AARHUS UNIVERSITET
INGENIØRHØJSKOLEN

Vejledning til udviklingsprocessen for projekt 2

Versionshistorik

Ver.	Dato	Initialer	Beskrivelse
0.01	17.11.14	KBE	Første version
0.02	24.11.14	TFJ	Rettet efter 1. review
0.03	26.11.14	KBE	Omskrevet analyse og design afsnit med inputs fra møde d. 19/11. Tilføjet beskrivelse af HW konstruktion, SW design og integrations-test.
0.04	27.11.14	KBE	Rettelser med inputs fra TFJ.
0.05	04.12.14	KBE	Rettelser med inputs fra FOH og TG. Fastlagt ordvalg for udviklings-processen.
0.06	05.12.14	KBE	Rettelser på baggrund af møde d. 5/12 (TG, KBE).
0.07	12.12.14	KBE	Rettelser tilføjet med inputs fra TG og GEK.
1.00	20.02.2015	KBE	Rettelser og første officielle version, fjernet skabeloner.

Indholdsfortegnelse

Indledning

Denne vejledning har til formål at beskrive den udviklingsproces, som følges på 2. semesterprojektet.

Projektet er fælles for E-, EP- og IKT-studerende på diplomingeniøruddannelsens 2. semester på Aarhus Universitet. Temaet er "Home Automation", hvor der skal udvikles både elektronik, PC og indlejret software, som er beskrevet i et separat projektoplæg [1].

Det er vigtigt at et projekt organiseres med en klar rollefordeling, for hvem der gør hvad, og at der etableres et godt samarbejde i projektgruppen. Der henvises til vejledningen om arbejdet i projektgruppen [2].

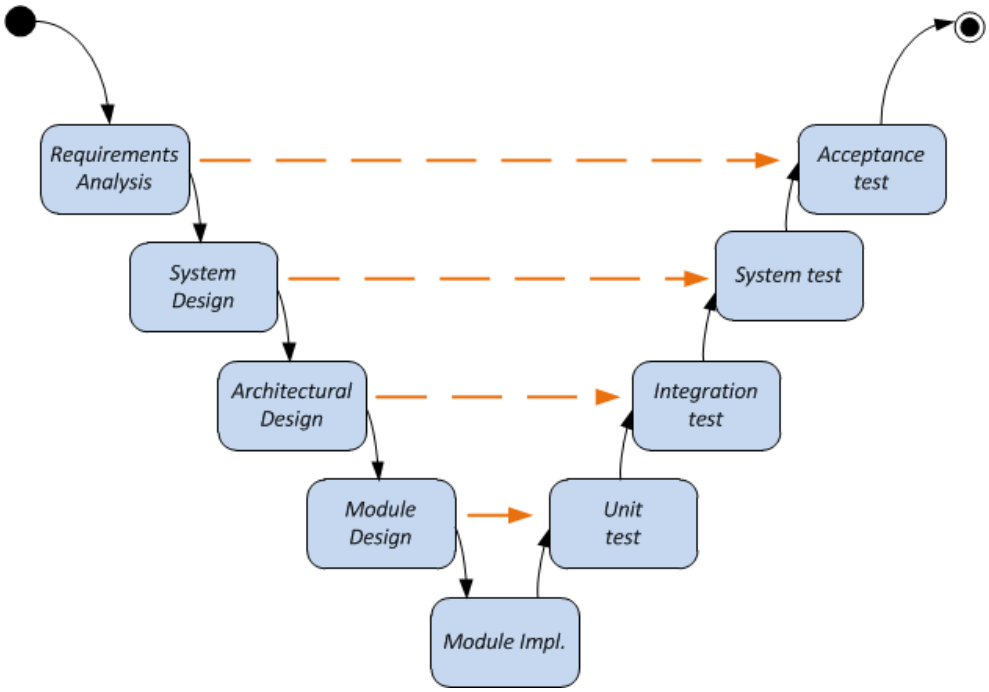
Semesterprojektet har fokus på at arbejde efter en struktureret udviklingsproces, hvor en del af emnerne fra 2. semester kurset "Introduktion til System Engineering" (herefter I2ISE) indgår i de enkelte faser af processen. Vejledningen har primært fokus på selve udviklingsprocessen og hvilken projektdokumentation, der skal udarbejdes i de enkelte udviklingsfaser. Projektdokumentationen udarbejdes i faserne og omhandler dokumenter som typisk også produceres i virksomhederne, med særlig fokus på specifikation, konstruktion og test af produkter. Faserne, som skal følges, omfatter projektformulering, specifikation, arkitektur, design, implementering og test.

