Daniel Einarson

Progressioner inom en projekt-baserad kurs baserad på principer från Software Engineering och CDIO

Abstract – För att möta kraven hos anställda inom industrin för moderna datorbaserade produkter behöver utbildningsanstalterna ge studenter möjlighet att ta till sig av såväl fundamentala och långlivade teorier som de senaste teknikerna. Detta är dock inte tillräckligt. Storskaliga produkter kräver samverkan mellan grupper av utvecklare och detta faktum i sig självt sätter fokus på ytterligare utmaningar. Själva arbetsprocessen måste synkroniseras, kommunikationen mellan utvecklare måste vara effektiv, osv. För att förbereda studenter på även sådana utmaningar måste således utbildningsanstalterna dessutom anpassa undervisningen så att även arbeten i projektgrupper, kommunikation inom dessa, styrning av arbetsprocesser, förhållningssätt, mm, ges lämpliga utrymmen.

CDIO är ett världsomspännande innovativt utbildningsramverk för att producera nästa generation av ingenjörer. I fokus för detta står projektbaserade undervisningsformer. Speciellt bejakar man arbetsfaserna Concieve (C), Design (D), Implement (I) och Operate (O), där man via dessa utvecklar en produkt som ska ha tillräckliga inslag av svårighet och komplexitet för att efterlikna de utmaningar man har i utveckling av verkliga produkter. Bakgrunden till detta initiativ ligger en identifiering av ett signifikant problem i teknikindustrin där man sett att utbildning av ingenjörer mestadels utgår från teoretiserade problem. Man har alltså ett gap mellan utbildningen och teknikindustrins krav. Vidare kräver reell produktutveckling samarbeten i arbetsteam, där även detta poängteras som CDIO-synvinkel.

Detta bidrag har tre huvudsyften. Dels kommer det att presentera CDIO och begrepp som förekommer inom detta. Dels kommer exempel ges på en kurs i Software Engineering, som kan ses som särskilt CDIO-inspirerad. Kursen har i sin form tre år bakom sig, där stegvis utveckling gjorts speciellt med fokus på att förbättra arbetsprocessen. Inspiration för progression av kursen har tagits såväl från CDIO som från området för Software Engineering. Inspiration från Software Engineering bör ses utifrån de barnsjukdomar och mognadsprocesser denna fortfarande tämligen nya disciplin har bakom sig. Därför kommer även detta att belysas. Vidare kan materialet bakom detta bidrag även delvis ses som en sammanställning av tre på varandra följande konferensbidrag till internationellt hållna CDIO-konferenser.

Nyckelord: Datavetenskap, Software Engineering, Projektbaserat lärande, CDIO, Arbetsprocesser.

1. Introduktion

Sedan ett antal år tillbaka har man vid datavetenskap, Högskolan Kristianstad, bedrivit utbildning baserad på de konkreta disciplinära kunskaperna för området, där träning i begrepp ofta ges i projekt-baserade former. En anledning till detta är att ämnet som sådant kan ses som naturligt tillämpat till sin karaktär. En annan anledning är att detta är det sätt man vanligtvis arbetar på i industrin. Förutom praktiska tillämpningar får man alltså en övning i ett sätt att arbeta.

Andra mer pedagogiska skäl bakom projektbaserade undervisnings-former rör livslångt lärande. Enligt [8] gäller att ”Projekt har en bestämd varaktighet, med en början och ett slut. Varje projekt är, till viss del, unikt”. Detta innebär att det ställs krav på deltagare i projekt att själva ansvara för de kunskaper som krävs av dem för att utföra sina uppgifter. Detta ger i sig självt ett stöd för idén om utbildningsanstalternas ansvar att ge en förberedelse för en föränderlig framtid, snarare än att leverera eviga sanningar. Såsom beskrivet i [2] ”*Universities are supposed to enable their students to engage in effective action in situations they are going to encounter but as the future is increasingly unknown, these situations are impossible to define in advance. What universities have to offer is knowledge and therefore you have to prepare for the unknown by means of the known*”.

Vidare nämns i [8] att ”Projekt är målorienterade. Projekt omfattar koordinering av uppgifter och de aktiviteter som är kopplade till dessa”. Detta refererar alltså till arbetsprocessen som en del i projektet. För just Software Engineering-baserade projekt är det här särskilt intressant att se arbetsprocesser mot bakgrund av de många misslyckade projekt ur denna disciplins barndom, samt de försök som gjorts att möta detta faktum. Utifrån begreppet Software Crises har utvecklats en rad processmodeller i samspel mellan akademi och industri, där detta sam-spel är pågående än idag och ses som väsentliga forskningsområden, befruktande för de ingående parterna. Att ta del av sådana utvecklade arbetsprocesser för studenter under utbildningen är naturligt eftersom det ger en god förberedelse inför en anställning som mjukvaruutvecklare.

Behovet av att utveckla utbildningar som ger en mogenhet hos nyut-examinerade att svara mot den flervärdiga komplexitet som råden inom teknikindustrin, är något som särskilt poängterats inom CDIO. CDIO ([16]) står för (projektfaserna) Conceive-Design-Implement-Operate och refere-

rar speciellt till projektformer i undervisning där dessa fyra faser är centrala. CDIO är vidare ett internationellt utbildningsramverk, med deltagare från hela världen, från USA till Australien, där lärandemål såväl som förhållningssätt i riktning mot CDIO-baserad undervisning föreslås. Software Engineering som område, såväl som CDIO ses som inspirerande för Datavetenskap vid Högskolan Kristianstad. Under ca ett års tid har man förberett för ett medlemskap i CDIO. Under december 2012 lämnades det in en formell ansökan om medlemskap. I januari 2013 blev man informellt medlemmar vid ett europeiskt-regionalt CDIO-möte i Århus, där man försvarade sin tidigare ansökan, samt slutligen blev man formellt medlemmar i mars 2013. Arbete pågår nu för att anpassa två av Datavetenskaps utbildningsprogram för att svara mot CDIO.

