

Rapportskrivning

Projektdokumentation (bilag) Procesbeskrivelse (bilag) og Projektrapport

Torben Gregersen - tg@ece.au.dk

Agenda

Procedure for PRJ3-eksamen

Overblik over den samlede PRJ3-dokumentation

Projektdokumentation

Procesbeskrivelse

Projektrapport

Procedure for PRJ3-eksamen

PRJ3-eksamen foregår således:

Censor og eksaminator vurderer det afleverede materiale og bliver enige om et udgangspunkt for en karakter for hver enkelt studerende.

De studerende i gruppen præsenterer deres projekt i fællesskab f. eks. vha. slides og en demonstration af projektet.

I stedet for en fysisk demonstration kan film eller fotos anvendes.

Der afsættes 20-25 minutter til dette.

Herefter interviewes hver enkelt studerende i 12-14 minutter. Den studerende skal demonstrere indgående indsigt i den del af projektet han/hun har været ansvarlig for. Han/hun skal yderligere have indsigt i projektet som helhed.

Efter endt eksamination bliver censor og eksaminator enige om den karakter, der skal gives til hver enkelt studerende. Karaktergivningen gives primært ud fra det afleverede materiale, men præsentationens kvalitet og interviewets kvalitet har også indflydelse på karakteren.

Hvis blot 1 studerende i gruppen ønsker at modtage karakter + feedback individuelt uden at den øvrige del af gruppen er til stede, skal resten af gruppen også modtage karakter + feedback individuelt. Ellers kan hele gruppen modtage deres individuelle karakter + feedback i samlet flok.

