

# Innovativ programmering - software craftsmanship, open source, design

## -Förslag på curriculum till ny studentexamen

Dokumentets författare: Lars Degerstedt

Innehåll: Mikael Kindborg, Arne Jönsson, Jalal Maleki och Anders Haraldsson

*I texten används genomgående det gamla poängsystemet med 40 poäng per läsår. Detta kommer anpassas senare till de nya högskolepoängen.*

## Introduktion

Innovativ programmering (IP) är en ny utbildning på 120-p (studentexamen) som är tänkt att komplettera de existerande IT-utbildningar C/D/DI. Den stora nyheten ligger i betoningen av **hantverkskunnande** inom programmering. De grundläggande programmeringsämnena undervisas likartat med existerande linjer. Men i existerande utbildning ges dessutom utvidgad matematisk och allmänorienterad teoretiskt fördjupning. Här väljer IP istället att fördjupa den **individuella skickligheten** som programmerare och att kunna fungera väl ihop med andra i **projekt och community**.

Undervisning i IP består av programmeringsnära teori och omfattande egen programmering/systemhantering. Betoningen ligger på att omsätta en relativt begränsad men högst användbar teori genom omfattande utveckling av den egna hantverkarkompetensen. Det viktiga är att den kunskap som lärs drivs ända fram till yrkesmässigt användbar förmåga, på bekostnad av att en mindre mängd ämnen kan täckas in. Ämnena väljs baserat på vilka tillämpningar som bedöms som industriellt relevanta när studenten går ut i arbetslivet 3-5 år senare. Dessa ämnen och universitetets kompetens för TP kommer därmed förändras över tiden på ett kontrollerat sätt. Vi förutser också att det trots allt kommer finnas en grund som inte förändras så snabbt, vad gäller t ex språk och plattformar och hantverksregler.

Tentativt tror vi att studenterna i den närmsta tidsrymden kommer få jobb bl a som: **programmeringskonsulter, tekniker, webb- och UI-utvecklare**. Betoningen ligger på att snabbt kunna fånga en kunds krav och leverera en bra lösning. Det kan röra sig om snabb skriptprogrammering, mer avancerade former av system och arkitekturer, samt mer system-drift-orienterade jobb. Därutöver vill vi också uppmuntra till **eget entreprenörskap**. Här tänker vi på det nya utrymme som finns för IT-baserade produkter och online-tjänster, där det  **kreativa och affärsmässiga** momentet är minst lika viktigt som det rent tekniska/formella utbildningsmässiga. Det är här viktigt att kunna paketera sin kompetens/tjänst/produkt och sälja den på ett bra sätt, t ex som underkonsult eller som online-företag.

Utbildningen har fem grundläggande teman som kurserna grupperas runt:

- Programspråk och plattformar
- Användbarhet/design
- Software craftsmanship
- innovativ egen tillämpning
- Open Source

Viktig **bok**: Software Craftsmanship av McBreen (kursbok i PP2-kursen/ETE247). Detta är en viktig grundläggande bok som betonar vikten av individuellt hantverkskap bland annat. Boken myntar begreppet SC och kan användas som en standardreferens för de värderingar som bör genomsyra denna utbildning.

Vi är också positiva till att utbildningen ges på **engelska** och erbjuds för både svenska och internationella studenter. För en programmerare är ju ändå arbetspråket engelska, så det känns naturligt för utbildningen.

## Blockbaserad översikt

År 1: Grundår

- språk+plattformar (15p)
- programmeringsnära MDI (5p)
- datateknik (5p)
- projekt + hantverk (15p)

År 2: Tillämpningsområden

- informationssystem och databaser (5p)
- Internet (5p)
- media, konsumentprodukter och spel (5p)
- operativsystem och systeminstallation (5p)
- övr teori t ex matematik (5p)
- projekt + metodik och design (15p)

År 3: Fördjupning

- programhantverk (5p)
- valfria kurser (15p)
- kreativt projekt + innovation (10p)
- industriellt projekt + industrikontakt (10p)

Vi ser ett antal olika utbildningsmoment som ska lyftas fram i utbildningen. Metodik för individuellt arbete och mindre grupp är en viktig del av software craftsmanship, t ex hantverksregler och agil metodik. Djup förtrogenhet med grundläggande utvecklingsverktyg och programmeringsspråk. Här kommer en större inventering av nuläget och framtid göras senare. Exempel på utvecklingsverktyg är texteditor, IDEer, byggverktyg, versionshantering. Egen erfarenhet att arbeta och lära sig av dominerande utvecklingsplattformar och teknikfronter med speciellt fokus på öppen källkod: exempel är Linux, Java, Apache. Kännedom och erfarenhet av dominerande tillämpningsområden (inkl målplattformar): exempel är enterprise computing, online-tjänster, dotnet, c# och produkter för Windows-plattformen.