Detta bidrag bygger delvis och särskilt på en projektbaserad Software Engineeringkurs där reflektioner och progressionsarbeten inom denna kurs legat till grund för CDIO-baserade konferensbidrag. Detta handlar dels om sätt att arbeta inom projekt, dvs. arbetsprocessen, men även dels om aspekter såsom etik och ansvar. Arbetena bakom dessa konferensbidrag har i sig bidragit till en inspirationskälla inför Datavetenskaps medlemskap i CDIO.

Denna skrift är strukturerat på följande sätt: Bakgrunden för Software Engineering belyses, CDIO, speciellt CDIO syllabus presenteras, samt den aktuella kursen betraktas och arbetet med dess progression, vilket också utgör skriftens mest väsentliga delars. Avslutningsvis ges en avslutande summering.

2. Angående Software Engineering

Software Engineering som disciplin är relativt ny, men med en betydande påverkan på samhället. Födelsen för området är vanligen daterad till 1968 och en NATO-baserad konferens ([10]) angående temat Software Crises, detta som ett svar på återkommande problem kring brister i programvaruprojekt gällande kvalitet, förmåga att hålla sina deadlines, mm. Software Engineering som ingenjörsvetenskap introduceras då och en strävan att nå en mognad inom detta område tas på allvar.

Som en följd av behovet av en mognadsprocess inom Software Engineering, betraktas processmodeller, dvs. strukturer för arbetsflöden. Flera sådana har blivit föreslagna, de senare som svar på brister man funnit i tidigare processmodeller. Exempel:

- Vattenfallsmodellen. Denna modell belyser arbetsfaser som följer på varandra, typiskt *Krav*, *Design*, *Implementation*, *Test*, *Drift* och *Underhåll*. Processmodellen är intuitivt lätt att ta till sig, men har uppenbara brister exempelvis i att kraven inte är förstådda fullt ut

då designfasen och även senare faser genomförs. Detta leder till en produkt som inte motsvarar förväntningar. För mer information om detta, se exempelvis [14].

- Alterativ och inkrementell modell (exempelvis RUP). I denna modell möter man problemen med ofullständiga eller till och med okända krav, genom att ha återkommande möten med mottagare av systemet. Inför dessa möten har projektgruppen typiskt utvecklat en prototyp som står som utgångspunkt för diskussioner, där kraven och ytterligare aspekter på projektet och produkten hanteras. Denna modell är särskilt intressant för detta bidrag, då den även står som modell för arbetsprocessen för den kurs som diskuteras senare. En närmre granskning av det Iterativa och inkrementella arbetssättet ges senare.

- EXtreme Programming (XP). XP är ett exempel på en så kallad agil, eller lättrörlig metod. Man framhäver här i metoden bland annat möjligheten att optimera anpassning efter en kunds önskemål. För Datavetenskaps del är även denna projektmodell intressant och ligger till grund för ytterligare en annan kurs. För mer information om XP se exempelvis [18].

Ett flertal andra aspekter har också diskuterats kring hur området Software Engineering ska bli en mogen och trovärdig disciplin. Detta handlar om att definiera kunskapsområdet, att etablera övergripande och internationellt samlande organisationer, eller att se på etiska aspekter angående professionellt beteende.

Det särskilt intressanta i kravet på förändring hos Software Engineering i detta sammanhang är att resultaten av detta kan appliceras på och som en del i utbildningen. Man kan alltså låta sig inspireras av de processmodeller som utvecklats och introducera dessa i projektbaserade kurser. Man kan betrakta föreslagna etiska aspekter och introducera dessa i utbildningen, för att på så vis eftersträva en större mognad hos studenterna. Sett utifrån perspektivet CDIO, så kan man säga att strävan mot professionalism inom Software Engineering, ger infallsvinklar till hur man kan driva CDIO-baserad undervisning och motivera hur detta kan göras på ett kvalitativt rimligt sätt.

Nedan beskrivs ytterligare angående CDIO. Senare tas exempel upp där innovationer från Software Engineering bidrar till grundstruktur och progression inom en kurs. Detta motiverar i sig självt det CDIO-baserade arbetet med kursen.

3. Begrepp inom CDIO

CDIO har en bakgrund i den kritik som uttryckts i teknikindustrin angående alltför tillrättalagda utbildningar. Man säger att man undervisat i ingenjörsvetenskap snarare än att förbereda studenter i verklighetsnära sammanhang ([16]). CDIO poängterar särskilt projektorienterad undervisning, där faserna *Conceive*, *Design*, *Implement* och *Operate*, är centrala. CDIO är ett utbildningsramverk för att möta upp mot ett antal identifierade aspekter man önskar att en nyexaminerad student ska svara upp emot, för att ta de utmaningar man möter i teknikindustrin. Förutom erfarenheter i projektarbeten och grundläggande kunskaper inom disciplinen, så handlar det om aspekter såsom kommunikation med sina medarbetare, förmåga att presentera material, etiska förhållningssätt och ansvar, mm. Centralt i CDIO är den så kallade CDIO syllabus ([17]) som sammanfattar ett antal lärandemål man önskar ska vara mer eller mindre uppfyllda inom en CDIO-baserad utbildning. CDIO syllabus presenteras på flera sätt med avseende på detaljrikedom. Man talar om att man har fyra nivåer, där nivå 1 har högst abstraktionsnivå, medan nivå 4 har högst detaljrikedom. CDIO syllabus är vidare indelad i fyra huvudkategorier refererande till typer av utbildningsmål. Dessa är:

1. **DISCIPLINARY KNOWLEDGE AND REASONING**, svarande mot kunskaper i ämnet.

2. **PERSONAL AND PROFESSIONAL SKILLS AND ATTRIBUTES**, svarande mot möjligheter till reflektion, analys, attityder, etik, ansvar, mm.

3. **INTERPERSONAL SKILLS: TEAMWORK AND COMMUNICATION**, svarande mot arbete i grupper, kommunikation, mm.

4. **CONCEIVING, DESIGNING, IMPLEMENTING, AND OPERATING SYSTEMS IN THE ENTERPRISE, SOCIETAL AND ENVIRONMENTAL CONTEXT**, svarande mot, i första hand, projektarbeten.