Vejledning til udfærdigelse af projektdokumentation

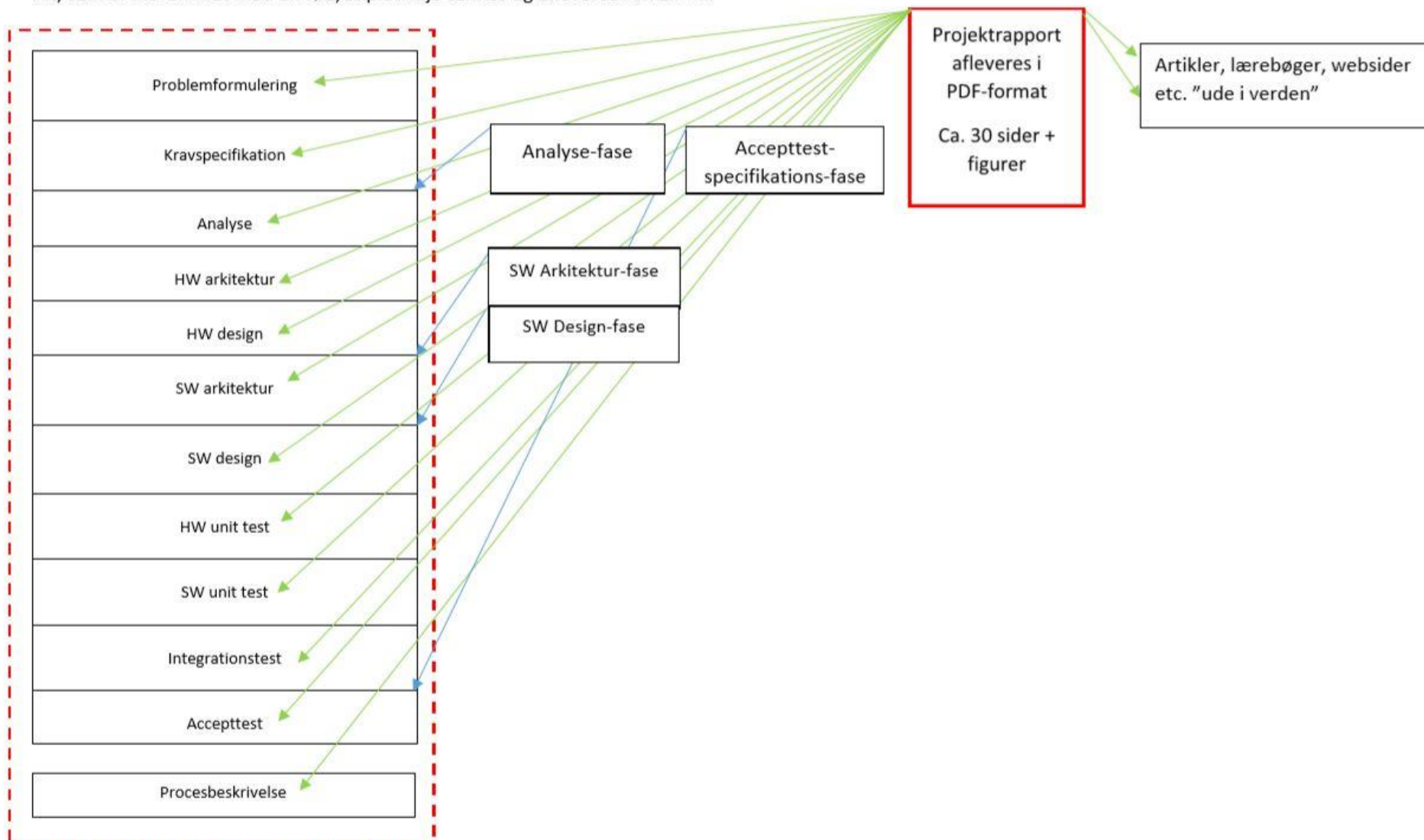


Projektdokumentation (Bilag)

Del-dokumenterne kan afleveres i ét samlet dokument eller afleveres i separate filer.
 Alt, som er indrammet med en rød, stipleet linje samles og afleveres i en ZIP-fil.

Projektrapport

Grønne pile markerer henvisninger



Problemformulering og kravspecifikation

Skal tilsammen tydeligt beskrive **hvad** der skal udvikles i projektet.

Problemformuleringen beskriver i prosaform hvad projektets idegrundlag er, støttet vha. nødvendige illustrationer.

Kravspecifikationen præciser problemformuleringens beskrivelse ved hjælp af:

- Aktør/Kontekst diagram
- Funktionelle krav
 - Use Case figur – eller Epic figur - eller User Story figur
 - Use Case tabeller eller Epic beskrivelser kombineret med User Story beskrivelser
- Ikke funktionelle krav (tekniske specifikationer (mål, vægt, hastighed etc.))
- MoSCoW
- FURPS (Kan dække kvalitetsparametre for hele systemet – både mht. funktionelle og ikke funktionelle krav)
- Brugergrænseflade-beskrivelse, dvs. layout af skærbilleder, trykknapper etc.

På basis af **problemformuleringen** og **kravspecifikationen** er der ikke længere tvivl om hvad systemet skal kunne!

Analyse = "proof of concept" på vigtige beslutninger i projektet

Eksempel-1:

3.3.1 Udviklingsboard: Gateway

Da gatewayen både skal fungere som en node i mesh netværket og skal kunne kommunikere med applikationen er det nødvendigt for denne at kunne forbinde til internettet. Da der kræves et eksternt modul til at forbinde nRF boardet til nettet samt for at kunne udvikle requesthåndtering med et højniveau framework, som .NET, skal GatewayNoden sammenkobles med et andet udviklingsboard.

	RaspberryPi 3 Model B [36]	BeagleBone Black Wireless [44]
CPU	Quad-core 64-bit ARM Cortex A53 1,2GHz	Octavo Systems OSD3358 ARM Cortex-A8 1GHz
Networking	Ethernet, 802.11n wireless	802.11b/g/n
Low Level Peripherals	SPI, I2C, UART	SPI, I2C, UART

Tabel 5: Specifikationer for gateway kandidater

På tabel 5 ses de to kandidater, der var taget i betragtning til gatewayen. De vigtigste egenskaber der er taget i betragtning er vist i tabellen. De to typer boards er meget ens i deres specifikationer, dog har RaspberryPi'en en anelse kraftigere processor og muligheden for at forbinde til nettet via Ethernet. Grundet gruppens kendskab til RaspberryPi'en samt boardets interface muligheder til nRF boardet, som beskrives nærmere i afsnit 3.3.2, er denne blevet valgt til den anden halvdel af Gateway komponenten.

