrivs på denna sida. Du ska utföra detta arbete främst under egen tid men viss resurstid finns inlagd i kursschemat. Börja med att läsa igenom materialet och förbereda dig inför första laborationen. Arbetet med portfolioprojektet sker i följande tre faser:

Att förbereda handlar om att se till att man vet vad som ska göras och vilka förutsättningar som existerar. Med konstruera avses såväl att utforma applikationen som att testa, implementera och dokumentera den.

Projektet avslutas genom att den färdiga och paketerade applikationen överlämnas till laborationsassistenter.

4.1 Förbered



I detta skede ska ni skapa bra förutsättningar för resten av projektet genom att ta reda på saker och planera. Istället för att själva ta reda på kundens behov och önskemål får ni en färdig specifikation som ska följas (se kravspecifikationen).

För att få idéer och inspiration får ni själva söka upp möjliga förebilder och inspirerande exempel på nätet. Efter detta ska ni ta fram handritade skisser på papper av hur ert eget portfoliosystem ska se ut. Att arbeta med papper och penna för att skapa prototyper av system är något som används flitigt av flera mjukvaruföretag. Intresset för detta arbetssätt har ökat starkt de senaste åren, i samband med webbutveckling och webbapplikationer.

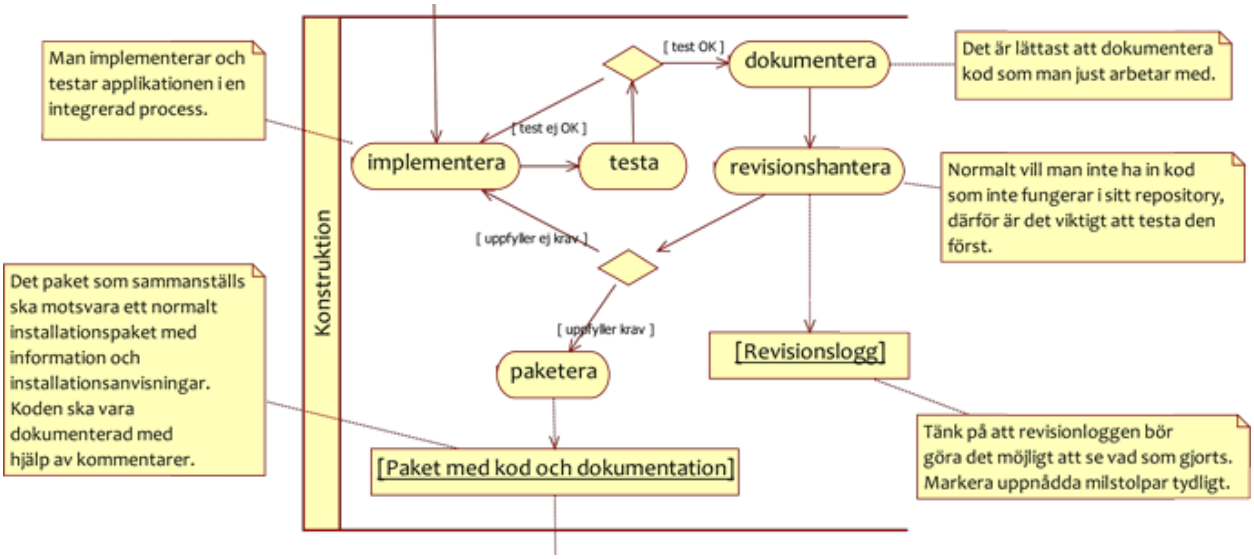
Förutom att veta *vad* ni ska göra behöver ni tänka igenom *hur* ni vill göra det och organisera ert arbete därefter. För att få kvalitet i detta moment finns en enkel projektplan med som inlämningsuppgift. Kom ihåg att ta med sådant som projektets tidsramar och resurser (till exempel arbetstimmar till förfogande för projektet) samt projektets målsättning. Men även att sätta upp era egna tidsramar och milstolpar.

Att lämna in:

- Skisser av användargränssnittet (LoFi-prototyp).
- Projektplan

4.2 Konstruera

Under konstruktionsfasen utformas och implementeras portfoliosystemet enligt följande bild:



Ni behöver identifiera lagom delmoment att implementera i taget. Ett kriterium kan vara att systemet ska klara godkänt på ytterligare en test. Genom att köra tester säkerställer ni att koden gör det den ska.