## Väggkarta och roller

**Hösten 2007**: första kullen studenter börjar.

**Industrikoppling**: En viktig komponent för att det tillämpningsnära inslaget ska fungera är nära koppling till olika företag.

**Lärarkompetens för IP**: IP måste kontinuerligt förnya och vidareutveckla lärarnas egen programmerarkompetens. Att personer har egen erfarenhet av konkret programmeringsarbete bör premieras vid val av kurser och lärare.

Utbildningen var ursprungligen ett förslag från avdelningen HCS på IDA. Fortsatt utformande av utbildningen bör dock ske i en mer gemensam arbetsgrupp där intressenter inbjudas att delta i utformningen av olika detaljer.

## En innovations och tillämpningsdriven utbildning

IP är en innovations och tillämpningsdriven utbildning. Teori-delen hålls relativt liten och driven av behov i tillämpningarna, fokus är på direkt omsättbar hantverkskunnande. Tillämpningarna **väljs med största omsorg**, extremt viktigt moment. Genomgång och utlärande av olika tekniker och verktyg är mer en form av grötrecepts-kunskap som dock måste läras ut/sjävläras i ganska stor omfattning (exakt hur kan man diskutera).

Den pedagogiska modellen för utbildningen är att ha en stor del projektarbete vid sidan av kurser. Projektarbetet sker till stor del enskilt eller i grupp beroende på uppgift.Handledning och avstämningar görs till stor del i stor grupp, t ex som halv/heldag-workshops. Samtidigt betonas och uppmuntras att arbete sker i utbyte med andras i studenternas egen community, inspirerat bl a av arbetssätt från Open-Source-rörelsen.

Det är **projekt delen** tredje året som kan ses som **målet** för utbildningen. Projekten ska fokusera på att vara "riktiga" projekt och leda fram till fungerande system/programvara/produkt/etjänster. Projekten ska i största möjliga mån ankyta till existerande företag, brancher och tillämpningar där det finns intresse och arbetstillfällen.

En preliminär lista över lämpliga tillämpningsområden:

- skriptad filhantering och systemadm
- Program för intranät
- Enklare informationstjänster via gui
- enklare webb- och telefonitjänster
- spelprogram
- konsumentprodukter
- enterprise computing
- nätverksprogrammering
- etc.

Listan bör vara föränderlig och svara mot existerande marknadsbehov på några års sikt.

Projektet ger studenten en **portfölj** av system som man sen kan visa upp när man söker jobb, vid sidan av examensbetyg. Projektarbetet ger egen erfarenhet av att ha deltagit i "hela processen" till färdig programvara och personer får jobba mycket individuellt (om än lära av varandra). Alla ska programmera själva inte bara sitta och titta på, vi har strikta regler som blir mer avancerade gradvis som de måste följa till punkt och pricka.

Områdena Software Craftsmanship och Open Source kommer ses som vägledande för hur den grundläggande pedagogiken med hur man lär upp personer i hantverket att bli en bra programmerare. Där finns bl a metodik som betraktar den individuella utvecklingen i stil med det traditionella hantverket (därav namnet craftsmanship). Här ser man individens utveckling i tre steg, från **apprentice** (lärling), till **journeyman** (gesäll) till **master** (mäster). Dessa tankegångar har testas i IDA-kurser inom Pragmatisk programmering idag (ETE247 och ETE248).

## Att få igenom studenter

Det allra viktigaste ekonomiska målet är att få igenom studenter. Här får man fundera extra noga. Vi har en avvägning mellan samläsning med C/D/DI och det faktum att detta är en mer praktiskt orienterad utbildning som bör fokusera på att lära studenterna att arbeta med hög kvalitet praktiskt först och främst.

Utbildningen kommer tilltala **nya grupper** än de som redan täcks av C/D/DI. Att undersöka och gradvis utvidga utbildningen mot nya typer av personer och intressen bör ingå som del i utbildningens motto. En grundbult är att den ska lära ut teknik utan att förutsätta traditionell matematisk och analytisk färdighet. Istället bör den förlita sig **på andra sidor** hos de sökande, t ex **kreativa, praktiska, ekonomiska, etiska, estetiska** och **sociala** förmågor och intressen. Men syftet är fortfarande att vi utbildar programmerare, dvs alla moment baseras på riktiga implementerade system.

Erfarenhet från tidigare kurser inom pragmatisk programmering är att det finns mycket tid/pengar att spara genom att använda Internet-relaterade tekniker t ex forum för den tekniska delen av undervisningen. Där finns också många möjligheter att uppmuntra till stöd student-student. Undervisningen kan på så vis till stor del bestå av **självinläring** i en community som finns i en bra mix både på campus och online.

## Varför behövs en ny utbildning?

Den stora skillnaden mellan IP och C/DI/D/IT är större fokus på praktiskt kunnande inom programkonstruktion. Detta motiverat av att:

- det finns en stor stabil arbetsmarknad för personer med färdigt praktiskt kunnande inom programmering
- teoretiskt och matematiskt kunnande leder inte automatiskt till gott praktiskt kunnande, och är inte heller en nödvändig förutsättning för det (Se t ex Steve McConnells diskussioner i ämnet).
- personer med fallenhet för programmering behöver inte nödvändigtvis ha intresse av eller fallenhet för mer teoretiska ämnen. De personer som saknar sådant intresse bör istället få utveckla sin hantverkskunskap som om det utvecklas kan bli mycket värdefullt för arbetsmarknad och den personens kärnkompetens.

Skillnaden mellan IP och en utbildning hos ett utbildningsföretag är att den grundläggande yrkesförberedande karaktären på utbildningen och att dess fokus på en konkurrensmässig kärnkompetens. Utbildningsföretagens kurser är ofta kortare och mindre sammanhängande. Deras fokus ligger på att ge korta konkurrensfördelar till personer och företag, med fokus på det senaste som efterfrågas på marknaden. Även IP bör beakta marknadsbehov men på en medellång sikt, snarare än kort. IPs fokus på kärnkompetens gör också att vi till stora delar kommer ta upp saker som ger en grundkompetens snarare än en konkurrensfördel. Dvs IP ger personer ett inträde till marknaden, snarare än fördelar mot konkurrenterna.

Skillnaden mellan IP och en programmeringsutbildning på gymnasienivå ("yrkesinriktad utbildning") är att den högre kognitiv mognad som studenten uppnått efter gymnasiet gör att vi kan börja arbeta på en högre nivå. Detta är troligen helt nödvändigt, eftersom programmering till skillnad från fysiskt orienterade hantverk (t ex snickare) kräver en stor dos av intellektuell förmåga vilket en gymnasieutbildning tillhandahåller som ett minimum.

## Exempel på existerande kurser för IP

Här kommer ett första förslag/exempel på hur **existerande kurser** anknyter till i olika block:

- språk+plattformar:
  - Datorsystem och imperativ programmering, 3 p
  - Perspektiv på datateknik/datavetenskap, 4 p
  - Programmering, Lisp och funktionell programmering, 4 p
  - TDDC37 Programmering, Java och objektorienterad programmering
- övrig teori/matematik:
  - HKGB13 Formella och matematiska grunder för Kognitionsvetenskap, 5p
- programmeringsnära MDI:
  - HKGB11 Användbara system 2p
  - Interaktionsprogrammering 5p
  - TDDB91 Människa-datorinteraktion, 3 p
- operativsystem och systeminstallation:
  - TDDI05 Systeminstallation
- informationssystem och databaser
  - TDDB48 Databasteknik, 5 p
- media, konsumentprodukter och spel:
  - TDDC29 Design och programmering av datorspel, 5 p
  - TNM079 Modellering och animering, 4 p
- Internet:
  - TDDB64 Web programming and Interactivity
- Programhantverk:
  - ETE246-EET247 Pragmatisk programmering

Hur pass mycket dessa kurser behöver anpassas, bör utredas vidare och blir delvis en ekonomisk avvägning.

## Förslag på viktiga nya kurser

Förslag på **nya kurser** som är speciellt centrala för IP:

- inom språk+plattformar:
  - fördjupning av kunskap om att arbeta via Linux och för Windows, samt att arbeta via/med öppen källkod
  - inledande programmeringskunskap, förslagvis baserat på Steve McConnells bok Code Complete
- inom informationssystem:
  - Design av objekt-orienterade informationssystem, t ex baserat på Patterns of Enterprise Application Architecture Martin Fowler. i HIBC20 Programmering II (för statistiker) gjorde vi en sådan kurs i miniatyr.
- Projektserie
  - En genomtänkt projektkursorganisation som en sammanhängande tråd i utbildningen, vilket leder till en "studentportfölj". Här kan man ta fasta på

lärlingsbegreppen från software craftsmanship som stöd. Erfarenheter med att undervisa i "praktisk programmering" från Pragmatisk programmeringskurserna och andra projektkurser som PUM ger en utgångspunkt att arbeta ifrån.