Analyse = "proof of concept" på vigtige beslutninger i projektet

Eksempel-2:

3.3.3 Framework: ASP.NET Core

I afsnit 3.3.1 vælges en RaspberryPi til Gateway-delen, som interfacer med omverdenen. For at smartphone applikationen kan interagere med mesh netværket eksponeres et web API, som hostes på RaspberryPi'en. Dette web API har til ansvar at modtage og behandle kommandoer fra applikationen og videresende dem til mesh netværket via I2C forbindelsen, nævnt i afsnit 3.3.2.

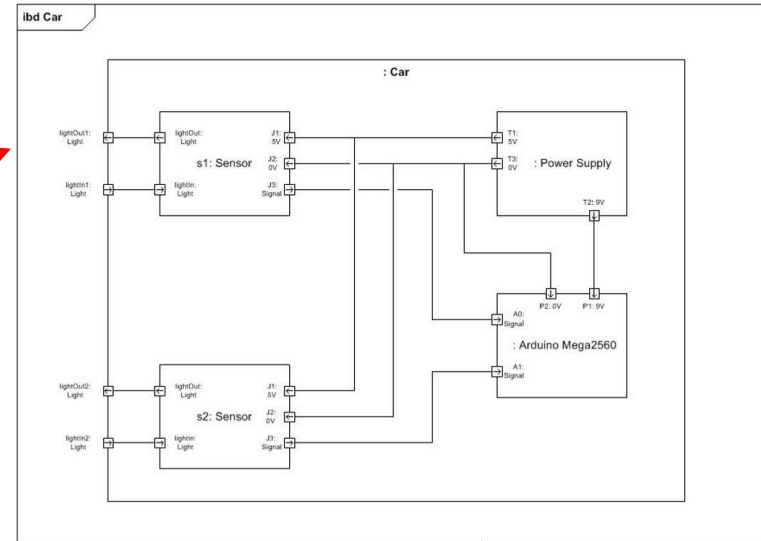
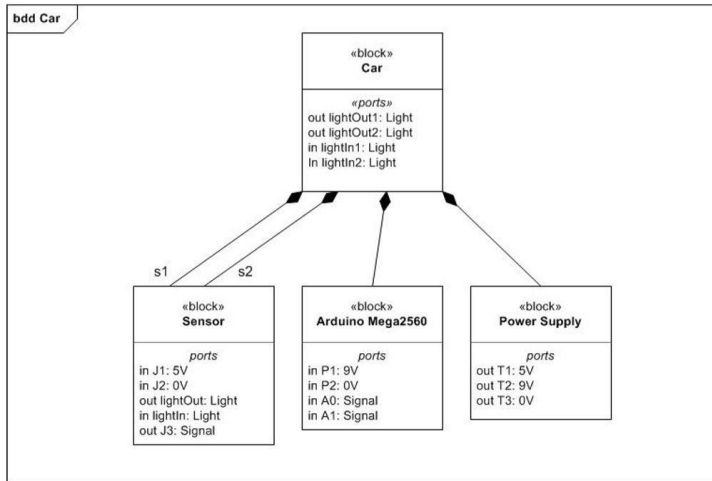
På RaspberryPi'en skal der hostes et web API, som giver applikationen tilgang til netværket. Der udstilles et RESTful API[47], som er en nyere tilgang til web service interfaces end alternativer som SOAP. API'et definerer en klar grænseflade til applikationen, som bruges til at sende HTTP requests op imod. REST er valgt, da denne bruger mindre båndbredde, har bedre performance og kan let integreres op mod andre applikationer, hvis dette skulle blive nødvendigt i fremtiden. Derudover tillader REST data formater som JSON, som er lettere at arbejde med i denne sammenhæng, da disse ville kunne mappes direkte ud til Java objekter i applikationen.

Web API'et udvikles ved brug af ASP.NET Core[48] da der er blevet stiftet bekendskab til frameworket i løbet af uddannelsens forløb. ASP.NET Core tilbyder cross-platform udvikling samt en letvægtig, høj performance HTTP requestline til håndtering af request kaldene og Parameter Binding[49], hvilket gør det lettere at håndtere requests da requestdata kan mappes direkte ud til metoderne.

