

# Studieordning for diplomuddannelsen i informationsteknologi

April 2007 [v3]

1	Introduktion.....	2
2	Formål .....	2
3	Indhold .....	2
4	Adgangskrav .....	3
5	Eksaminer.....	3
6	Studieplan.....	3
6.1	Formelle modeller for programmering .....	4
6.1.1	Diskret matematik .....	4
6.1.2	Kontraktbaseret programmering .....	5
6.1.3	Regularitet og automater.....	6
6.2	Programmering af store objekt-orienterede systemer .....	6
6.2.1	Værktøjer og teknikker for store objekt-orienterede systemer .....	6
6.2.2	Arkitektur, patterns og frameworks .....	7
6.2.3	Programmeringsprojekt i store objekt-orienterede systemer .....	8
6.3	Hovedopgave .....	9
7	Merit og dispensation.....	10

# 1 Introduktion

Ved Aarhus Universitet udbydes en diplomuddannelse i Informationsteknologi, linjen i Softwarekonstruktion.

Uddannelsen udbydes i henhold til Undervisningsministeriets bekendtgørelse om diplomuddannelsen i informationsteknologi under IT-Vest samarbejdet (Bekendtgørelse nr. 744 af 22. august 2001).

Uddannelsen hører under Studienævn A ved Det Naturvidenskabelige Fakultet, Aarhus Universitet.

## 2 Formål

Diplomuddannelsen i informationsteknologi har til formål at forbedre voksnes erhvervskompetence og personlige kompetence inden for informationsteknologi. Uddannelsen skal give de studerende en teoretisk, analytisk og praktisk indsigt i informationsteknologi. Uddannelsen skal herudover sætte de studerende i stand til at medvirke ved udvikling og implementering af ny informationsteknologi i virksomheder, institutioner m.v., herunder:

- 1) at forstå og anvende teorier og metoder inden for de informationsteknologiske discipliner,
- 2) at kunne indgå aktivt i forbindelse med planlægning, udvikling og ibrugtagning af informationsteknologiske systemer samt tilrettelæggelse af arbejdsgange i forbindelse med anvendelse af disse systemer og
- 3) at kritisk kunne deltage i vurdering af hvilke muligheder og begrænsninger der er for anvendelsen af informationsteknologien.

## 3 Indhold

Uddannelsen kvalificerer til at løse informationsteknologiske problemstillinger, herunder anvendelse og vurdering af væsentlige informationsteknologiske metoder, teorier, teknikker og arbejdsformer, der knytter sig til udvikling og konstruktion af IT-systemer og software.

Uddannelsen dækker følgende fagområder, der udgør uddannelsens kerneområder:

- 1) Programmering, herunder specielt objektorienteret programmering.
- 2) Algoritmer og datastrukturer.
- 3) Arkitekturer for datamater og datamatnetværk.
- 4) Distribuerede systemer med fokus på deres egenskaber og programmering.
- 5) Softwarearkitektur omfattende såvel klassiske arkitekturer som frameworks og design patterns.
- 6) Strukturering af information og data herunder databaser og database management.
- 7) Metoder og arbejdsformer i softwareudvikling.
- 8) Ledelse og organisering af softwareudviklingsprojekter.

Linjen i Softwarekonstruktion kan tillige dække bl.a. følgende fagområder:

- 1) Menneske-maskin-grænseflader, herunder brugbarhed af software

- 2) Sikkerhed og kryptering
- 3) Test og verifikation
- 4) Dokumentation
- 5) Konfigurationsstyring og versionsstyring
- 6) Projektøkonomi
- 7) Kvalitetsstyring og kvalitetsstyringssystemer

## 4 Adgangskrav

En uddannelse som datamatiker, datanom, akademiuddannelse i informationsteknologi, eller tilsvarende kvalifikationer og minimum to års relevant erhvervs erfaring efter den adgangsgivende eksamen.

## 5 Eksaminer

Uddannelsen består af et antal prøver. Hver prøve skal bestås for sig. Beståede prøver kan ikke tages om.

Prøverne bedømmes enten med intern eller ekstern censur. Interne prøver bedømmes af eksaminator samt ingen eller flere interne censorer. Eksterne prøver bedømmes af eksaminator og en eller flere ministerielt beskikkede censorer.

Ved prøverne anvendes enten karakterskala efter gældende regler eller bedømmelsen bestået/ikke-bestået.

Bedømmelsen af en prøve skal ved sommereksamen foreligge senest 1. juli, ved vintereksamen senest 1. februar.