Att dokumentera koden blir svårare ju längre tid som gått sedan den skrevs. Vill ni som utvecklare lägga så lite tid som möjligt på att dokumentera tjänar ni tid på att göra det så snart en funktion är implementerad och testad.

En grundregel när man har koden i ett *repository* är att all kod som finns i dess huvudgren (eng. *branch*) ska vara testad med godkänt resultat. Annars stör man andra utvecklare, till exempel genom att de inte längre kan köra systemet. Om projektet tillämpar *continuous integration* (vilket ni kommer att göra i framtida projekt) så blir detta ett måste! I portfolioprojektet är ni två personer i en grupp och det viktiga är att ni bägge vet vad som pågår. I större projekt på IP-programmet kommer det krävas att ni lär er att tillämpa revisionshantering på ett mer avancerat sätt.

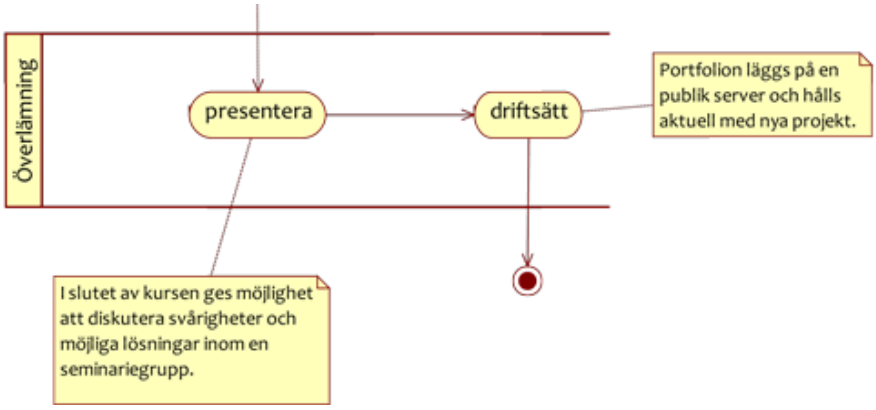
En annan viktig aspekt av revisionshantering är att man skriver bra kommentarer vid varje *commit*. I dessa finns källkodens historik. En anledning är att man helt enkelt ska veta vad man har gjort och i vilken ordning. Denna information kan utgöra grunden för en revisionslogg (eng. *changelog*).

Att lämna in:

- Revisionslogg (via Git)

4.3 Överlämna

Kursen avslutas med att presentera och sjösätta portfoliosystemet enligt följande bild:



Inför kursens avrundning lämnas paketet med kod och dokumentation in till er laborationsassistent. Kom ihåg att paketet ska innehålla information om hur det är strukturerat samt installationsanvisningar och andra dokument. Ladda gärna ner ett eller flera *open source*-projekt för att få exempel på hur det kan se ut.

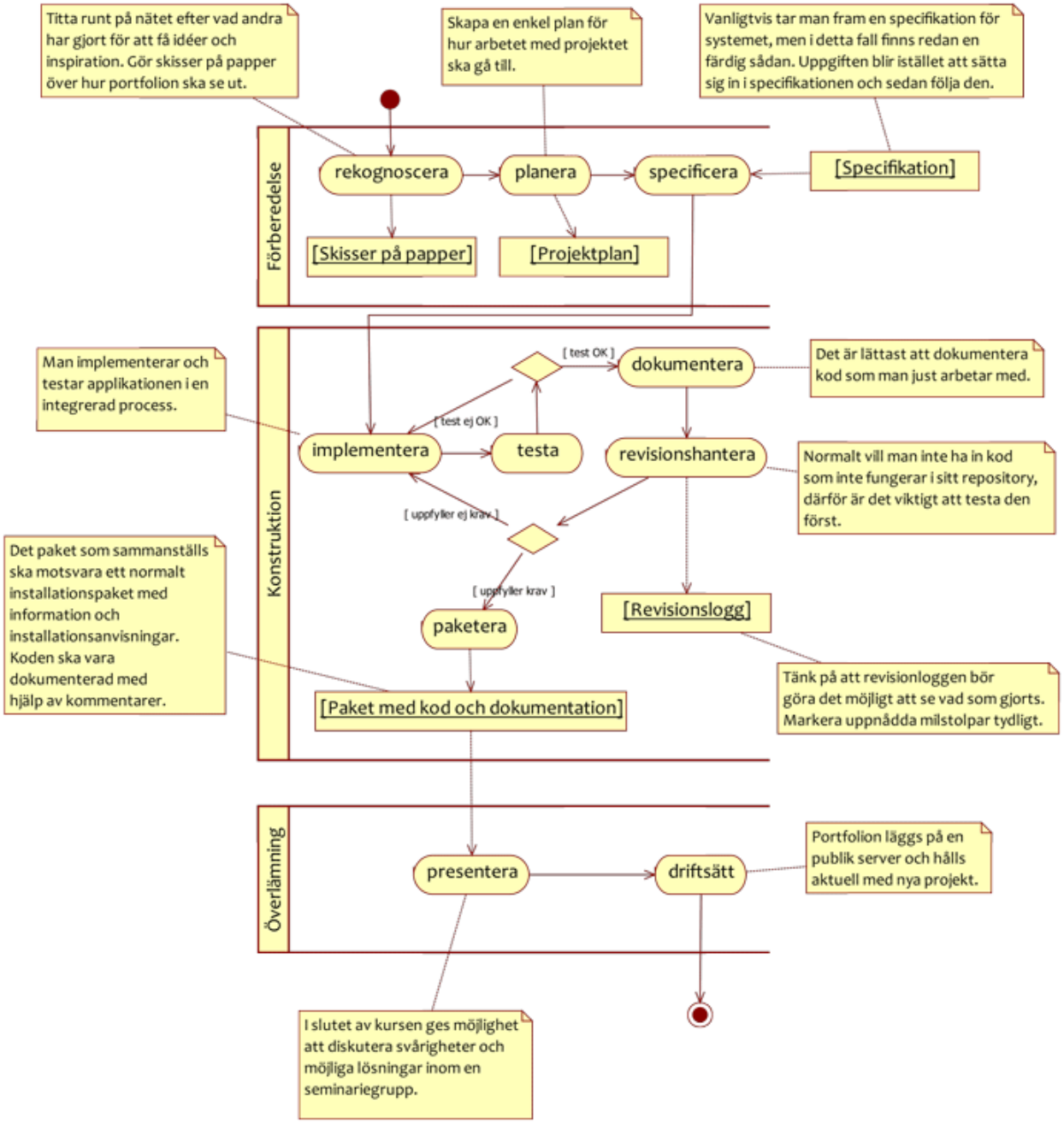
Som förberedelse inför slutseminariumet kommer ni att få källkoden från en annan grupp. Gå igenom källkoden och lär av deras lösningar på olika svårigheter i projektet samt se över vad ni uppfattar som lätt eller svårt att förstå av deras källkod. Ni ska också kunna förklara för andra hur ni tänkt i olika delar av ert eget system. Slutseminariet kommer att hållas med hela klassen.

Att lämna in:

- Revisionslogg (via Git)
- Paket med (kommenterad) kod och dokumentation (enligt överenskommelse med er handledare)

4.4 Hela processen

Här visas en bild över hela projektförloppet i sammanfattning:



Diagrammet är ett aktivitetsdiagram enligt UML, se *Introduction to Activity Diagrams* `http://www.agilemodeling.com/artifacts/activityDiagram.htm` eller *UML2 Tutorial* `http://sparxsystems.com.au/resources/uml2_tutorial/uml2_activitydiagram.html` om du vill veta mer.

5 Arkitektur

När vi talar om ett system övergripande kallar vi detta för systemarkitektur. Denna sida tar upp de krav som ställs på arkitekturen i ert portfoliosystem.

Portfoliosystemet ska byggas i termer av två delsystem. Med ett delsystem menar vi rent konkret en eller flera python-moduler som tillsammans bildar en avgränsad del av systemet. Dessa delsystem kommer kommunicera med varandra i enlighet med förutbestämda format. Därmed uppnår vi en utbytbarhet för varje delsystem. Detta kommer bland annat göra att ni kan byta till exempel användargränssnitt med andra grupper vid projektets slut. Underrubrikerna förklarar varje delsystems enskilda uppgift.

5.1 Delsystem Presentation

Delsystemet Presentation ansvarar för gränssnittet mot användaren och att ta emot data som kommer från användaren. Här hanteras även funktionalitet som har direkt att göra med den grafiska presentionen för användaren. Till exempel kan en tom lista visas som ”Inga sökresultat för ... finns att visa!” istället för att det bara blir tomt. Ett annat exempel på sådan funktionalitet kan vara att ge raderna i en tabell alternerande bakgrundsfärger.

Att ta emot data från användaren är det här delsystemets ansvar, vilket kan inkludera även en första filtrering av indata. Kom ihåg att på webben är all data som kommer från användare potentiellt en säkerhetsrisk. Därmed måste delsystemet Presentation tänka på säkerhet mot oönskade intrång.

5.2 Delsystem Data

Delsystemet Data ansvarar för hur data som rör projekten lagras och hämtas. Delsystemet Data erbjuder delsystemet Presentation ett antal funktioner och tar hand om allt som rör konkret datarepresentation på fil, det vill säga läsning och ”hållning” av data under exekvering. Till delsystemet Data räknas även funktionalitet för sökning av data och sortering av hittade resultat.

5.3 Kommunikation mellan delsystem

De två delsystemen kommunicerar i sekvens Presentation till Data vid inkommande användarförfrågan (till exempel att användaren klickar på en länk). Resultatet går sedan åt andra hållet, Data till Presentation, vilket till sist bör resultera i att en ny HTML-sida visas med resultatet.

Genom att bygga portfoliosystemet i två separerade delsystem så kan det ena delsystemet bytas ut utan att det påverkar det andra. En situation där det här är användbart är då information önskas presenteras i ett annat format, som ren text eller PDF. Då behövs endast presentationslagret bytas ut. Denna arkitektur kan ses som en enkel variant av en vanlig idé som används även för större system, då ofta kallad lager-arkitektur (som för större system består av fler lager än två). Se kravspecifikationen angående krav på implementationen av denna arkitektur. Till exempel vilka filnamn olika delar av systemet ska ha.

6 Systemdesign

Här tar vi upp de krav som ställs på systemdesignen av portfoliosystemet. Med systemdesign menar man normalt saker som rör källkodens utseende men utan att precisera implementationen exakt.

6.1 Programgränssnitt

För portfoliosystemet har vi ett obligatoriska krav ställt på systemdesignen: programgränssnitten till Datamodulen ska följa en fastställd standard. Det här gör vi för att göra delsystemen utbytbara. Programgränssnitt kallas ofta *API* (Application Program Interfaces). Vi kommer hädanefter att använda den termen synonymt med programgränssnitt.

Portfoliosystemets API ligger i sin helhet på `http://www.ida.liu.se/~TDP003/current/portfolio api`

Dokumentation är genererad med hjälp av *epydoc* `http://epydoc.sourceforge.net/` och visar vilka funktioner som modulen `data.py` ska innehålla. Dokumentationen visar också vilka argument och vilket returvärde varje funktion ska ha. I beskrivningen framgår också hur dessa argument kan sättas.

Observera att modulerna får innehålla fler funktioner men att bara API:ets funktioner får användas av andra delsystem. Det är också tillåtet och uppmuntras att fler moduler introduceras för varje delsystem vid behov.

7 Testning

Ditt portfoliosystem ska testas av tredje person. Dokumentera hur systemtestet genomfördes och vilka problem och fel som upptäcktes. Denna dokumentation ska ingå i systemets dokumentation. Minimum är att två personer, externa för ert projekt, testar systemet. Låt användarna först testa systemet på egen hand, utan att ni hjälper dem alls, och observera beteendet. Guida sedan användarna genom systemet och se till att de provar alla funktioner. Samtliga funna felaktiga beteenden eller systemkraschar ska åtgärdas i den slutliga versionen av systemet. Kom ihåg att det är er uppgift att dokumentera den testning tredje person gör.