ASP.NET Core gør brug af en underliggende Kestrel webserver, hvilket er en letvægtig og kraftfuld webserver[50]. Kestrel er ikke en fuldt udstyret webserver, hvilket betyder at denne er hurtigere end andre typer som IIS[51]. Kestrel er default webserveren for ASP.NET Core applikationer. Der er dog mulighed for at udskifte den underliggende webserver, men IIS servere, kræver dyre Microsoft licenser og er beregnet til større opgaver end der kræves af prototypen. Derfor vælges Kestrel da denne er hurtig, open-source og egner sig til opgaven.

HW Arkitektur vha. SysML

Eksempel:

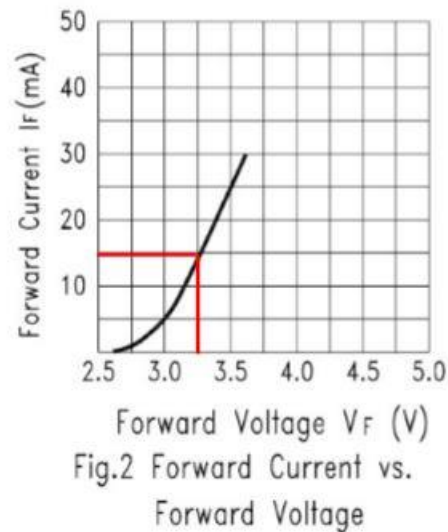


Signalnavn	Funktion	Område	Port 1 (source)	Port 2 (destination)	Kommentar
ground	Reference til analog spænding		Power Supply, T3	Sensor1, J2 Sensor2, J2 Arduino Mega2560, P2	Stel
sensorPower	Forsyningsspænding	4,9-5,1V	Power Supply, T1	Sensor1, J1 Sensor2, J1	
arduinoPower	Forsyningsspænding	7-12V	Power Supply, T2	Arduino Mega2560, P1	
lightOut	Fysisk lys		lightOut		Lys ud til refleksbrik
lightIn	Fysisk lys			lightIn	Lys ind fra refleksbrik
sensor1Signal sensor2Signal	Indikerer modtaget fysisk lys	0-5V	Sensor1, J3 Sensor2, J3	Arduino Mega2560, A0 Arduino Mega2560, A1	Rise time < 3.5ms

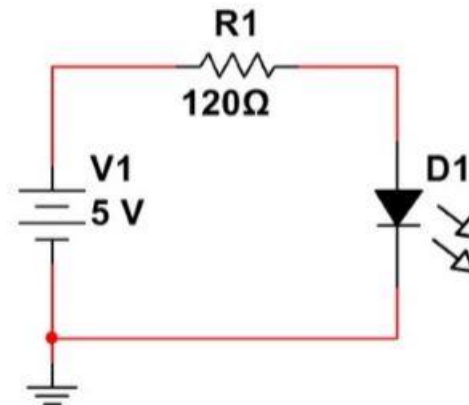
HW Design


Eksempel:

LED fabrikat: LITEON
Part No. : LTW-2S3D7-012A



Strømmen gennem lysdioden D1 besluttet til at være 15 mA
Ifølge databladet giver dette et spændingsfald over D1 på 3,25V
Forsyningsspændingen V1 er 5V
Spændingsfaldet over R1 er da: $V_{D1} = 5V - 3,25V = 1,75V$
R1 kan derfor beregnes således: $R1 = 1,75V / 15mA = 116,7 \Omega \sim \underline{120 \Omega}$



