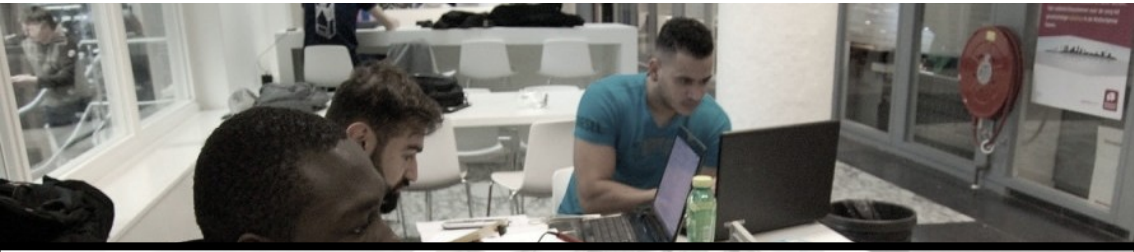
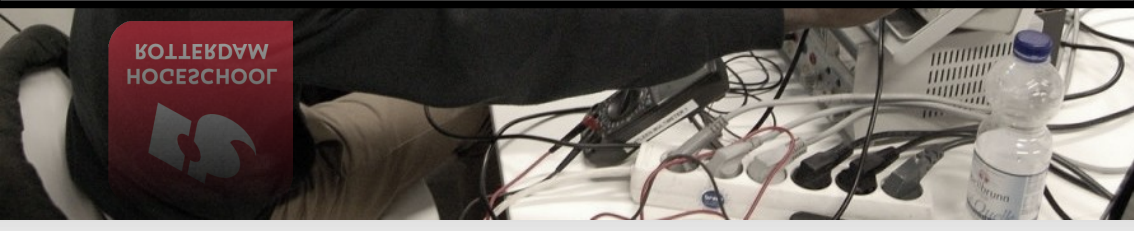


HOGESCHOOL ROTTERDAM

PROJECT

Projecthandboek HR-ELE



Versie 1.0

E. van de Logt
J.W. Peltenburg
M. Dubbeld
J.Z.M. Broeders
D. Versluis

Versiehistorie

Datum	Versie	Omschrijving	Auteur
18-10-2017	1.0[1]	• Engelse termen zoveel mogelijk omgezet naar het Nederlands. • Documenten samengevoegd tot één projectdocument.	VersD
09-12-2016	0.9	• Terminologie en figuren aangepast. • Delen die ook in Grit staan vervangen door verwijzingen. • Teksten over het beoordelen van projecten verwijderd.	BroJZ
21-11-2016	0.8	• LaTeX-versie gemaakt.	BroJZ
26-11-2014	0.7	• Verschillende bijlages verwijderd om handleiding onafhankelijker van de beoordeling te maken.	PelJW
25-11-2011	0.6	• Conversie naar nieuwe template. • Inhoudelijke aanpassingen m.b.t. focustabellen in bijlage A en B. • Bijlage C toegevoegd m.b.t. I/O van fasen V-model. • Bijzondere beoordelingsgevallen toegevoegd en beoordelingsformulier template in bijlage D. • Kleine correcties in taal en formulering na review E. van de Logt. • Inspectiebeheer toegevoegd.	PelJW DubbM

Wordt vervolgd op de volgende pagina.

Datum	Versie	Omschrijving	Auteur
13-12-2010	0.5	• Bij V-model architectuur toegepast. • Tekst van engineering proces up to date gemaakt. • Update m.b.t. nieuwe curriculum. • Introductie processen aangepast. • Conversie naar word 2007. • Tekst project management bijgewerkt.	PelJW DubbM
20-05-2010	0.4	• Projectcodes toegevoegd bij tabel 1 en tabel 2. • Update van wanneer welk deelproces ingevoerd wordt. • Beoordeling projecten (H4) vereenvoudigd.	LogEH
13-05-2005	0.3	• Hoofdstuk 4 'Beoordeling' toegevoegd. • Diverse kleine correcties n.a.v. discussie met docenten.	LogEH
13-05-2005	0.2	• Update n.a.v. commentaar docenten. • Verdeling nu per semester i.p.v. per jaar. • Engelse termen waar mogelijk omgezet in Nederlands. • Schattingen en Risico Management ingevuld. • Engineering fasen (behalve Inspecties) ingevuld.	LogEH
18-03-2005	0.1	• Eerste versie opgezet n.a.v. discussie met v.d. Berg, de Blois, van Loon en Peters.	LogEH

[1] Toelichting versiecodering *A.Bc*: A = grote aanpassing, B = kleine aanpassing, c = taal- of wiskundige correcties.