Som afslutning på projektet, skal der skrives en projektrapport. Her beskrives selve projektet og processen, hvordan opgaven er løst og hvordan projektet er gennemført. Her skal de enkelte valg, opnåede erfaringer samt styrker/svagheder beskrives og fremhæves. Det er projektrapporten, som skal læses af både censor og vejleder og den danner grundlaget for bedømmelsen af jeres arbejde. For yderligere information henvises til vejledning for dokumentation af semesterprojekter [3].

I det følgende vil baggrunden for udviklingsprocessen blive beskrevet, hvorefter selve 2. semester udviklingsprocessen samt de enkelte faser vil blive gennemgået. Der henvises til kompendium, slides og øvelser i faget I2ISE for eksempler på brug af metoder og diagrammer som anvendes i de forskellige udviklingsfaser, herunder specifikation med Use Cases, systemarkitektur med SysML og softwaredesign med UML.

Baggrund

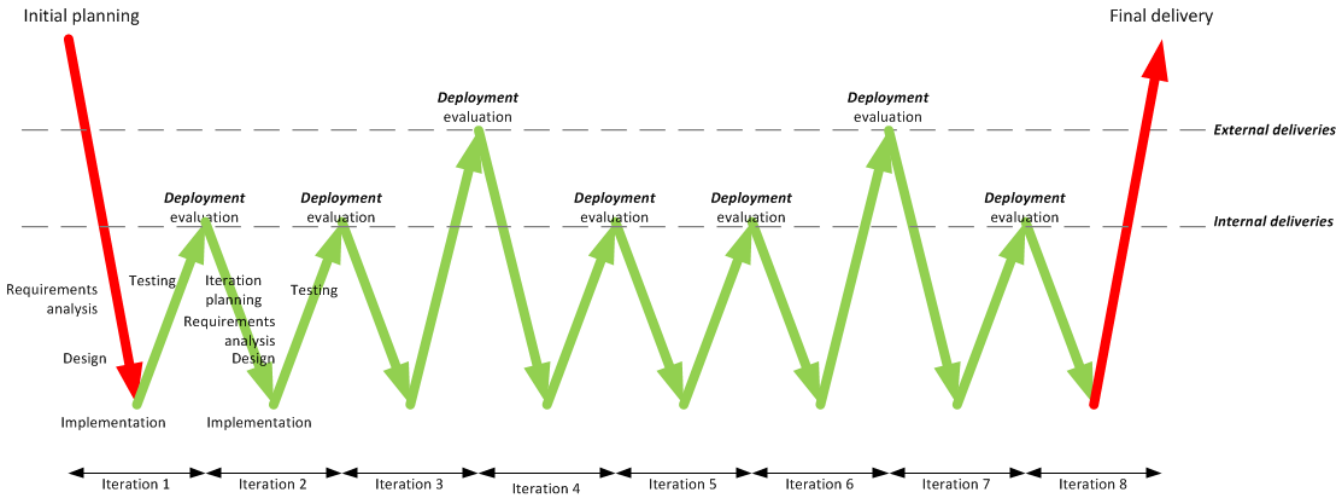
Udviklingsprocessen beskrevet i dette dokument er inspireret af vandfaldsmodellen og V-modellen som vist i figur 1. V-modellen omhandler udviklingsfaser og dertil relaterede testfaser, der indeholder test af resultatet for tilhørende specifikation, arkitektur eller design fase. Når man f.eks. er færdig med at specificere, hvad systemet skal gøre planlægges også, hvordan det færdige system skal testes når det afleveres til kunden (accepttest). Det er ikke alle faser i figur 1, der indgår i udviklingsprocessen som dette dokument omhandler.



Figur 1 V-modellens udviklingsfaser

Vandfaldsmodellen er bedst egnet til mindre, statiske projekter, hvor krav til projektet kan fastlægges og disse ikke ændrer sig i løbet af projektets gennemførelse. Det er netop tilfældet for 2. semesterprojektet, hvor gruppen selv fastlægger kravene til projektet. Vandfaldsmodellen er *ikke* egnet, hvis det er uklart hvad kunden gerne vil have udviklet og kravene dermed hyppigt ændres undervejs, eller hvis projektet har middel eller høj kompleksitet. Vandfaldsmodellen er heller ikke egnet, hvis teknologien er ukendt eller er umoden. I disse tilfælde er det en god idé at gennemføre et for-projekt eller bruge iterative metoder, der er bedre til håndtering af usikkerhed om krav og ændringer.

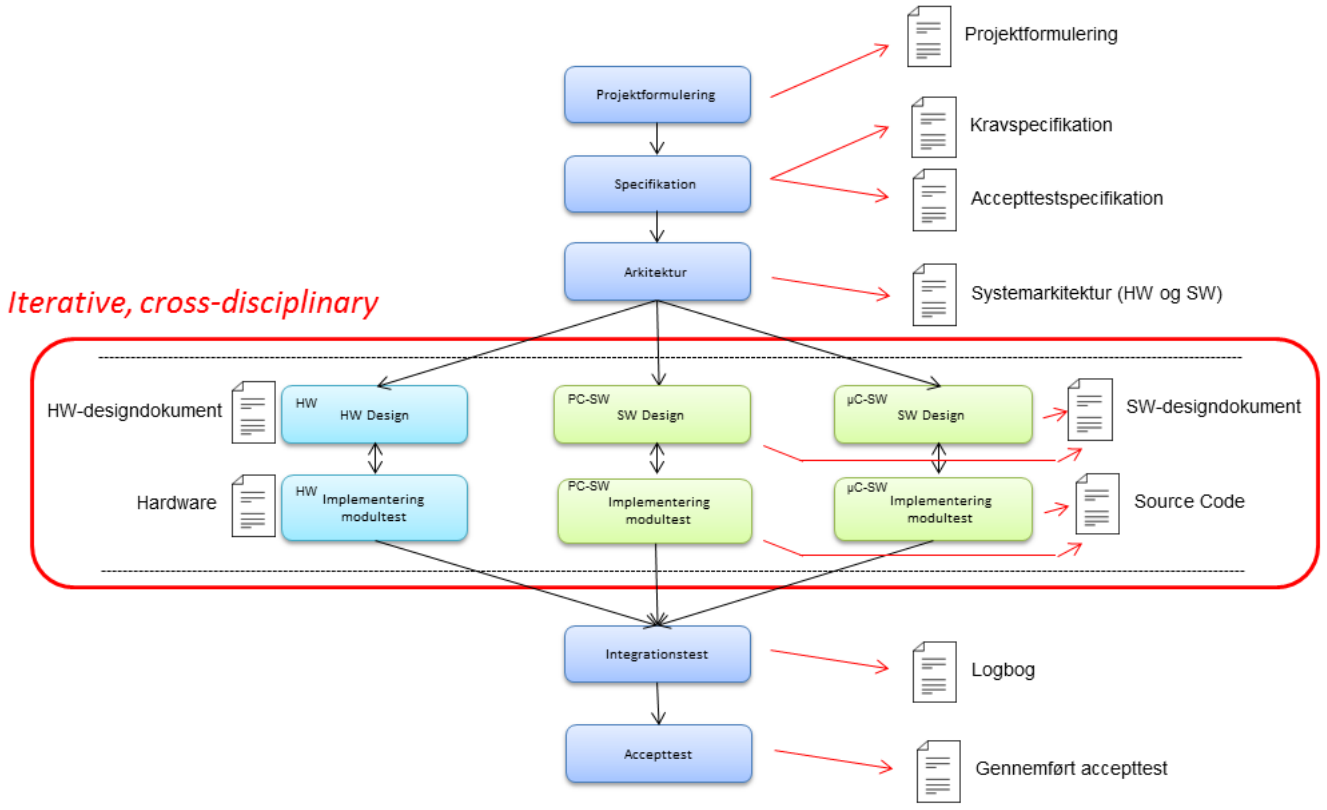
Figur 2 viser hvordan en udviklingsproces kan gøres mere agil med udgangspunkt i vandfalds- og V-modellen. Her planlægges flere iterationer, hvor kun en del af systemet udvikles i hver iteration. I en iteration gennemløbes faserne i V-modellen omfattende specifikation, arkitektur, design, implementering og test, i nogle tilfælde med levering til kunden. Systemet udvikles iterativt, hvor der ved hver delleverance tilføjes mere og mere funktionalitet til systemet.



Figur 2 Plan for en udviklingsproces med iterationer og delleverancer

Semesterprojektmodellen

Semesterprojektmodellen for 2. semester består af en række udviklingsfaser som vist i figur 3. Hver fase har et klart mål og afsluttes med en leverance i form af et eller flere dokumenter, færdige komponenter og/eller et testet system. De første faser gennemføres i fællesskab med det formål at få defineret, specificeret og opdelt systemet i blokke, så de efterfølgende faser med detaljeret design, implementering og modultest kan gennemføres parallelt. De 3 parallelle faser bør gennemføres som små iterationer, hvor der for hver iteration besluttes, hvad der skal leveres. De færdige dele af systemet samles og testes i flere integrationstests, hvor systemet gradvist tilføjes mere funktionalitet og afprøves internt af projektgruppen. Det er på 2. semester valgfrit om modultest og integrationstest dokumenteres. Enhedstest omfatter test af enkelte hardwareblokke eller softwareklasser, hvor andre blokke simuleres efter behov. Integrationstest planlægges og udføres i trin, hvor flere og flere dele af systemet sættes sammen og testes.



Figur 3 Semesterprojekt modellen illustreret med de faser som projektet gennemløber

Som afslutning på udviklingen af et system gennemføres en accepttest, hvor der udarbejdes en rapport med kommentarer til fejlende tests eller tests, som ikke kan gennemføres. Det forventes ikke ubetinget at alle Use Cases, som er specificeret og beskrevet i kravspecifikationen, er implementeret. Det skal dog klart fremgå af de enkelte dokumenter, hvilke krav og Use Cases, som systemet er designet og implementeret til, at skulle opfylde.

Der henvises til Scrum [6] som inspiration til et styringsværktøj for udførelsen af aktiviteterne for HW design, SW design, implementering og test. Her ville det være relevant at inddele hver iteration i et eller flere sprints og holde de møder, som er defineret for Scrum.

Der er i projektet defineret en række deadlines, hvor der skal være gennemført review af dokumenterne. Disse reviews er listet nedenfor:

1. Deadline omfatter review af Projektformulering, Tidsplan, Kravspecifikation og Acceptestspecification
2. Deadline omfatter review af Systemarkitektur og evt. en første version af HW/SW-designdokumenter
3. Deadline omfatter gennemførelse af accepttest med vejleder

Fælles for alle produktdokumenter er, at de skal indeholde følgende:

- Forside med titel, version, dato og gruppenummer, samt navne på projektgruppens deltagere
- Versionshistorik med resumé af ændringer med dato, version og initialer
- Indholdsfortegnelse
- Indledning med kort beskrivelse af dokumentets formål
- Referencer til relaterede dokumenter

I de følgende kapitler vil indholdet af de forskellige faser og dokumenter bliver beskrevet. Der henvises til kompendium, slides og øvelser i faget "Introduktion til System Engineering" for eksempler på, hvordan specifikationen med Use Cases, systemarkitektur med SysML og software med UML kan beskrives og formuleres.

Projektformulering

Det første skridt i projektarbejdet er at få beskrevet, hvad projektet går ud på. Gruppen diskuterer med udgangspunkt i projektoplægget, visionen og målet for projektet. Projektets overordnede fokus og funktionalitet beskrives på 1-2 sider. Hvad er problemet som systemet skal kunne løse i relation til omverdenen, og hvordan bidrager systemet til løsningen af problemet? En overordnet illustration kan være værdifuld i beskrivelsen af jeres idé. Projektet skal afgrænses, så det står klart hvad der er med og hvad er ikke med. Det skal ligeledes fremgå hvilken funktionalitet, der er højst prioriteret og hvilke dele af løsningen, der vil kunne vente til en senere opgradering af systemet.

Projektformuleringen skal godkendes af gruppens vejleder og udarbejdes sammen med en første tidsplan med milestones. En milestone er et veldefineret tidspunkt i projektforløbet, hvor en eller flere dokumenter er godkendt og opdateret, eller hvor et produkt kan demonstrere en vis grad af færdiggørelse.

Specifikation

I denne fase specificeres systemet i detaljer. Formålet er at få detaljeret kravene til hvad systemet kan, både i funktionalitet og kvalitet.

De funktionelle krav beskrives i form af use cases (UCs). Kvalitetskrav/ikke-funktionelle krav skal være målbare og formuleret så de kan testes. Som inspiration til kvalitetskrav kan kategorierne fra FURPS [5] bruges. Kravene kan prioriteres med formulering ved brug af MoSCoW [4] prioritering. På dansk bruges udsagnsordene *skal* (Must), *bør* (Should,) *kan* (Could) eller *vil ikke* (Won't).

En **kravspecifikation** bør indeholde nedenstående emner:

- Systembeskrivelse
- Funktionelle krav formuleret ved UCs
    - Beskrivelse af systemets aktører

    - UC diagram
    - Fully dressed UC-beskrivelser
- Kvalitetskrav/ikke-funktionelle krav
    - Samtlige testbare kvalitetskrav til systemet
- Andre krav:
    - Krav til anvendt udviklingsproces
    - Tekniske krav
    - Krav til grænseflader (ekstern HW/SW og kommunikation med andet udstyr)
    - Krav til anvendte udviklingsværktøjer
- Skitse/prototype af brugergrænseflade
    - F. eks. skærmbilleder (User Interface)

For at validere kravspecifikationen, udarbejdes en accepttestspecifikation. Denne formuleres i forlængelse af kravspecifikationen og review'es sammen med denne. Formålet er at sikre at alle krav fra kravspecifikationen kan testes når systemet er færdigudviklet. Findes der ikke testbare krav bør kravspecifikationen opdateres.

Accepttestspecifikationen er desuden et aftalegrundlag med kunden for, hvilke test der skal kunne gennemføres for at systemet kan accepteres af kunden. Accepttestspecifikationen bør formuleres så enkelt og klart som muligt, så kunden (eller dennes repræsentant) selv kan gennemføre testen og kontrollere resultatet.

En **accepttestspecifikation** bør indeholde nedenstående:

- Beskrivelse af testsystemet
    - Skitse af testopstilling og udstyr, der er påkrævet for at gennemføre testen
    - Testkomponenter (antal, evt. ID, versionsnummer)
- Testscenarier for hvert forløb i hver UC, med definition af konkrete test data. Testscenarierne bygges op i teststeps. For hvert test step specificeres følgende:
    - Brugerens handling
    - Systemets forventede respons herpå
    - Systemets faktiske respons herpå
    - Godkendelse af teststep'et (eller reference til fejlrapport)

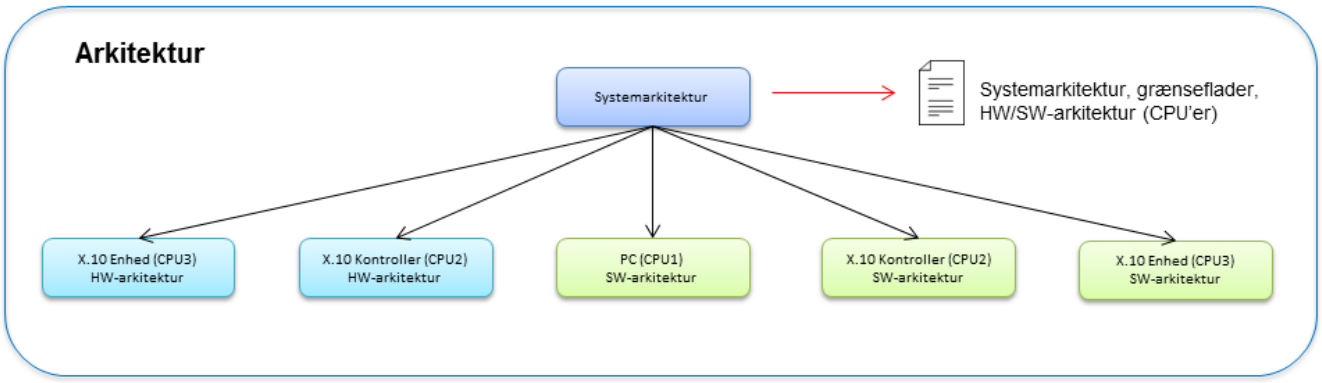
Accepttestens scenarier er enkelte, veldefinerede scenarier for de enkelte UCs. Scenariernes teststeps skal formuleres så de følger steps i den UC, der testes. Fordi en funden fejl skal kunne genskabes/reproduceres.

Fasen afsluttes med et review af krav- og acceptestspecifikationerne, som opdateres på baggrund af input fra reviewet.

Arkitektur

Arkitekturfasen har til formål at tilvejebringe en systemarkitektur på baggrund af de krav, der er specificeret for systemet. I denne fase nedbrydes systemet i blokke og grænsefladerne mellem disse blokke specificeres. Målet er at skabe et arbejdsgrundlag for den efterfølgende SW design-, HW design- og implementeringsfase. På baggrund af systemarkitekturen skal det være muligt at arbejde parallelt i flere teams med de enkelte blokke i systemet. Det skal således være klart, hvad en given blok skal gøre, og hvordan der kommunikeres mellem blokkene. Som minimum skal hardwareblokkens forbindelser (input og output) og signaler være specificeret.

For softwaren udarbejdes en domænemodel for hele systemet og applikationsmodel for hvert delsystem (hver CPU) vha. UML. Applikationsmodellen beskriver vha. klasse- og sekvensdiagrammer, hvordan grænseflade-, domæne- og kontrolklasser arbejder sammen for at opfylde kravene i en given use case. Som minimum skal klasserne i applikationsmodellen udarbejdes med et indledende forslag til metoder og attributter. Det er på dette tidspunkt ikke nødvendigt med en præcis, detaljeret definition af alle parametre og typer for hver metode og attribut.



Figur 4 Arkitekturen som den skal beskrives for 2. semesterprojektet med systemarkitektur, del-systemerne (CPU'erne) PC, X.10 kontroller og X.10 enhed.

Arkitektur-fasen består således af følgende aktiviteter illustreret på figur 4:

1. *Fastlæggelse af systemarkitekturen* med opdeling af systemet i delsystemer.
   Systemarkitekturen indeholder en SysML-beskrivelse, der fastlægger hvilke HW-blokke (delsystemer med hver sin "CPU") systemet skal bestå af, udarbejdet vha. SysML strukturdiagrammer (BDD'er og IBD'er). På et separat SysML BDD vises desuden allokeringen af software applikationer til delsystemerne (CPU'erne). Dette kan gøres vha. stereotypen `<<allocates>>` på associationer fra software- til hardwareblokke.

2. *Fastlæggelse af grænsefladen mellem de enkelte delsystemer*:
   Kommunikationsprotokollen mellem systemets enheder defineres og beskrives, herunder – for 2. semesterprojektet – den serielle kommunikation mellem PC og X.10 kontroller (STK500-kit) samt selve X.10 protokollen.

3. *Udarbejdelse af HW-arkitektur*:
   Hvert delsystem (”CPU”) nedbrydes til mindre HW- blokke illustreret vha. SysML struktur diagrammer (BDD’er og IBD’er) indtil tilstrækkelig overskuelighed opnås. Der udarbejdes desuden en tabelbeskrivelse af HW-interfaces.

4. *Udarbejdelse af SW-arkitektur*:
   Hvert delsystem (”CPU”) nedbrydes til mindre SW-bestanddele (klasser) illustreret vha. UML klassediagrammer indtil tilstrækkelig overskuelighed opnås. Der udarbejdes klasse- og sekvens-diagrammer baseret på relevante use cases til udarbejdelse af en applikationsmodel.

Beskrivelsen af systemarkitekturen kan således indeholde nedenstående:

- Systemarkitektur
  - Overordnet beskrivelse af delsystemer i relation til omgivelserne med SysML strukturdiagrammer (BDD’er og IBD’er)
  - Beskrivelse af scenarier jf. use cases med sekvensdiagrammer for kommunikation mellem delsystemerne, og mellem delsystemer og aktører

- Grænseflader
  - Kommunikationsprotokoller, der bruges på snitfladerne mellem delsystemer og eksterne enheder (aktører)

- HW-arkitektur
  - Beskrivelse af HW-arkitekturen med SysML strukturdiagrammer (BDD’er og IBD’er) samt evt. sekvensdiagrammer, alle suppleret med tekst eller tabeller, som beskriver
    - Blokkenes formål og funktion
    - Grænseflade for forbindelser mellem blokke
    - Specifikation af elektriske signaler og protokoller

- SW-arkitektur
  - En domænemodel for hele systemet
  - En applikationsmodel (UML) for hvert delsystem (”CPU”), bestående af sekvensdiagrammer og statiske UML klassediagrammer
  - Indledende forslag til attributter og metoder i klassediagrammerne for SW-interfaces

Fasen afsluttes med et review af systemarkitekturen, som efterfølgende opdateres på baggrund af input fra reviewmødet.

HW Design, Implementering og modultest

Hver blok (print eller kredsløb), som er identificeret og specificeret i systemarkitekturen for hardwaren, designes. Blokkene beskrives med kredsløbsdiagrammer, hvor eksterne forbindelser relateres til porte i SysML diagrammerne. Teori, formler og beregninger beskrives og udarbejdes i forbindelse med konstruktion af blokken. Designet skal indeholde komponentberegninger, udarbejdelse af diagrammer (schematics), styklister og evt. PCB-layout.

Simulerings og testresultater præsenteres med grafer og resultater i tabeller. Testopstillinger for verifikation af de enkelte moduler beskrives med præsentation af måleresultater.

SW Design, Implementering og modultest

SW designet detaljeres med opdaterede klassediagrammer med attributter og metoder og parametertyper. Hver klasse beskrives i detaljer, hvor public metoder beskrives med navn, returtyper, parameter og funktionalitet. Dokumentation af klasserne skal være i overensstemmelse med den faktiske implementerede kode. Hvis applikationsmodellen indeholder mange klasser opdeles designet i logiske pakker. Beskrivelsen kan suppleres med sekvens-diagrammer, hvor det vil være relevant. Bruges som eksempel til beskrivelse af et kompliceret kommunikationsforløb mellem klasserne. State-diagrammer kan medtages til beskrivelse af tilstande for en given klasse, hvis denne er implementeret som en state-maskine. Komplicerede metoder beskrives med pseudocode eller forklarende tekst.

Det beskrives hvordan de enkelte klasser af designet er testet og hvordan resultatet er verificeret. Det kunne være små testprogrammer og brug af modultest. Som minimum noteres test, resultater og fejlrettelser i projektets logbog.

Integrationstest

Delsystemerne sættes sammen og testes. Der udarbejdes en plan for, hvordan systemet sættes sammen og testes i flere små iterationer med brug af top-down eller buttom-up tests (drivers og stubbe). Eksempelvis kan PC kommunikation med X.10 controlleren testes uafhængigt om forbindelsen mellem X.10 controller og X.10 enheden er færdig. Der simuleres med LED'er på X.10 controlleren kommunikation med X.10 enheden. Denne fase dokumenteres ikke nødvendigvis, men væsentlige resultater medtages i den endelige rapport, hvor integrationstesten kort dokumenteres.

Som minimum noteres test, resultater og fejlrettelser i projektets logbog. Der tilføjes ligeledes løbende fejlrapporter indeholdende reference til det fejlede test step, alvorsgraden heraf (f. eks. mindre fejl, større fejl, kritiske fejl), hvordan fejlen rettes, og fejlens konsekvens for testen (testen afbrydes, testen skal gentages senere eller lignende).

Accepttest

I den afsluttende fase forberedes accepttesten. Der udarbejdes en testopstilling, og der holdes generalprøve på testen inden den gennemføres med vejlederen. Evt. manglende dele af systemet gøres bekendt for vejlederen inden testens start.

Hvis *ikke* alle dele af systemet er færdiggjort, er det gyldigt at simulere de ting som mangler. Det kunne f.eks. være at tænde en lysdiode i stedet for en rigtig lampe, eller simulere softwaredele af systemet, som mangler at blive færdiggjort.

Mens accepttesten gennemføres, udfyldes accepttestspecifikationen med faktiske observationer og resultat.

Referencer

[1] Projektoplæg 2. semester.pdf
[2] Arbejdet i projektgruppen 2. semester.pdf
[3] Vejledning til dokumentation af semesterprojekter.pdf
[4] Prioritering af krav med MoSCoW, http://en.wikipedia.org/wiki/MoSCoW_method
[5] Beskrivelse af krav med FURPS+, http://en.wikipedia.org/wiki/FURPS
[6] Scrum – agile development framework, http://en.wikipedia.org/wiki/Scrum_(software_development)