Software Arkitektur & Design – dokumenteret vha. N+1 View (beskrevet af Craig Larman i bogen "Applying UML and Patterns")						
Logical View (som i I2ISE-kursus)	Process View	Deployment View	Data View	Security View	Implementation /Development View	
<p>A) Domænemodel tegnes/beskrives.</p> <p>B) Systemsekvensdiagrammer tegnes/beskrives for alle use cases / user stories. De beskriver interaktionen mellem aktører og CPU'er i systemet. Såvel OK-scenarier og extensions/exceptions beskrives i nødvendigt omfang.</p> <p>C) Applikationsmodeller beskrives for alle use cases / user stories. De beskriver interaktionen mellem aktører og de enkelte objekter i applikationen på hver enkelt CPU i systemet. Såvel OK-scenarier og extensions/exceptions beskrives i nødvendigt omfang. Formålet er at "opdage" de vigtige klasser/metoder og parametre i systemet"</p> <p>1) Beskriv konceptuelle klasser i use case'n med angivelse af <<controller>>, <<boundary>> og <<domain>></p> <p>2) Beskriv interaktionen mellem disse klassers instanser (objekter) i et applikationsmodel-sekvensdiagram.</p> <p>3) Overfør resultatet fra [2]) til et statisk klassesdiagram for den givne use case / user story. Dette klassesdiagram er applikationsmodellen for use casen på CPU'en.</p> <p>4) Overfør eventuelt (hvis der er plads) alle "use case"-klasse-diagrammer til ét samlet klassesdiagram. Dette er den endelige applikationsmodel for CPU'en.</p> <p>Punkt 1-4 gennemføres for hver CPU i systemet.</p>	<p>Systemets Concurrency beskrives her, dvs. beskrivelse af tråde/processer i systemet – og samspillet mellem disse.</p> <p>Samspillet kan f.eks. være anvendelsen af mutex, semaphore, mailbox, pipes etc. for at opnå sikker tilgang til kritiske data – med det formål at undgå crash i systemet.</p> <p>Samspillet kan også være en beskrivelse af opstartsækkefølge for de enkelte CPU'er i systemet.</p>	 <p>Anvendte kommunikations-protokoller beskrives her:</p> <p>De "lave" protokoller i protokol-hierakiet omtales her (kort). F.eks. Bluetooth, WiFi, kablet Ethernet osv. (anvend henvisninger)</p> <p>De "mellemsste/øvre" protokoller i protokol-hierakiet omtales her (kort). F.eks. TCP/IP, UDP/IP osv. (anvend henvisninger)</p> <p>De øverste lag i protokol-hierakiet, dvs. applikationslags-protokollerne beskrives her. Disse er ofte proprietære, dvs. de er "skræddersyet" til projektet og skal derfor beskrives fuldt ud, da de skal anvendes til kommunikation mellem de enkelte CPU'er i systemet.</p> <p>F.eks. Kommando "Light Off": Byte 0 = 0xC7 Byte 1 = 0x00</p> <p>Kommando "Light On": Byte 0 = 0xC7 Byte 1 = 0x01</p>	<p>Beskrivelse af layout af data i systemet, f.eks. databaser (ER-diagrammer), tabeller, arrays etc.</p> <p>F. eks: User Name: 30 bytes Password: 14 bytes</p>	<p>Beskrivelse af systemets sikkerhed.</p> <p>F.eks. sikkerhed i forbindelse med login.</p> <p>F.eks. sikkerhed i forbindelse med datakommunikation (kryptering etc.)</p>	<p>Beskrivelse af aspekter, som er "tæt på" source code'n, herunder design-overvejelser:</p> <p>Algoritmer i de komplicerede metoder/funktioner, som blev beskrevet i "Logical View". Anvend f. eks. aktivitetsdiagrammer (flowcharts), pseudo code, state machine diagrammer.</p> <p>Evt. anvendt matematik, formler, beregninger som er involveret i source code'n.</p> <p>Ved microcotroller programmering kan setup af registre beskrives her i nødvendigt omfang.</p> <p>Anvendte compilere, libraries, frameworks inkl. versionsnumre.</p> <p>Projektets filer og den mappestruktur der er anbragt i.</p> <p>Udsnit af source code, som fortjener en særlig forklaring.</p>	<p>S O U R C E C O D E</p>

Test-dokumentation

Modultest: beskrivelse af teststrategi og testresultat for hvert enkelt HW/SW-modul, så vidt disse kan testes isoleret. Ellers beskrives teststrategi og testresultat for små grupper af HW/SW-moduler, der muliggør at funktionaliteten kan testes (dvs. en delvis integrations-test).

Integrationstest: beskrivelse af teststrategi og testresultat for trinvis integration af alle systemets HW/SW-moduler.