Inhoudsopgave

1 Inleiding

Binnen het opleidingsprogramma van Elektrotechniek, dat zich regulier over vier jaren uitstrekt, wordt een significant deel besteed aan het in groepsverband uitvoeren van projecten. Dit projecthandboek is opgezet om deze projecten zoveel mogelijk te laten aansluiten op echte projecten uit het bedrijfsleven. De aanpak en methodiek, zoals hier beschreven, komt overeen met een veelgebruikte aanpak uit het bedrijfsleven en vind je terug in bedrijven die op een professionele manier met projectmanagement omgaan (enkele voorbeelden zijn hier Philips, Freescale, Thales, Croon, ENGIE, Siemens, Spie, Mapper, Ericsson e.d.). Veel van de hier beschreven methoden, worden ook beschreven in *A Guide to the Project Management Body of Knowledge: PMBOK Guide* [2].

De projectenlijn begint al in het eerste jaar en loopt door tot in het vierde jaar. Om te voorkomen dat de student zich teveel ineens eigen moet maken is gekozen voor een gefaseerde invoering van de diverse methoden en technieken. De vierdejaars student werkt uiteindelijk met het gehele scala aan methoden en technieken.

Met deze opzet wordt het volgende geprobeerd te bereiken:

- Voor de student wordt duidelijk wat projectmatig werken allemaal omvat.
- Er is voor die student een duidelijk leerpad aangegeven.
- De opzet van projecten wordt voor iedereen (ook voor docenten) duidelijk.
- De studenten krijgen meer controle over het project dat ze uitvoeren, wat uiteindelijk tot minder stress aan het einde van het project kan leiden.

De indeling van dit projecthandboek is gestructureerd aan de hand van de diverse processen, waarmee je te maken krijgt in een projectorganisatie. Een veelvoorkomende indeling is de volgende:

- **Projectmanagementproces**: alle processen die te maken hebben met het managen van het project zoals: projectplanning, voortgangsbewaking, schattingen, risicomanagement e.d. Zie hoofdstuk 2.
- **Engineeringproces**: alle processen die te maken hebben met het daadwerkelijk uitvoeren van het project. Het engineeringproces wordt in dit

projecthandboek opgedeeld in verschillende fasen volgens het V-model. Zie hoofdstuk 3.

- **Ondersteunende processen**: alle processen die het projectmanagement- en/of engineeringsproces ondersteunen. Denk hierbij aan zaken als kwaliteitsborging (uitvoeren van inspectiesessies en audits e.d.) en configuratiemanagement (versiebeheer, e.d.). Zie hoofdstuk 4.

Dit projecthandboek dient gezien te worden als een aanvulling op het boek *Projectmanagement* van Grit [1]. Het is geen vervanging, regelmatig wordt in dit projecthandboek dan ook verwezen naar specifieke uitleg in het boek van Grit. Een student dient dus zowel dit projecthandboek als het boek van Grit te gebruiken. Van alle documenten die in dit projecthandboek worden besproken zijn templates beschikbaar. Deze templates zijn te vinden op de studentenschijf van de Hogeschool Rotterdam onder opleiding Elektrotechniek.

2 Projectmanagementproces

Projectmanagement is een continu proces dat primair uit twee deelprocessen bestaat:

- **Projectplanning vooraf**: dit zijn alle activiteiten die uiteindelijk resulteren in de oplevering van een plan van aanpak, waar alle betrokken partijen zich in kunnen vinden en zich aan verbinden. Omdat het maken van een planning (met behulp van een tool zoals GanttProject of Microsoft Project) apart aandacht vergt, is er een extra paragraaf aan gewijd, zie paragraaf 2.3 en ook [1, hoofdstuk 4].
- **Voortgangsbewaking**: Het controleren of het project nog steeds overeenkomt met het plan en de planning. Dit wordt in de literatuur ook 'project tracking' of 'project monitoring & control' genoemd. De docent heeft wekelijks een vergadering met het voltallige projectteam. De voortgang van het project is daarbij een belangrijk agendapunt.