Uppställning ovan svarar i sin tur mot en nivå 1 av CDIO syllabus. Nivå 2 kan ses nedan. För ytterligare nivåer refereras till mer material angående CDIO syllabus ([17]).

1 DISCIPLINARY KNOWLEDGE AND REASONING
1.1 KNOWLEDGE OF UNDERLYING MATHEMATICS AND SCIENCE
1.2 CORE FUNDAMENTAL KNOWLEDGE OF ENGINEERING
1.3 ADVANCED ENGINEERING FUNDAMENTAL KNOWLEDGE, METHODS AND TOOLS
2 PERSONAL AND PROFESSIONAL SKILLS AND ATTRIBUTES
2.1 ANALYTICAL REASONING AND PROBLEM SOLVING

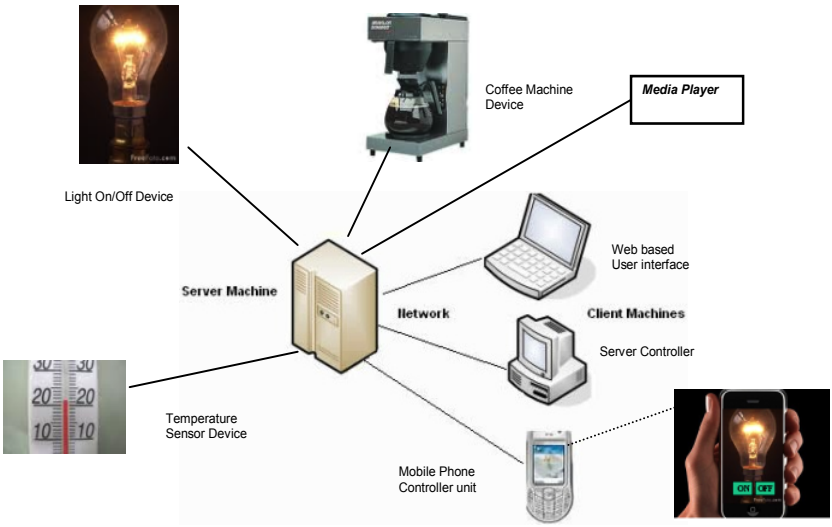
2.2 EXPERIMENTATION, INVESTIGATION AND KNOWLEDGE DISCOVERY
2.3 SYSTEM THINKING
2.4 ATTITUDES, THOUGH AND LEARNING
2.5 ETHICS, EQUITY AND OTHER RESPONSIBILITIES
3 INTERPERSONAL SKILLS: TEAMWORK AND COMMUNICATION
3.1 TEAMWORK
3.2 COMMUNICATIONS
3.3 COMMUNICATIONS IN FOREIGN LANGUAGES
4 CONCEIVING, DESIGNING, IMPLEMENTING, AND OPERATING SYSTEMS IN THE ENTERPRISE, SOCIETAL AND ENVIRONMENTAL CONTEXT
4.1 EXTERNAL, SOCIETAL AND ENVIRONMENTAL CONTEXT
4.2 ENTERPRISE AND BUSINESS CONTEXT
4.3 CONCEIVING, SYSTEMS ENGINEERING AND MANAGEMENT
4.4 DESIGNING
4.5 IMPLEMENTING
4.6 OPERATING

Intressant är här att se hur man kan skapa kursinnehåll som svarar mot ett innehåll av ovanstående slag, detta kommer delvis att presenteras senare i detta bidrag. Man bör känna till att CDIO ska se som ett ramverk som tillämpas för ett program som helhet och det är alltså inte så att man ska eftersträva att möta alla lärandemål från CDIO syllabus i en och samma kurs.

4. Kursexempel inom Software Engineering med kopplingar till CDIO

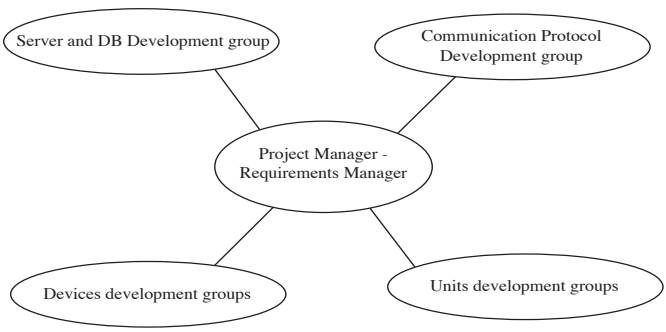
Den kurs som betraktas här löper på halvfart under höstterminen. Sin nuvarande form har den haft tre år i rad. Kursen har haft ca 50-60 studenter vid varje tillfälle och man är indelad i tre större grupper om ca 20 studenter i varje, för att slutföra ett större projekt. Varje grupp är sedan indelad i subgrupper om ca 3 studenter som var och en utför en mindre del av hela projektet. Detta ger stora utmaningar i kommunikation och koordination av arbetsuppgifter. Det är därför lämpligt att man följer en processmodell man är överens om och att man väljer lämpliga verktyg för kommunikation.

Projektet handlar om att sammanställa ett så kallat smart hus, där enheter i detta styrs av datorer och mobiltelefoner. Figur 1 visar huvudstrukturen i detta och denna struktur ligger också till grund för hur man organiserar sig i subgrupper.



Figur 1: Strukturen för Smart Hus-projekt

Fysiska enheter man jobbar mot här är exempelvis lampor, termometer, värmeelement, fläktar, simulerade enheter som kaffebryggare, och musikspelare. Dessa styrs via grafiska webbaserade interface, eller smart phones. En server och en databas håller i information om styr-enheter och styrda enheter. Väsentlig blir också att definiera kommunikationen mellan samtliga enheter. Detta ger att en projektgrupp lämpligen organiserar sig enligt figur 2.