En studerende kan højst tre gange indstille sig til den samme prøve. Studienævn A ved det Naturvidenskabelige Fakultet kan ved dispensation tillade indstilling for fjerde gang, hvis særlige forhold gør sig gældende.

Aarhus Universitet udsteder bevis for gennemført uddannelse. Til studerende, der forlader uddannelsen uden at have gennemført den, udsteder Aarhus Universitet bevis for beståede dele af uddannelsen.

## 6 Studieplan

Uddannelsen er tilrettelagt med et normalforløb svarende til 6 semestre med en belastning på 10 ECTS på hvert semester. De tre første semestre (seks kvarterer) dækkes af to obligatoriske fagpakker. En fagpakke er på 15 ECTS og dækker et overordnet tema, men er inddelt i mindre enkeltfag af 5 ECTS belastning. De tre sidste semestre dækkes af perspektivfag og hovedopgave.

## Samlet oversigt over uddannelsens fag/prøver:

Fagpakke / enkeltfag	Censur	Vægtning	Bedømmelse
<i>Formelle modeller for programmering</i>			
Diskret matematik	Intern	5 ECTS	Bestået/ikke-bestået
Kontraktbaseret programmering	Intern	5 ECTS	7-skala
Regularitet og Automater	Ekstern	5 ECTS	7-skala
<i>Programmering af store objekt-orienterede systemer</i>			
Værktøjer og teknikker	Intern	5 ECTS	Bestået/ikke-bestået
Arkitektur, patterns og frameworks	Intern	5 ECTS	7-skala
Programmeringsprojekt	Ekstern	5 ECTS	7-skala
<i>Perspektivfag</i>		15 ECTS	
<i>Hovedopgave</i>	Ekstern	15 ECTS	7-skala

### Perspektivfag

I uddannelsen indgår perspektivfag af et omfang svarende til 15 ECTS. Perspektivfagene varierer år for år og dækker forskellige emner indenfor de områder, som er beskrevet i bekendtgørelsen og beskrevet i afsnit 3. Eksamensform og censur vil være defineret ud fra disse kurser. Perspektivfag kan erstattes af andre kursustilbud, hvis dette aftales med uddannelsens faglige koordinator og godkendes i studienævn A.

## 6.1 Formelle modeller for programmering

Fagpakken består af tre enkeltfag.

### 6.1.1 Diskret matematik

#### Læringsmål

Deltagerne skal ved afslutningen af kurset kunne:

- **definere** og **referere** den basale terminologi.
- **anvende** begreber og teknikker på givne problemstillinger.
- **forklare** og **udføre** beviser for simple problemstillinger.

#### Indholdsbeskrivelse

- Tal (elementær talteori, talsystemer, primtal, fibonacci tal, etc.)
- Konkrete funktioner (specielt logaritme- og eksponentialfunktioner)
- Matematiske strukturer (mængder, multimængder, sekvenser, funktioner, relationer, grafer)
- Propositionslogik (booleske værdier, operatorer, regneregler)
- Prædikatalogik (typedede variabler, kvantorer, regneregler)
- Bevisførelse (induktion, modstrid, reduktion, etc.)

#### Eksamensform

En mundtlig prøve.

*Vægtning*  
5 ECTS

## 6.1.2 Kontraktbaseret programmering

### *Læringsmål*

Deltagerne skal ved afslutningen af kurset kunne:

- **definere** og **referere** den basale terminologi.
- **formulere** og **skrive** kontrakter i form af programudsagn og funktionelle specifikationer for metoder og klasser baseret på prædikatlogik.
- **anvende** løkke- og klasseinvarianter til systematisk udledning af kode i metoder og klasser.
- **forklare** og **vurdere** sammenhængen mellem programudsagn og programkode.

### *Målbeskrivelse*

Målet med kurset er at den studerende lærer grundlaget for systematisk, kontraktbaseret konstruktion af programmer.

Efter modulet vil den studerende i stand til at læse og skrive kontrakter i form af programudsagn og funktionelle specifikationer for metoder og klasser baseret på prædikatlogik, redegøre for sammenhængen mellem programudsagn og programkode samt gøre brug af løkke- og klasseinvarianter til systematisk udledning af kode i metoder og klasser.

### *Indholdsbeskrivelse*

- Specifikationer vha. prædikatlogik
- Specifikation versus implementation
- Programudsagn ("assertions") og gyldighed
- Sammenhæng mellem udsagn og kode
- Løkkeinvarianter (checkliste for løkker)
- Klasseinvarianter
- Systeminvarianter (UML m.m.)
- Praktisk programmering med udsagn (assert-mekanismer)
- Udsagn og test

### *Forudsætning*

Enkeltfaget forudsætter at kursisten har fulgt faget "Diskret matematik".

### *Eksamensform*

En mundtlig prøve.

*Vægtning*  
5 ECTS

### 6.1.3 Regularitet og automater

#### *Læringsmål*

Deltagerne skal ved afslutningen af kurset kunne:

- **definere** den basale terminologi (strenge, sprog, klasser af sprog, samt basale operationer på disse).
- **beskrive** basale abstrakte sprogformalismer (regulære udtryk, endelige automater, regulære grammatikker, kontekstfri grammatikker) - fra intuitivt niveau og konkrete eksempler til formel notation og generelle definitioner.
- **beskrive** egenskaber ved formalismerne, bl.a. ækvivalens, begrænsninger og beslutningsprocedurer.
- **forklare** og **udføre** algoritmer, der oversætter mellem formalismerne eller afgør beslutningsproblemer - fra konkrete eksempler til generelle og formelle beskrivelser.
- **bevise** og **analysere** egenskaber ved formalismerne (ved hjælp af konstruktive beviser og induktionsbeviser) - fra intuitivt niveau til formelle detaljer.

#### *Indhold*

Kurset vil dække følgende emner:

- endelige automater, regulære udtryk og regulære grammatikker
- egenskaber ved disse, bl.a. ækvivalens og begrænsninger
- relation til mere generelle beregningsmodeller som kontekst-fri grammatikker og Turing-maskiner
- bevisteknikker
- eksempler på praktiske anvendelser

#### *Forudsætning*

Enkeltfaget forudsætter at kursisten har fulgt faget "Diskret matematik".

#### *Eksamensform*

En mundtlig prøve.

#### *Vægtning*

5 ECTS

## 6.2 Programmering af store objekt-orienterede systemer

Fagpakken består af tre enkeltfag.

### 6.2.1 Værktøjer og teknikker for store objekt-orienterede systemer

#### *Læringsmål*

Deltagerne skal ved afslutningen af kurset kunne:

- **beskrive** objekt-orienterede systemer ved hjælp af UML og arkitekturviews.
- **beskrive, anvende, og programmere** avancerede sprogkonstruktioner til concurrency, undtagelseshåndtering, genericity, og distribueret programming.
- **definere** og **anvende** databaser, relationel algebra samt query sprog (f.eks. SQL) på mindre problemstillinger.
- **beskrive** værktøjer til håndtering af store objekt-orienterede programmer (f.eks. build-management, software konfigurationsstyring, test-management), **forklare** deres virkemåde og **anvende** dem på givne problemstillinger.
- **definere** systematisk test teknikker, og **anvende** dem på mindre problemstillinger.

### *Indhold*

Kurset vil præsentere teori og begrebsdannelse inden for et antal områder, samt indeholde et større antal praktiske design og programmeringsopgaver som underbygning og illustration af teorien.

Kurset vil dække emnerne (eksempler på underemner angivet i parentes):

- Grafiske notationer for objekt-orienterede programmer (UML, views).
- Sprogkonstruktioner (concurrency, genericity, pakker, undtagelseshåndtering)
- Teknikker (systematik test, test-dreven udvikling, versionskontrol, build-management, persistens i relationelle databaser, distribueret programmering)
- Værktøjsstøtte (integreerede udviklingsmiljøer, testomgivelser, versionsstyring, build-management)

### *Eksamensform*

En mundtlig prøve.

### *Vægtning*

5 ECTS

## 6.2.2 Arkitektur, patterns og frameworks

### *Læringsmål*

Deltagerne skal ved afslutningen af kurset kunne:

- **Beskrive** softwarearkitektur
- **Beskrive** og **klassificere** kvalitetsattributter for softwarearkitektur
- **Analysere** og **sammenligne** parametrisk, polymorph, og kompositionel design.
- **Anvende** og **programmere** teknikker til variabilitetshåndtering på en konkret problemstilling.
- **Beskrive** og **implementere** gænge design patterns.
- **Beskrive** og **konstruere** frameworks.
- **Forklare, evaluere** og **anvende** patterns for test-dreven udvikling
- **Anvende** værktøjer til dokumentation og styring af store systemer såsom UML, Ant og JUnit.
- **Anvende** Java eller et andet moderne objekt-orienteret sprog.
- **Diskutere** og **perspektivere** de enkelte kursusemner til hinanden.

### *Indhold*

Kurset vil præsentere teori og begrebsdannelse inden for et antal områder, samt indeholde et større antal praktiske design og programmeringsopgaver som underbygning og illustration af teorien.

Kurset vil dække emnerne:

- Responsibility-driven design og variabilitetshåndtering
- Principper for fleksibel og genanvendelig software
- Software kvalitet, metrikker og måleteknikker
- Design og arkitektur mønstre/patterns
- Frameworks
- Test-dreven udvikling

### *Forudsætning*

Enkeltfaget forudsætter at kursisten har fulgt faget ”Programmering af store objekt-orienterede systemer: Værktøjer og teknikker”.

### *Eksamensform*

En mundtlig prøve.

### *Vægtning*

5 ECTS

## **6.2.3 Programmeringsprojekt i store objekt-orienterede systemer**

### *Læringsmål*

Deltagerne skal ved afslutningen af kurset kunne:

- **Anvende** begreber, teknikker og praktiske metoder fra fagpakken til at **analysere, designe**, og **implementere** en pålidelig og fleksibel løsning for en given problemstilling af større kompleksitet.
- **Evaluere** og **diskutere** teknikkers, metoders, og fundamentale begrebers styrker og svagheder i den konkrete sammenhæng.
- **Beskrive** og **dokumentere** løsningen og analyser klart og utvetydigt på skrift.

### *Indhold*

Projektet formuleres som en central, bunden, opgave samt krav om en eller flere udvidelser som vælges af kursisten.

Projektet vil tage udgangspunkt i et antal centrale problemstillinger:

- Objektorienteret design og designdokumentation
- Flexibilitet gennem brug af variabilitetshåndtering, design- og arkitektur patterns
- Anvendelse og konstruktion af frameworks.
- Systematisk test og test-dreven udvikling
- Værktøjsstøtte, build-management, versionskontrol.

Desuden vil kursisten kunne inddrage et eller flere andre aspekter i projektet, såsom relationelle databaser, distribution og concurrency, o.a.

#### *Forudsætning*

Projektet forudsætter at kursisten har fulgt fagene ”Programmering af store objekt-orienterede systemer: Værktøjer og teknikker” samt ”Programmering af store objekt-orienterede systemer: Arkitektur, patterns og frameworks”

#### *Eksamensform*

Et skriftligt projekt.

#### *Vægtning*

5 ECTS

### **6.3 Hovedopgave**

#### *Formål*

Ved udarbejdelsen af hovedopgaven skal den studerende demonstrere fortrolighed med almindelige principper for videnskabelig metode og færdighed i at anvende metoder og teorier til selvstændigt at afgrænse og behandle problemstillinger inden for området softwarekonstruktion.

#### *Vejledning*

Uddannelsens hovedopgavestudium sker under vejledning. Ved starten på hovedopgavestudiet aftales projektets titel samt tidspunkt for aflevering af hovedopgave. Med mindre andet er fastlagt, er det den studerendes ansvar at finde en vejleder, men den faglige koordinator for uddannelsen er naturligvis behjælpelig hermed.

Hovedopgaven kan tage udgangspunkt i problemstillinger, der er defineret i samarbejde mellem den studerende, vejlederen, og eksterne parter såsom virksomheder eller institutioner.

#### *Eksamensform*

Prøven er et frit skriftligt arbejde og en mundtlig prøve (et forsvar). En hovedopgave kan være individuel eller i gruppe med op til tre studerende. Ved gruppebesvarelse skal mindst halvdelen af den enkelte studerendes bidrag kunne gøres til genstand for individuel bedømmelse.

Den mundtlige prøve (er individuel og) finder sted ved at de(n)studerende efter aflevering af det skriftlige arbejde og senest en uge inden selve forsvaret modtager en opgave formuleret med udgangspunktet i det skriftlige arbejde. Forsvaret udformer sig som en besvarelse af den udleverede opgave fulgt af en samtale.

Censur og bedømmelse: ekstern censur efter 7-skalaen. Der gives én samlet karakter efter 7-skalaen for det skriftlige arbejde og den mundtlige prøve. Der medvirker ekstern censor ved bedømmelsen; vejleder fungerer som eksaminator.

*Vægtning*  
15 ECTS

## **7 Merit og dispensation**

Studienævn A ved det Naturvidenskabelige Fakultet kan godkende, at prøver bestået under andre uddannelser kan erstatte prøver i uddannelserne beskrevet i studieordningen. Studienævnet kan endvidere tillade individuelle ændringer i studieprogrammet, når dette ikke strider med uddannelsens målsætning eller bestemmelser fastsat i bekendtgørelserne for uddannelserne.