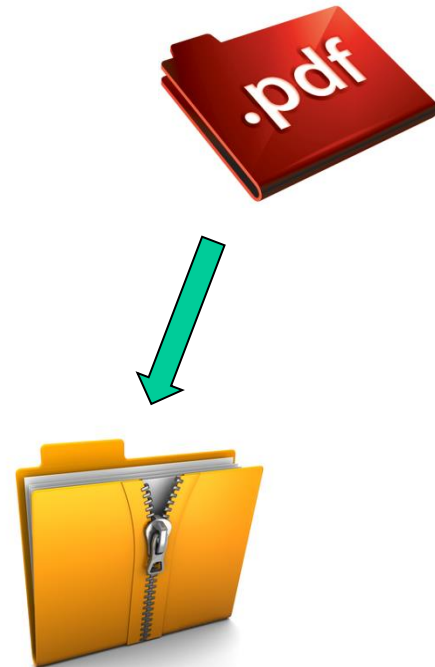
Accepttest: Beskriver på tabelform resultaterne af accepttesten, hvor denne test blev planlagt sideløbende med at kravspecifikationen blev skrevet. Når testen er udført kan OK'erne / ikke OK'erne indsættes i tabellen.

Vejledning til udfærdigelse af procesbeskrivelse



Procesbeskrivelsens disposition

Forside
Grubedannelse
Samarbejdsaftale
Udviklingsforløb
Projektledelse
Arbejdsfordeling
Planlægning
Projektadministration
Møder
Konflikthåndtering
Referenceliste



Vejledning til udfærdigelse af projektrapporter

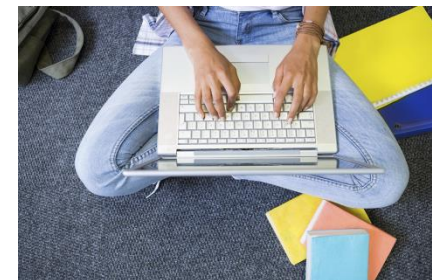


Motivation for et standard layout:

- At opnå en hensigtsmæssig struktur i projektrapporten.
- At sikre at den gennemførte proces kan beskrives uden at fokus mistes på den tekniske del af projektet.
- At tilpasse formatet af det afleverede materiale til ”elektronisk aflevering”



Til trods for anvendelsen af et standard layout er det de studerende, der skriver hvert eneste ord i projektrapporten.



Det afleverede 7BAC-projektmateriale består af

En projektrapport (PDF)

Heri beskrives selve projektet, set ud fra et teknisk synspunkt.

Kort sagt: hvad er opgaven/problemet, hvordan er det løst og hvilke resultater er der opnået.



Bilag til projektrapporten (ZIP)

Heri dokumenteres i nødvendigt omfang projektets tekniske del og yderligere den tilhørende procesmæssige del.



Det afleverede materiales overordnede struktur

Projektrapport, max. 30 normalsider (1 side = 2400 tegn, mellemrum tælles med)
Yderligere: figurer



Bilag til projektrapport, samlet i en ZIP-fil:

Teknisk del:

PDF-dokumenter som beskriver: kravspecifikation, analyse, arkitektur, design, implementering, modultest, integrationstest og accepttest.

Desuden: datablade, printlayout, source code, doxygen-genereret dokumentation etc.

Procesmæssig del:

Et PDF-dokument, der beskriver processen, her uddybes den korte omtale af processen som blev beskrevet i projektrapporten (6-10 normalsider)

Desuden: samarbejdsaftale, tidsplaner (hhv. planlagt og reelt udført), mødeindkaldelser, mødereferater, logbog, Scrum-dokumenter etc.



Projektrapportens disposition

Forside
Resumé / Abstract
Indholdsfortegnelse
Forord
Indledning inkl. problemformulering
Krav
Afgrænsning
Metode og proces
Analyse
Arkitektur
Design
Implementering
Test
Resultater
Diskussion af resultater
Konklusion
Fremtidigt arbejde
Referenceliste



Aflevering af bilag

Bilagene til projektrapporten skal afleveres i én ZIP-fil.

Bilagene kan f.eks. anbringes i disse foldere:

