Reflektionsrapport PA1435

Mikael Svahnberg*

2016-12-19

1 Introduktion

Kursen *PA1435, Objektorienterad Design* omfattar 6hp och är en av i nuläget fyra kurser som på olika vis berör samma ämnesområde. De olika kurserna är listade i Tabell 1. Totalt sett rör det sig om mellan 180 till 200 studenter som går de olika kurserna.

Kursmålen (Se Tabell 2 beskriver en viss evolution, där PA1415 har de äldsta målen (senast reviderade 2014, ej i det "nya" formatet) och PA1443 har de nyaste. Det är heller inte ett fullständigt överlapp mellan de ämnen som skall tas upp i varje kurs. För att förstå överlappet behövs en kortfattad genomgång av ämnet.

1.1 En Kortfattad Genomgång av Ämnet

Kärnan i kursen är ett arbetssätt med notationsstöd för att skapa mjukvarudesigner. Detta arbetssätt är i princip en miniatyr-version av Conceive-Design-Implement -tankarna. Detta designarbetssätt passas in i ett utvecklingsprojekt, där man lägger ytterligare tonvikt på kravinsamling, projektplanering, och – såklart – implementation och testning. När systemen blir större behöver man strukturera det på olika nivåer, både den mikronivå som är kärnan i kursen, men också på en "makronivå", eller en arkitekturnivå. Genomgående för alla nivåer är att man det finns inte ett enda rätt eller fel, utan det finns ett antal goda egenskaper man förväntar sig av ett system (där kanske det främsta är att man enkelt skall kunna underhålla systemet), och ett antal principer för konstruktionen som leder till en design eller arkitektur med de förväntade egenskaperna. Att lära ut notationen och arbetssättet är relativt enkelt – det är "bara" för studenterna att härma [1]. Att förstå konsekvenserna av att göra designen på ett visst sätt, eller varför ett visst analys-steg ens är nödvändigt, kräver en djupare förståelse för dessa grundläggande designprinciper.

*Mikael.Svahnberg@bth.se

[1] Se *Cargo-Culting*, t.ex. `https://en.wikipedia.org/wiki/Cargo_cult`

Table 1: Relaterade Kurser

Kurskod	Namn	Omfattning	Program
PA1415	Programvarudesign	7.5	BSc i Spelprogrammering, BSc i IT-Säkerhet
PA1434	Grunder i Objektorienterad Design	4	CI i IndEk
PA1435	Objektorienterad Design	6	CI i Spel- och Programvaruteknik, CI i Industriell Ekonomi
PA1443	Introduktion till Programvarudesign och Arkitektur	5	BSc i Programvaruteknik

Table 2: Kursmålen för de nuvarande kurserna

Kurskod	Kursmål
PA1415	- på en grundläggande nivå i grupp kunna ta fram krav på en programvara och uttrycka dem i en kravspecifikation
PA1415	- i grupp producera en översiktlig utvecklingsprojektplan baserat på en kravspecifikation
PA1415 (PA1435/PA1434)	- i grupp kunna skapa en detaljerad objektorienterad design för ett mjukvaruprogram
PA1415	- i grupp kunna implementera ett mjukvaruprogram inom rimlig tid, baserat på en kravspecifikation och en objektorienterad design
PA1415 (PA1443)	- på en grundläggande nivå i grupp kunna planera och genomföra testning av producerad programvara, baserat på en kravspecifikation
PA1415 (PA1435/PA1434)	- skapa och analysera objektorienterade artefakter uttryckta i UML
PA1415 (PA1435/PA1434/ PA1443)	- kunna motivera och använda designmönster i utvecklingen av mjukvarusystem
	Kunskap och förståelse
PA1435/PA1434/ PA1443	- kunna visa förståelse för grundläggande principer i objektorienterad programvaruutveckling.
PA1435/PA1434	- kunna visa förståelse för UML som modelleringsspråk.
PA1435/PA1434/ PA1443	- kunna visa kunskap om grundläggande designprinciper.
PA1435/PA1434/ PA1443	- kunna visa kunskap om grundläggande designmönster.
PA1443	- kunna visa kunskap om grundläggande mjukvaruarkitekturstilar
	Färdigheter och förmåga
PA1435/PA1434/ PA1443	- kunna uttrycka strukturen och beteendet hos ett system i termer av objektorienterade koncept.
PA1435/PA1434	- kunna korrekt använda UML för att uttrycka struktur och beteende hos ett system.
PA1435/PA1434	- kunna korrekt transformera en objektorienterad design till källkod.
PA1435/PA1434	- kunna tillämpa designprinciper och designmönster i allmänhet och inom en särskild domän.
PA1443	- kunna tillämpa grundläggande designmönster i en objektorienterad design.
PA1443	- kunna skapa en objektorienterad design för ett system enligt goda objektorienterade designprinciper
PA1443	- kunna tillämpa grundläggande arkitekturstilar för ett mjukvarusystem
PA1443	- kunna resonera om de kvalitetsegenskaper ett system med en viss arkitekturstil har eller bör ha
PA1443	- kunna resonera om och skapa en grundläggande testplan för ett objektorienterat system
	Värderingsförmåga och förhållningssätt
PA1435/ PA1434	- kunna analysera källkod för eventuella förbättringar.
PA1435/PA1434/ PA1443	- kunna analysera och kritiskt diskutera en design för eventuella förbättringar.

Detta skall då sys ihop i olika kurser, med något olika fokus. Grovt kan man indela ämnet i *Kravinsamling, Utvecklingsmetodiker, Projektplanering, Objektorienterad Analys,*

Table 3: Berörda delämnen per kurs

Delämne	PA1415	**PA1434**	**PA1435**	PA1443
Projektplanering	Ja			
Utvecklingsmetodiker	delvis	delvis	delvis	
Kravinsamling	Ja	delvis	delvis	
Objektorienterad Analys	Ja	Ja	Ja	delvis
Objektorienterad Design	Ja	Ja	Ja	Ja
Överföring av Design till Implementation	Ja	delvis	Ja	delvis
Mjukvaruarkitekturer	delvis			Ja
Testning	Ja	delvis	delvis	Ja

Objektorienterad Design, Överföring av Design till Implementation, Mjukvaruarkitekturer, och *Testning* – där mycket av detta syftar till att också få ett underhållbart system. Tabell 3 sammanfattar hur jag valt att sprida dessa ämnen över de olika kurserna. Notera att samtliga delämnen (förutom kärnämnena i detta kurspaket, markerade med fetstil i tabellen) också har en eller flera egna kurser á minst 7.5hp – många av dem på avancerad nivå.

2 Nuläge

Nuläget är inte helt enkelt att beskriva eftersom jag nyligen tog över kursen av en pensionerad kollega. Jag var inblandad i föreläsningarna den senaste omgången, och håller nu på att arbeta om kurserna för att bättre ta hand om de olika perspektiven och nytillkommna kurser. Jag väljer därför att beskriva hur kurserna kommer se ut när de ges till våren, eftersom det är den mest relevanta bilden.

"Drick din egen Champagne" är ett uttryck som jag försöker tillämpa i den här kursen. Jag har tagit intryck av moderna utvecklingsmetodiker inom Agile/Lean, och Scrum i synnerhet, och har delat in kursen i en mängd små block, så kallade *sprintar*, som fokuserar på ett mindre delämne (mindre ändå än de som är listade i Tabell 3). Dessa sprintar har för avsikt att:

- Ge en kortfattad textbaserad introduktion till delämnet.
- Ge anvisningar om vad studenten förväntas läsa för att sätta sig in i delämnet. Detta innefattar kapitel i kursboken, andra relevanta böcker, onlinematerial, och ibland forskningsartiklar.
- Erbjuda ett antal videoföreläsningar där de rent mekaniska greppen (t.ex. notationer) gås igenom.
- Erbjuda ett antal övningar för självstudier
- Erbjuda ett antal uppgifter på ett större exempel (för senare examination).

Till varje sprint är även kopplat en eller fler föreläsningar, där fokus är på att – tillsammans med studenterna – diskutera hur man gör en *bra* design, dvs. hur man tillämpar de genomgånga designprinciperna för att skapa en hållbar design/arkitektur med de egenskaper man önskar. Det är alltså en blandform mellan klassiska föreläsningar och seminarier.

Ett större exempel löper genom hela kursen, med ett för studenterna relevant mjukvarupronblem. Studenterna förväntas med moderna verktyg genomföra alla moment som lärs ut. Med jämna mellanrum finns så kallade *release-sprintar* där de får paketera ihop sina lösningar till en leverabel som sedan används för examination och betygssättning. Kopplat till detta större exempel finns övningstillfällen i mindre grupper, där studenterna får diskutera uppgifterna och sina lösningar.