Figur 2: Projektgruppstruktur

De tekniska förutsättningarna för detta projekt presenterades vid [7], medan de pedagogiska förutsättningarna presenterades vid [4], [5] och [6]. De tre senare presenterar kursen som en progression i tre steg, detta kommer att beskrivas mer senare. I centrum står här även CDIO och dess utgångspunkter.

Vad som är intressant här är att se att kursen refererar till flera av de lärandemål som tas upp i CDIO syllabus. Till att börja med introduceras kursmaterial för att kunna hantera projektets enheter och kommunikationen dem emellan. Detta refererar till 1. **DISCIPLINARY KNOWLEDGE AND REASONING**. Ur 2. **PERSONAL AND PROFESSIONAL SKILLS AND ATTRIBUTES** kan man se flera lärandemål som berörs av kursen, såsom

> 2.1 ANALYTICAL REASONING AND PROBLEM SOLVING
> 2.2 EXPERIMENTATION, INVESTIGATION AND KNOWLEDGE DISCOVERY
> 2.3 SYSTEM THINKING

Senare kommer även aspekter såsom attityder och etik diskuteras. Ur den tredje kategorin **INTERPERSONAL SKILLS: TEAMWORK AND COMMUNICATION** berörs samtliga punkter av CDIO syllabus, (nivå 2). COMMUNICATIONS IN FOREIGN LANGUAGES berörs då kursen ges på engelska. Vidare handlar den fjärde kategorin särskilt om projektarbete som studiearbetsform som sådan.

Nedan kommer kursen att beskrivas som en serie i tre av kurstillfällen. Problem vid de olika tillfällena kommer att beröras och hur man försökt eftersträva lösning presenteras. Moment inom kursen kommer då också att behandlas mer ingående och inspiration särskilt från Software Engineering för att möta problem kommer att redovisas. Denna källa anknyter sedan väl till CDIO som princip.

5. Progression av kurs, ht 2010 – Steg 1, initialt steg

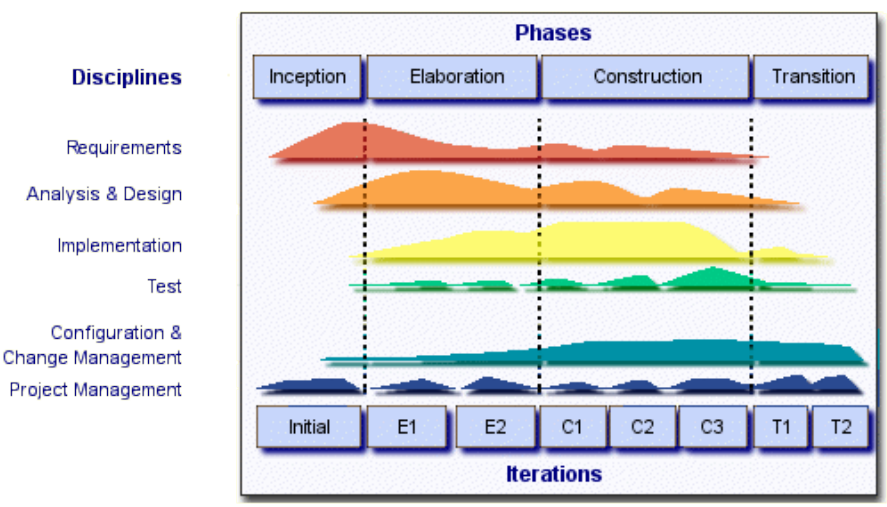
Kursen Software Engineering 2,om 15 hp, löper under höstterminen på halvtid, med en avslutning i januari. Den är å ena sidan en kurs som tar upp principer inom området Software Engineering, dvs med alla de aktiviteter och artefakter som behövs för att slutföra ett projekt. Å andra sidan exemplifieras kursen med det projekt som introducerades ovan. Kursen hölls första gången i denna skepnad hösten 2010 och gavs således tredje gången hösten 2012.

Här kommer en ytterligare beskrivning av principer kring den processmodell som används på kursen att ges. Hur denna appliceras i kursen

visas även. Detta arbete ligger sedan till grund för den första av den serie om tre konferensbidrag som antagits vid CDIOs konferenser. Detta arbete presenteras, liksom kopplingar till CDIO.

5.1 RUP, en iterativ och inkrementell processmodell

Till skillnad från Vattenfallsmodellen, med tidmässigt linjära på varandra följande faser, är RUP (Rational Unified Process) snarare uppbyggd som en tvådimensionell modell, enligt figur 3 nedan. Huvudfaserna är *Inception* (att sätta sig in i problemet, avgöra resursbehov mm), *Elaboration* (skapa huvudsaklig design), *Construction* (huvuddelen av den konkreta systemutvecklingen) och *Transition* (att sätta systemet i drift). Parallellt med detta löper deldiscipliner enligt figur 3. Betydelsen i detta är att det att avgöra exempelvis systemkraven görs under en längre tid och är inte betraktat som avslutat vid en fas. Detsamma gäller övriga discipliner. Med iterativt och inkrementellt, menas här att man har olika milstolpar inför vilka man utför en del av den totala mängden av projektet. Vid dessa punkter, såsom exempelvis efter E1, E2, C1, C2, osv. stämmer man av med mottagare av systemet varpå korrigeringar kan göras, bekräftelser kan fås, mm. Detta arbetssätt ökar sannolikheten att den färdiga produkten faktiskt motsvarar kundens/beställarens förväntningar och behov.



Figur 3: Faser och discipliner i RUP ([12])

Översatt i termer av kurser inom Datavetenskap, så innebär denna typ av arbetssätt att man har återkommande möten med lärare inför vilka man förbereder dokument svarande mot krav, design, test, osv. Dessa dokument svarar mot status för projektet vid just detta tillfälle. Processen kommer att beskrivs mera nedan. För mer information om RUP se exempelvis [12].

5.2 Kursens arbetsflöde

Arbetsflödet i kursen svarar mot de iterativa stegen, med möten mellan projektgruppen och läraren. Här diskuteras problem och tänkbara lösningar, här visas även prototyper på systemet under utveckling. Dokumenten, eller artefakterna man utvecklar till dessa tillfällen blir då också centrala. De artefakter man valt att ha fokus på i denna kurs är:

- **Vision document**, Detta beskriver projektets huvudsakliga vision. Typiskt kan det ge uttryck för projektbeställarens förväntningar på systemet. Detta blir på så vis också en utgångspunkt för kravdokumenten. 4

- **Project risks**, Projektriskerna kan kategoriseras enligt följande ([14])
    - *Project risks*, som kan ge effect på tids- och resursplanering
    - *Product risks,* som kan ge effect på kvalitet eller exempelvis prestanda hos den utvecklade produkten.
    - *Business risks,* kan ge effect på utvecklingsorganisationen.

- **Requirement document**, Svarar mot systemets funktionella krav, dvs. vad systemet ska göra, eller den service den ska ge.

- **Supplementary requirements**, Svarar mot icke-funktionella krav, eller så kallade kvalitetsattribut. Detta är typiskt tre-falt ([14]):
    - *Product requirements*, Krav på produkten som handlar om aspekter såsom prestanda, pålitlighet, användarvänlighet, mm.
    - *Organisational requirements*, Krav på organisationen, typiskt som konsekvens av policies, processtandarder, mm.
    - *External requirements*, Svarande mot krav externa till systemet. Detta kan handla om juridiska aspekter, integritetspolicies, mm.

- **Design document**, Typiskt svarande mot systemets arkitektur och design. Detta bidrar sedan till guidning av hur systemet ska konstrueras.

- **Verification, Validation, and Test**, Dokument som tar upp hur tester ska utföras/har utförts.

Denna mängd artefakter laddas upp på kurssidan inför varje möte med läraren. Man har en sådan mängd dokument per subgrupp, samt ett par extra styrande dokument för hela gruppen. Typiskt används här dokumentet för icke-funktionella krav för projektgruppen som helhet, där processtandarder, utvecklingsstandarder, mm. kan ges uttryck. Artefakterna utvecklas på individnivå, vilket ger en grund för individuell betygsättning.
Mellan de formella mötena med läraren, har projektgrupperna interna informella möten, som inte är styrda av läraren. Dessa kan hållas som fysiska sammankomster, eller virtuella sammankonster via systemstöd för detta. Studenterna väljer här själva hur de ska kommunicera sinsemellan. Typiska val här är interna facebookgrupper för diskussioner inom gruppen och DropBox för att dela dokument.

Kursen har två presentationstillfällen, en tidig för att delge resultat så långt, samt en slutpresentation. Vid slutpresentationen ska var och en av studenterna presentera en bit av det utvecklade projektet, vanligtvis görs detta via Power Point. Man ska även visa en komplett körning av hela systemet.

5.3 Angående CDIO konferensbidrag, 2011

Bidraget till CDIO-konferensen 2011 ([4]) utgör en utgångspunkt för Datavetenskaps senare inträde i CDIO. Detta arbete är två-falt:

- Jämförelser mellan Software Engineering och CDIO. Å ena sidan dras paralleller mellan de arbetssätt som man tvingats introducera inom Software Engineering för att säkerställa färdigställande av projekt med tillräckligt hög kvalitet, å andra sidan har vi CDIO och dess bakgrund

- Visa ett exempel från en projektdriven Software Engineeringkurs och se att denna kan ses som en CDIO-baserad kurs.

Den första punkten blir här särskilt intressant då vi kan se att med det kursupplägg man har haft inom Datavetenskap, grundat på principer ur Software Engineering, så ligger man redan innan inträdet i CDIO väldigt nära CDIO. I princip är det stora arbetet så att säga redan gjort och kvar har man att formalisera uttryck i kurser, kursplaner, utbildningsplaner, mm. så att dessa tydligare stämmer med CDIOs ramverk. Speciellt kan man även här se den stora överensstämmelsen mellan CDIOs Conceive-Design-Implement-Operate och RUPs Inception-Elaboration-Construction-Transition. Likheterna i tankesätt här är slående.

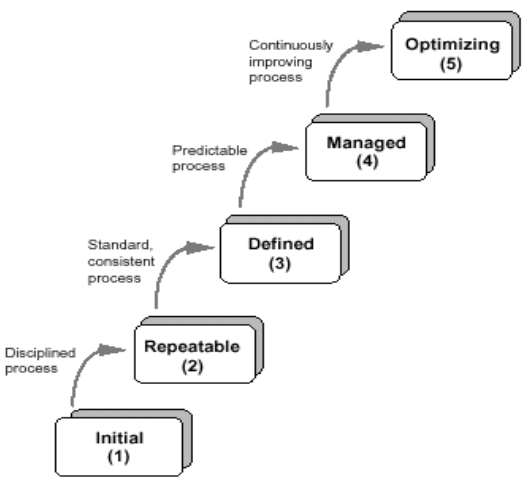
Den andra punkten rör, förutom att ge ett exempel på en Software Engineering- eller CDIO-kurs, även kursens upplägg och diskussioner kring kursens utfall och vad som kan förbättras. Vad man speciellt kunde se var att studenterna hade svårt att förstå arbetssättet. Man lyckades visserligen slutföra färdigställandet av sitt system, men arbetsprocessen var otydlig, vilket i sig bidrog till risker beträffande själva systemets färdigställande. För mer information kring detta, se [4].

6. Progression av kurs, ht 2011 – Steg 2, med utgångspunkt i CMM

För att komma tillrätta med de uppenbara problem man kunde se i kursen under ht 2010, så undersöktes metoder för att föra över mer ansvar kring processmodellen till studenterna själva. Inspiration hämtades återigen från Software Engineering och i detta fall utifrån en modell för att värdera mognadsgrad hos företag inom programvarubranschen. Metoden kallas CMM och beskrivs nedan. Förutom att värdera status, så kan den även tjäna syftet att vägleda företag mot en högre mognadsgrad.

6.1The Capability Maturity Model, CMM

The Capability Maturity Model, CMM, har utvecklats som ett medel att värdera utvecklingsprocessen hos mjukvaruföretag. Den är i huvudsak utvecklad av Software Engineering Insitute (SEI) som ett samarbetsprojekt mellan akademi och företag ([13]), med medel från U.S. Department of Defense. Detta är ett steg i att finna strategier för att skapa förbättringar i mjukvaruorganisationers arbetsprocesser. I sin kärna beskrivs CMM som en 5-nivåmodell där varje övre nivå står för en högre nivå av mognad ([14]). Figur 4 illustrerar detta.



Figur 4: Nivåer i CMM ([3])

Sammanfattningsvis kan dessa nivåer beskrivas enligt nedan ([15]). För mer information om CMM, se exempelvis [14]).

- Nivå 1 (**Initial** - ingen kunskap): Man har ingen utvecklingsprocess och ingen kvalitetskontroll. Man kan inte förutsäga utvecklingskostnad eller produktkvalitet, och man har inte någon kunskap om dem i efterhand.
- Nivå 2 (**Repeatable** - viss kunskap): Man jobbar på liknande sätt varje gång, men utvecklingsprocessen är inte så bra att man med någon större säkerhet kan förutsäga utvecklingskostnad eller produktkvalitet. Däremot är de kända i efterhand.
- Nivå 3 (**Defined** - full kunskap): Man har en utvecklingsprocess som man själv har bestämt hur den skall se ut. Man kan med hög noggrannhet förutsäga utvecklingskostnad och produktkvalitet.
- Nivå 4 (**Managed** - viss valbarhet): Man kan mäta hur förändringar i utvecklingsprocessen påverkar utvecklingskostnad och produktkvalitet, och man kan förändra sin process så att den blir bättre.
- Nivå 5 (**Optimized** - full valbarhet): Man kan, från fall till fall, välja den utvecklingskostnad och produktkvalitet som passar i det aktuella projektet.

CMM har visat sig värdefull i reflektioner kring mognadsgraden i utvecklingsprocesser. Idag diskuteras dock utvecklingsorganisationens mognadsgrad snarare än processen. För detta har istället introducerats CMMI ([15]) medan CMM inte längre understödjs och utvecklas som begrepp. I detta fall, däremot, kan CMM fortfarande fungera som en utmärkt källa till inspiration och utveckling. För projektkursens del har CMM varit värdefull och nedan beskrivs mer kring på vilket sätt denna fungerat som utgångspunkt för utveckling av kursens projekt.

6.2 Angående CDIO konferensbidrag, 2012

Arbetet inför bidraget till CDIOs konferens 2012 ([5]) utgår ifrån de uppenbara svårigheter som studenter tycktes ha i att förstå modellen för arbetsprocessen. Detta uttrycktes även av studenterna vid den kursundersökning som gjordes. Istället för att lägga mer arbete än tidigare på läraren för att än mer förklara arbetssätten, så testades sätt att få studenterna mer aktiva i att själva reflektera över sin arbetsprocess och även påverka denna. Deras egen påverkan skulle innefatta egna beslut i utföranden, diskussioner inom gruppen och medföra större förståelse för i helhet och detaljer. Som modell för att introducera detta har använts CMM. Nivå 1, den initiala nivån tycks ofta användas på projektkurser. Nivå 2 är den som svarar mot hur kursen genomfördes första året. Den nya ansatsen handlar om att försöka att föra upp projektet för kursen till nivå 3. För mer information om denna nivå, se uppställningen ovan.

I fokus står återigen arbetet kring dokumenten. Tidigare har man fått mallar för dokument, nu uppmanas man att utveckla dessa så att de får en form som man kommer överens om i gruppen. Man uppmanas att vara klarare i sina styrande dokument angående egna standarder för arbetsprocessen, mm. Vidare uppmanades man att se på kommunikationen mellan de olika dokumenten. Diskussioner skulle genomföras exempelvis angående kring hur kravdokumentet svarar mot designdokumentet. Diskussioner fördes vidare kring hur mätvärden skulle kunna introduceras svarande mot hur långt man kommit i att realisera exempelvis ett krav.

Nedan följer typdiskussioner vid projektgruppsmöten med lärare. Exemplen är hämtade från [5]. För mer information, se det arbetet.

Why is the style of your document different from the other of your subgroup? Do you not have any agreements on how you shall work, and the outlook of the outcome of your work? How is your internal communication actually? This is a risk document, what about the risks of your group? What are the states of the encountered risks? How can you modify your risk document so you make sure you handle the risks of your internal collaboration?

How does this requirement correspond to any of the design items, or test items? How can you see what design item follows from what requirement? How can we see cooperation and coordination of work in your subgroup? How far have you come in developing and implementing this requirement?

This Supplementary requirements list is written in an inappropriate way. You mix product requirements with process requirements. How can you change the artifact to categorize the supplementary requirements clearer?

De slutsatser som läraren kunde dra av denna ansats var att arbetet tycktes förbättras avsevärt. Det fanns en generellt sett större kontroll över vad man gjorde och det tycktes som att man hade en klart större förståelse för arbetsprocessen som sådan. Dock kunde man se exempel på stora slitningar speciellt inom en av projektgrupperna. Slutsatserna från lärarens sida var att det krävdes ytterligare insatser i undervisningen, insatser som snarare hade att göra med den roll studenten hade i förhållande till sig själv och till gruppen. Arbetet med detta belyses under steg 3, nedan. I förhållande till CDIO kan man inte se att ytterligare lärandemål från CDIO syllabus berörs genom introduktionen av CMM som

begrepp. Däremot är det av stort intresse att se detta som ett bidrag till projekt-baserad undervisning som sådan.

7. Progression av kurs, ht 2012 – Steg 3, inkluderande etiska aspekter

Vad man kunde se av resultatet av kursen ht 2011, var ett uppenbart steg framåt generellt sätt i studentgruppen angående förståelse av arbetsprocessen. Men uppenbara problem kunde också ses i subgrupper av studenter där arbetsmoralen verkade bristfällig. Detta skapade mycket stor irritation särskilt hos de projektledare som såg ett ansvar för projektets fortsatta utveckling.

Redan tidigare uppstod idéer hos huvudläraren kring att se mera av etiska aspekter kring beteende inom en projektgrupp. Reflektioner kring detta gjordes genom studier och genomgång i studentgruppen av de etiska koder som introducerats inom Software Engineering, för att öka mognadsgraden och trovärdigheten för området genom guidning av aktiva inom programvaruutveckling beträffande beteenden, moral och målsättningar. Nedan tas dessa etiska förhållningssätt upp.

Dessa etiska aspekter är inte enbart av intresse som något som bör presenteras för studenter inför deras kommande yrkesliv. Ett särskilt intressant perspektiv handlar om hur man kan få studenter att anamma dessa idéer redan i sin studiesituation inom projektkurser och därmed förbättra arbetsprocessen och projektarbetet redan under studietiden. Detta är också något som introducerades och studerades av huvudläraren under kursen ht 2012. Resultatet av studien tas upp i bidraget till CDIO-konferensen 2013 och redovisas nedan, efter en genomgång av etiska principer inom Software Engineering.

7.1 Angående mognaden hos en disciplin

Enligt [11] behöver ett flertal nyckelaspekter uppfyllas för en disciplin för att den ska betraktas som en mogen disciplin. Detta handlar exempelvis om en tydlighet inför en så kallad *Body of knowledge*, svarande mot summan av alla kunskaper som utgör ett område, *professional societies*, dvs. organisationer som står för en expertis inom området, såsom ACM och IEEE, *Code of Ethics*, svarande mot etiskt beteende hos aktiva inom området, eller *Initial professional education system*, dvs. väldefinierade utbildningsprogram för studenter och lärosäten.

Särskilt intressant är här att lyfta fram etiskt beteende. Detta för att det dels faktiskt är ett lärandemål som utvärderas av (tidigare) Högskoleverket (dess omorganisation tas inte upp här) som ett väsentligt lärande-

mål att uppfyllas inom utbildningen. Dels är detta också ett lärandemål som tas upp bland CDIOs lärandemål. Detta lärandemål diskuteras senare, liksom hur detta tas upp och används inom den aktuella kursen.

För att definiera etiskt beteende har ACM tillsammans med IEEE utvecklat *The Code of Ethics*. Denna tar som primär punkt upp hur man bör förhålla sig till allmänheten, samt sekundärt hur man förhåller sig till sitt arbete, enligt uppställningen nedan.

1. **PUBLIC -** Software engineers shall act consistently with the public interest.
2. **CLIENT AND EMPLOYER -** Software engineers shall act in a manner that is in the best interests of their client and employer consistent with the public interest.
3. **PRODUCT -** Software engineers shall ensure that their products and related modifications meet the highest professional standards possible.
4. **JUDGMENT -** Software engineers shall maintain integrity and independence in their professional judgment.
5. **MANAGEMENT -** Software engineering managers and leaders shall subscribe to and promote an ethical approach to the management of software development and maintenance.
6. **PROFESSION -** Software engineers shall advance the integrity and reputation of the profession consistent with the public interest.
7. **COLLEAGUES -** Software engineers shall be fair to and supportive of their colleagues.
8. **SELF -** Software engineers shall participate in lifelong learning regarding the practice of their profession and shall promote an ethical approach to the practice of the profession.

The Code of Ethics presenteras i två nivåer med avseende på detaljrikedom. Ovan är kortformen. Denna, liksom den längre, mer utförligt beskrivna, kan ses via [1].

7.2 Angående CDIO konferensbidrag, 2013

Processmodellen är här den samma som tidigare år. Nyheten är introduktion av reflektion kring etiskt beteende. Studien utförs i kursen genom följande steg:

1. Materialet Code of Ethics, gås igenom vid en tidig föreläsning i september, strax efter kursstart. I samband med detta klargörs också att ett experiment kommer att utföras.

2. En vecka till in i kursen utförs del ett av experimentet. Detta går ut på att studenterna läser kortformen av the Code of Ethics på egen hand och uppmanas att reflektera över denna. Man skriver

ner sina reflektioner, per punkt, på papper, som om det vore sina egna ställningstaganden i förhållande till punkterna i the Code of Ethics. Man skriver under pappret som om det vore ett kontrakt. Information ges dock angående att det hela är helt informellt och inte på något sätt betygsgrundande.

3. The Code of Ethics diskuteras vid projektmötena och hur man lyckas med att förhålla sig till dessa inom gruppen tas också upp. Särskilt poängteras här det egna ansvaret och förhållningssättet till sina kollegor och projektledare.

4. Ytterligare ett antal veckor in på kursen presenteras den längre formen av the Code of Ethics. Ett nytt kontrakt, av samma typ som tidigare, ska skrivas under. Detta görs för att fördjupa reflektionsgraden kring the Code of Ethics.

5. I samband med kursevalueringen ombeds studenterna skriva ner sina åsikter kring introduktionen av The Code of Ethics.

Resultatet av experimentet har blivit överlag bra. Slitningar i projektgrupperna av det slag man haft tidigare år, har inte synts utåt. Senare har det framkommit att det ändå inte varit helt friktionsfritt, men uppfattningen är ändå att framstegen varit stora sett i relation till tidigare kurstillfällen.

Den grupp som tog the Code of Ethics på störst allvar under projektgruppsdiskussionerna var också den grupp som klarade sig klart bäst. Detta gäller både angående arbetsprocessen och angående det utvecklade systemet. Kursevalueringen var mycket positiv i sina omdömen kring introduktionen av The Code of Ethics. Detta gällde både som förberedelse till kommande arbetsliv och som reflektionsform i kursprojektet. Intressant är här att nämna att man genom denna ansats når CDIOs lärandemål

2.4 ATTITUDES, THOUGHT AND LEARNING
2.5 ETHICS, EQUITY AND OTHER RESPONSIBILITIES

Vad som kan ses som än mer intressant är att HSVs granskning av Datavetenskaps utbildningsprogram tar upp bland annat just lärandemål kring etiska aspekter. För att visa att Datavetenskap svarar upp mot detta lärandemål har just experimentet med The Code of Ethics inom denna kurs särskilt belysts. För mer angående detta arbete, se [6].

7.3 Reflektioner

Detta är det tredje och i nuläget sista av de CDIO-bidrag som genererats som ett resultat av arbetet med kursen och ansträngningar att nå en progression av kursen. Antalet studenter som man sett problem för i denna

kurs har minskat och nått ner på en nivå för enskilda studenter. Kursen kan i nuläget ses som tämligen stabil, som ett resultat av de förbättringar som gjorts.

Tankar finns kring att nå ännu ett steg ner i fingranularitet med avseende på grupper av studenter, genom att introducera anonym studentbetygsättning. Ett traditionellt problem inom projektbaserad undervisning handlar om att studenter alltför lättvindigt hänger med i andras arbeten där detta döljs av medstudenternas krav på lojaliteter. Här kan sådan form av kontinuerlig granskning bidra till mindre utrymme för sådan typ av problem.

8. Summering

Detta bidrag har betraktat och diskuterat en projektbaserad kurs, utvecklad för att förbereda studenter för en framtid i datasystemutvecklingsindustrin. Som utgångspunkt för detta har inspiration tagits dels från Software Engineering som disciplin och dels från det internationella utbildningsramverket CDIO. Båda dessa har utvecklats som krav från industrin för att möta de behov som finns angående komplexa arbetsformer och komplexa produkter. CDIO har utvecklats som ett utbildningsramverk för universitet och högskolor, med särskilt fokus på projektformer, men betraktar även en uppsättning lärandemål, sammanfattade i den så kallade CDIO syllabus. Disciplinen Software Engineering har uppstått ur en historia med misslyckade projekt, där arbetsprocesser utvecklats som svar på detta, tillsammans med ett flertal andra komponenter, även mjukare såsom förslag på etiskt hållbart beteende.

Diskussioner kring kursen utgår ifrån de tre på varandra följande år som den getts i nuvarande skepnad och de progressioner som gjorts under denna tid. I huvudsak har kursen haft följande progressioner:

1. Initialt steg. Kursens arbetsprocess definieras enligt processmodeller föreslagna inom Software Engineering. Detta svarar väl och tydligt mot flera av CDIOs lärandemål, såsom angående *Team work*, *Communication* och de lärandemål som direkt rör projektarbete som arbetsform.

2. Introduktion av CMM. Denna modell hämtas från Software Engineering för att värdera och guida i utvecklingsorganisationers mognadssteg angående deras arbetsprocess. Arbetet med detta har en gett en utmaning hos studenterna att mera ta ansvar för sin egen arbetsprocess, med resultat i högre kvalitet i arbetsprocessen såväl som projektets slutprodukt. Detta förhöjer värden utpekade av CDIO.

3. Introduktion av etiska principer. Inspiration hämtas här från Software Engineering och The Code of Ethics och bidrar bland annat till att studenterna mer reflekterar över sig själva och deras relation till övriga projektdeltagare. Följden blir att irritationer inom projektgruppen reduceras och kvaliteten i arbetsprocessen ökar ytterligare, parallellt med projektets produktkvalitet. Detta svarar mot särskilt utpekade lärandemål inom CDIO och även motsvarande från den svenska högskoleförordningen.

De tre stegen ovan har alla presenterats vid CDIOs internationella konferenser. Reflektioner från kursansvarig har varit att det setts som mycket intressanta och lyckade insatser. Gruppen studenter som kan ses som att ha svårigheter med kursen har steg för steg reducerats i storlek. De steg som introducerats kommer att bibehållas inom kursen, samtidigt som betraktelser av nya idéer kring progressioner kommer att göras.

Referenser

[1] ACM, Association for Computing Machinery, Advancing Computing as a Science & Profession, "Software Engineering Code of Ethics and Professional Practice", http://www.acm.org/about/se-code

[2] Bowen J. and Marton F., The University of Learning – Beyond Quality and Competence, Routledge, 1998.

[3] Capability Maturity Model. http://ibiblio.org/gferg/ldp/SCM-OpenSource/scm-traditional.html

[4] Einarson D., "Working- vs. Educational Processes in Software Engineering vs. CDIO", *7th International CDIO Conference*, 2011.

[5] Einarson D., "Approaching CMM to an Educational CDIO based Software Engineering Process", *8th International CDIO Conference*, 2012.

[6] Einarson D., "Ethics and Respopnsibilities in a CDIO based Software Engineering Process", *9th International CDIO Conference*, 2013.

[7] Einarson D., "Internet of Things in Education", STTC, China Zhejiang International Science and Technology Cooperation Conference – part Internet of Things (IOT) at Jiaxing, Hangzhou China, 2010, (No paper, only presentation).

[8] Eriksson Mikael, Lilliesköld Joakim, *Handbok för mindre projekt*, Liber, ISBN 91-47-05270-8, 2004.

[9] Högskoleverktet, HSV, www.hsv.se

[10] Naur P. and Randell B. (Eds.), Software Engineering: Report of a conference sponsored by the NATO Science Committee, Garmisch, Germany, 7-11 Oct. 1968, Brussels, Scientific Affairs Division, NATO (1969) 231pp.

[11] Pour G., Griss, M. L., and Lutz M. "The push to make software engineering respectable", *IEEE Internet Computing*, May, 2000, 35-43.

[12] Rational Unified Process in Education, UPEDU, http://www.upedu.org/

[13] On The Software Engineering Institute (SEI), http://en.wikipedia.org/wiki/Software_Engineering_Institute
[14] Sommerville I., Software Engineering, Addison Wesley, 2010.
[15] Wikipedia om Capability Maturity Model, CMM, http://sv.wikipedia.org/wiki/Capability_Maturity_Model
[16] Worldwide CDIO Initiative: A Framework for the Education of Engineers, http://www.cdio.org/
[17] Worldwide CDIO Initiative: The CDIO Syllabus, V2.0
[18] XP, angående eXtreme Programming, http://xprogramming.com/index.php