Jag har medvetet valt att låna begrepp från mjukvaruutvecklingsdomänen för att ytterligare hammra in modern utvecklingsmetodik. Varje sprint har alltså ett antal *user stories* (Agil-världens motsvarighet till krav) som är skrivna ur perspektiv av de yrkesroller som är berörda av det som avhandlas i sprinten. Vidare finns det *acceptanskriterier* för varje uppgift så att studenterna själva kan göra en bedömning av om de löst uppgiften eller inte. En *testplan* utvecklas kontinuerligt av studenterna genom sprintarna där de för sin egen del beskriver hur de kan testa att de lärt sig det de förväntas i sprinten. Studenterna förväntas också underhålla en *kurs-backlog* (en backlog är en lista med prioriterade user stories), där de kan lägga in frågor de behöver få besvarade, tankar som de vill följa upp, eller fudneringar på hur man kan använda det de lärt sig i ett vidare sammanhang. Idén är att när kursen är slut så innehåller backloggen ett antal pekare framåt för hur studenterna skall använda det de lärt sig i andra kurser och i sitt framtida yrkesliv. Slutligen har varje sprint ett *acceptanstest* där studenterna enkelt kan checka av att de gjort allt de skall i sprinten.

Kursen i dess tidigare utformning var väldigt akademisk, och som motvikt så har jag omformat den baserat på ett stort antal diskussioner med yrkesverksamma från hela världen. Vi har inte diskuterat just denna kursen, utan mer hållt allmänna diskussioner om modern programvaruutveckling, och vilka kunskaper och värderingar som de önskar se i bra programvaruutvecklare. Denna rörelse är också i enighet med vad jag hört att studenterna efterfrågar, så jag skulle vilja påstå att kursen är baserad på (om än inte granskad eller validerad av) kunskaper och önskemål från såväl lärare, studenter, och näringslivsrepresentanter. Jag tror det är svårt att få en kurs där både lärare och näringslivsrepresentanter är nöjda, då forskningsområdet drar starkt åt ett håll (mer formalism), och näringslivet drar lika starkt åt det andra hållet (mer pragmatism). Tyvärr är denna skism ganska djup och oöverstigbar. Min lösning är att ge studenterna ett pragmatiskt verktyg men med översiktlig kunskap om den av akademin så omhuldade formalismen.

3 Svagheter, Utvecklingspotential

Svagheter som jag kan se i nuläget är att många av de inblandade utbildningsprogrammen har en förväntan att alla kurser skall handla om deras särintresse. Jag kommer inte hinna möta denna förväntan under VT2017, utan jag kommer ge en generisk bild där jag kryddar friskt med exempel, problemställningar, och anekdoter från de olika särintressena. Över tid planerar jag att göra en eller flera sprintar om varje särintresse som kan bidra med "rätt" perspektiv till den generiska bild som erbjuds i kärn-sprintarna.

En annan utmaning är det stora gapet mellan forskningsfronten och nuvarande industripraxis. Industripraxis är dessutom synnerligen pragmatiskt och rudimentärt när det gäller programvarudesign. Här har jag valt sida och hävdar att det är viktigare att först förstå hur man kan använda teorierna i praktiken och vilka begränsningar man kommer stöta på, snarare än den abstrakta teoribildningen. Mitt bidrag är måhända att jag samtidigt erbjuder en verktygslåda som är mer mångsidig än den som i nuläget används av industrin, med ledtrådar till när respektive verktyg är lämpligt att använda eller (inte minst viktigt) inte användas. Samtidigt erbjuder jag en viss översikt över forskningsfronten också. Tyvärr är, som redan antytts, denna forskningsfront så långt ifrån någonting praktiskt användbart att studenterna sällan inser nyttan med den.

Det är många utbildningsprogram inblandade i de olika kurserna, och det är därför en utmaning att synkronisera och integrera innehållet med övriga kurser på deras respektive program. Jag ser det här som ett ständigt pågående arbete att föra diskussioner med de olika programansvariga för att åstadkomma denna integration. I nuläget har dock fokus varit på att få en översikt över materialet och genomförandet, så jag har främst vilat på historiska lagrar.

4 Utvecklingsförslag och mål i förhållande till utbildningsplan och CDIO syllabus

Jag har ett antal olika mål med hur jag genomför kurserna i programvarudesign:

- Studenterna skall främst förstå vad som är en bra design, och vad man skall göra för att åstadkomma en bra design.
- Studenterna skall möta en blandning av realistiska problem – både stora och små – där man tydligt kan resonera om styrkor och svagheter med olika lösningar.
- Studenterna skall skaffa sig en bred och mångsidig verktygslåda för programvarudesign, med en ingående förståelse för syftet med varje verktyg.
- Studenterna skall skaffa sig en översiktlig förståelse för forskningsfronten inom ämnet och av skilnaderna mellan forskning och produktion.
- Studenterna skall använda modern teknologi och moderna arbetssätt när de arbetar i kursen.
- Kursen skall använda sig av modern teknologi och moderna arbetssätt för att erbjuda kursinnehållet.
- Studenterna skall – i alla kurser – utsättas för ett arbetssätt och en mentalitet som är baserad på modern industripraxis.