Daarnaast zijn op dit gebied een aantal andere deelprocessen te vinden die gebruikt worden bij projectplanning en bij voortgangsbewaking. Deze zijn:

- **Schatten**: het maken van schattingen. Het maken van een goede schatting is bijzonder lastig, vooral als je werk moet schatten waar je nog geen beeld van hebt. Toch moet je, ook in dat geval, een schatting afleveren. Een eenvoudige methode, die goede resultaten oplevert is de zogenaamde 'Wideband-Delphi'-methode, die dan ook in paragraaf 2.4 zal worden beschreven.
- **Risico Management**: het belangrijkste van projectmanagement is in feite het continu omgaan met nieuwe risico's die opduiken. Dit is dus geen eenmalig gebeuren (aan het begin van het project), maar een activiteit die bij iedere voortgang centraal staat! Zie ook [1, paragraaf 3.12]. Om dit goed te kunnen doen is een methode nodig die een risico kan kwantificeren, zodat de prioriteit van risico's kan worden aangeven. In paragraaf 2.5 wordt zo'n methode beschreven.

2.1 Plan van Aanpak

Het maken van een plan van aanpak wordt uitgebreid behandeld in [1, hoofdstuk 6]. Op de studentenschijf is een template voor een plan van aanpak beschikbaar.

2.2 Voortgangsbewaking

Na goedkeuring van het plan van aanpak worden de diverse werkzaamheden, zoals beschreven in dat plan, uitgevoerd. Het is nu van belang om regelmatig te kijken in hoeverre de werkelijke activiteiten overeenkomen met de geplande activiteiten. Als de afwijking hiertussen te groot dreigt te worden, dan moet er ingegrepen en bijgestuurd worden. Dit kan op diverse manieren:

- Bijstellen van de projectplanning.
- Bijstellen van de activiteiten.

Er worden wekelijks voortgangsvergaderingen gepland met het hele team, waarvoor de begeleidende docent ook wordt uitgenodigd. Deze wekelijkse voortgangsvergadering heeft een vaste agenda, zie [1, hoofdstuk 7] voor details. Op de studentenschijf is een template voor een agenda beschikbaar.

Een aantal onderwerpen moet elke vergadering opnieuw besproken worden, zoals:

- **Planning**. De planning wordt voor aanvang van iedere voortgangsvergadering door de projectleider geactualiseerd. Welke taken zijn afgerond en kunnen in de planning op compleet gezet worden? Welke taken lopen uit? Wat zijn daarvan de consequenties? Moeten er extra taken bijkomen e.d.?
- **Risico's**. Zijn er nieuwe risico's bijgekomen? Zijn risico's groter / kleiner geworden? Welke risicoreducerende maatregelen moeten genomen worden? Zie hiervoor ook paragraaf 2.5
- **Actielijst**. Welke acties staan nog op de actielijst en welke komen erbij?

Van deze wekelijkse voortgangsvergaderingen worden notulen gemaakt.

2.3 Maken van een planning (Gantt Chart)

Het maken van een eerste versie van een planning wordt gedaan tijdens het opstellen van het plan van aanpak. Maar nadien wordt deze planning **wekelijks bijgewerkt** aan de hand van de opmerkingen van het team tijdens de voortgangsvergadering. In [1, paragraaf 4.3] wordt het maken van een zogenoemde strokenplanning besproken. In programma's voor projectmanagement zoals GanttProject[2], Gantter[3] en Microsoft Project[4] wordt dit een Gantt Chart genoemd.

De planning is dus **een levend document**, dat op vrijwel ieder moment de werkelijke status van het project aangeeft. Om dit te kunnen realiseren is een redelijk gedetailleerde planning nodig. Binnen een programma voor het maken van een planning is het vaak mogelijk om hoofdactiviteiten te definiëren, waaronder meer gedetailleerde taken geplaatst kunnen worden.

Als de taken in de planning zijn geplaatst is het belangrijk aan iedere taak een aantal uren toe te kennen. Vervolgens worden er een of meerdere resources (= personen

[2] GanttProject is een gratis, open source, programma dat beschikbaar is voor Windows, Mac OS en Linux. Zie http://www.ganttproject.biz/.

[3] Gantter werkt in de cloud, er is dus altijd een up-to-date versie van de planning online voor alle projectleden beschikbaar. Zie http://www.gantter.com/.

[4] Microsoft Project is een professioneel planningsprogramma en beschikbaar op de schoolcomputers. Zie http://www.microsoft.com/project/.

die het werk gaan uitvoeren) aan gekoppeld en kan de planning vervolgens laten zien of zo'n resource niet overbelast raakt. Als er bijvoorbeeld maar 11 uur in een week beschikbaar zijn en een resource is in een bepaalde week voor 20 uur ingepland, dan zal de planning bijgesteld moeten worden. De uren die aan een taak toebedeeld worden zijn afkomstig uit een Wideband-Delphi schattingssessie waar het hele projectteam aan bijgedragen heeft (zie paragraaf 2.4).

2.4 Schattingen m.b.v. de Wideband-Delphi methode

Een gedetailleerde beschrijving van deze Wideband-Delphi methode is te vinden in het artikel "Stop Promising Miracles" [3].

Het doel van de Wideband-Delphi methode is om een zo nauwkeurig mogelijke urenschatting te leveren van het werk dat gedaan moet worden. De procedure hiervoor is de volgende:

1. Maak een Work Breakdown Structure (WBS). Een WBS is een zo gedetailleerd mogelijke opsomming van alle werkzaamheden die gedaan moeten worden. Een handig hulpmiddel hierbij kan een Microsoft Project sjablone voor een planning zijn. Het maken van zo'n WBS kan het beste door het hele team gedaan worden, omdat teamleden verschillende inzichten hebben en iedereen hiermee akkoord moet zijn. Probeer zo volledig mogelijk te zijn in de opsomming van alle werkzaamheden.
2. Ieder teamlid maakt nu, onafhankelijk van de andere teamleden, een schatting van het benodigde aantal uren voor iedere taak. Er vindt dus geen overleg plaats tussen de verschillende teamleden!
3. De teamleden overhandigen de eigen schatting aan de projectleider. De projectleider berekent voor iedere taak het gemiddelde aantal uren en de grootste afwijking (in procenten) voor iedere taak.
4. Alle taken die meer dan 50% afwijken worden besproken in de groep. Vaak blijkt dat de verschillende teamleden verschillende ideeën hebben over wat zo'n taak nu precies inhoudt. Nadat iedereen hetzelfde beeld van de taken

heeft gekregen, wordt een tweede schatting gedaan van alle taken die meer dan 50% afweken.

5. Dit wordt maximaal nog 1 keer herhaald tot een totaal van 3 keer.
6. De projectleider telt de gemiddelde uren van iedere taak bij elkaar op. Hierdoor ontstaat een schatting van het totaal aantal uren.
7. Deze schatting dient als basis voor de planning.

2.5 Risicomanagement

Het doel van risicomanagement is om continu de projectrisico's in kaart te brengen, te kwantificeren, te volgen en passende risicoreducerende maatregelen te nemen. Risicomanagement wordt gebruikt tijdens het opstellen van het plan van aanpak, maar ook tijdens de wekelijkse voortgangsvergaderingen.

Een goede formulering van een risico heeft de volgende structuur: Als gevolg van *oorzaak* vindt de volgende *gebeurtenis* plaats die de volgende *gevolgen* heeft voor de *klanteisen*.

De procedure voor risicomanagement is de volgende:

1. Discussieer met alle teamleden welke risico's zij zien. Probeer te komen tot een zo volledig mogelijke lijst.
2. Doe het volgende voor ieder risico:
    - Bediscussieer de kans van optreden van het risico:
        - Laag: 0% – 30%, kleine kans van optreden. Index = 1.
        - Gemiddeld: 30% – 70%. Index = 2.
        - Hoog: 70% – 100%: grote kans van optreden. Index = 3.
    - Bediscussieer de impact van het risico op het project:
        - Laag: vrijwel geen extra uren nodig als het risico optreedt. Index = 1.
        - Gemiddeld: duidelijk merkbaar als het risico optreedt, flinke afwijking t.o.v. het plan van aanpak. Index = 2.

    - Hoog: Impact erg groot: als risico optreedt, kan dit leiden tot het niet succesvol afronden van het project. Index = 3
  - Bereken de 'exposure'. De exposure van een risico is het product van de kans van optreden en de impact. Voorbeeld: Een gemiddelde kans van optreden (index = 2) en een hoge impact van het risico (index = 3) leidt tot een exposure van $2 \times 3 = 6$.
3. Zet alle risico's in de risicotabel (bijvoorbeeld in een Excel spreadsheet), inclusief de kans van optreden, de impact en de berekende exposure. Sorteer de risico lijst op exposure, zodat de risico's met de grootste exposure bovenin de lijst komen te staan.
4. Bedenk voor ieder risico een risicoreducerende maatregel. Voor een risicoreducerende maatregel zijn de volgende mogelijkheden aanwezig:
  - Accepteren: accepteer het risico. Hierbij gok je er op dat het risico niet optreedt of beslis je dat andere risicoreducerende maatregelen te duur zijn.
  - Reduceren: neem concrete maatregelen om de kans van optreden en/of de impact op het project te verminderen.
  - Overdragen: draag het risico over aan een andere partij. Dit overdragen lukt alleen als de andere partij het risico ook accepteert en overneemt. Gebruik dit als je weinig controle over het risico hebt en de andere partij dat wel heeft.
  - Voorkomen: neem zodanige maatregelen dat het risico niet meer kan voorkomen (de kans van optreden en/of de impact worden zeer klein).
  - Contingentie: plan activiteiten die in gang gezet worden als het risico optreedt. Of reserveer uren/geld die/dat gebruikt kunnen worden als het risico optreedt.
5. Werk de risicolijst wekelijks bij. De kans van optreden en/of de impact van een risico kunnen wekelijks veranderen. Ook kunnen er nieuwe risico's bijkomen.

3 Engineeringproces

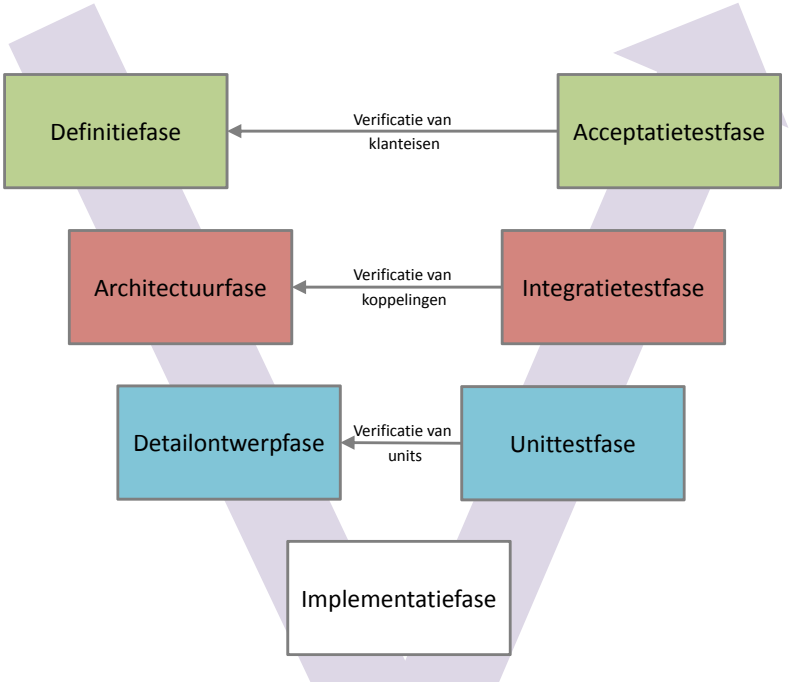
Het te gebruiken engineeringproces verschilt van industrietak tot industrietak. Voor projecten, die een relatief korte looptijd hebben, wordt vaak een eenvoudig model gebruikt.

Het in dit projecthandboek beschreven model is gebaseerd op het veel gebruikte V-model, dat afgeleid is van het watervalmodel en in de praktijk het meest gebruikt wordt in software-, maar ook wel in hardwareontwikkelomgevingen. Het V-model kent vele interpretaties en varianten. Het doel van dit projecthandboek is dan ook om de variant van de opleiding Elektrotechniek aan de Hogeschool Rotterdam weer te geven zoals die in de studentenprojecten gebruikt dient te worden. In het bedrijfsleven zullen varianten van het V-model vaak een stuk uitgebreider zijn.

Het V-model dat in dit projecthandboek behandeld wordt bestaat uit zeven fasen, zie figuur 1.

Bij aanvang van het project moet een beeld gevormd worden wat er (functioneel gezien) gemaakt gaat worden. Het hoe (de implementatie) is op dat moment nog niet zo van belang. In deze handleiding wordt er van uitgegaan dat het project een elektrotechnisch systeem, kortweg systeem genoemd, moet opleveren. Van dit systeem wordt in afstemming met de klant een lijst met eisen (de zogenoemde klanteisen) gespecificeerd. Dit gebeurt tijdens de **definitiefase**.

Bij de volgende fase, de **architectuurfase**, wordt het te maken systeem in eerste instantie bezien vanaf de buitenkant, als een zwarte doos (Engels: black box). In de architectuurfase is het belangrijk om deelsystemen en vooral ook de koppelingen tussen die deelsystemen te onderscheiden en te specificeren. Binnen het systeem heeft elk deelsysteem een eigen, goed afgebakende functie. Zaken die betrekking hebben op het gehele systeem zoals bijvoorbeeld het beveiligingsconcept moeten in het architectuurontwerp worden meegenomen. Ieder deelsysteem kun je waar nodig verder uitwerken in je architectuurontwerp totdat je op een niveau komt, waarbij het ontwerpen van dit deelsysteem eenvoudig is geworden en waarbij het



Figuur 1: Het V-model

weinig zin heeft het deelsysteem nog verder op te delen. Zo'n niet verder op te delen deelsysteem wordt een **unit** genoemd.

Bijvoorbeeld bij het maken van een audioversterker kunnen de volgende units worden onderscheiden: kast, voeding, voorversterker, toonregeling, volumeregeling, eindversterker en kortsluitbeveiliging.

Vervolgens kom je in de **detailontwerpfase** terecht. Hierin worden de units in detail ontworpen. Je tekent in deze fase elektrische schema's en maakt de bijbehorende berekeningen. Ook kunnen PCB layouts, One-line diagrams, software flowcharts of toestandsdiagrammen hier vervaardigd worden. Je maakt in deze fase belangrijke ontwerpkeuzes die goed onderbouwd moeten worden in het projectdocument. Het resultaat van deze fase is dus een gedetailleerd ontwerp voor

de implementatie van elke unit. In deze fase kunnen er ook simulaties van de units worden uitgevoerd om het ontwerp te verifiëren en eventueel te verbeteren.

Bijvoorbeeld bij het maken van een audio-versterker kan in deze fase een common-emitter schakeling worden getekend en berekend om de spanning te versterken, die functioneel gezien de rol van de voorversterker gaat vervullen.

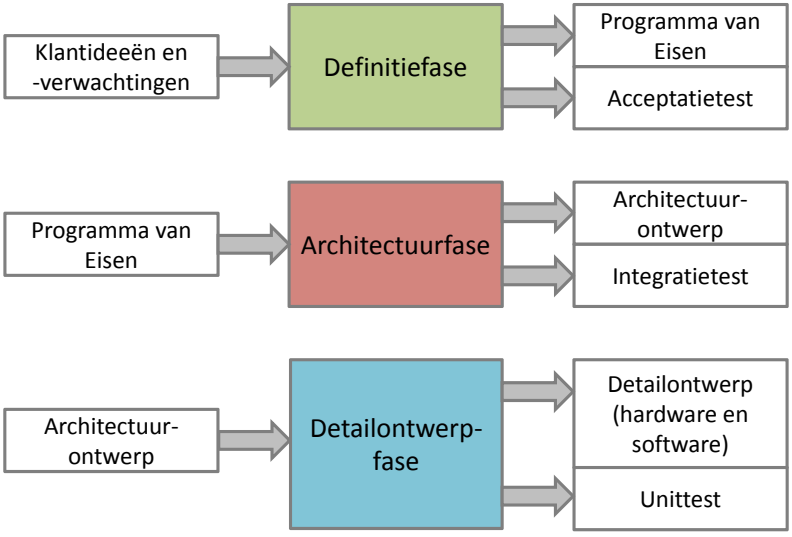
In de **implementatiefase** worden de units in elkaar gezet zodat deze straks getest kunnen worden. Ook het daadwerkelijk coderen van de software gebeurd in deze fase.

In de volgende fase, de **unittestfase**, wordt elke unit onderworpen aan een serie testen om te kijken in hoeverre de unit doet wat deze moet doen. Om terug te komen op het voorbeeld van de versterker zou dat de unit voorversterker kunnen zijn waarbij je gaat controleren of een bepaalde ingangsspanning binnen de marges van de gespecificeerde factor wordt versterkt. Omdat je in deze fase weet hoe de unit is geïmplementeerd kun je bij het uitvoeren van deze testen gebruik maken van deze kennis. Om deze reden worden deze manier van testen in veel literatuur 'glass box testing' genoemd. Een andere veelgebruikte naam voor 'glass box testing' is 'white-box testing' als tegenhanger van 'black-box testing'.

Als de unittestfase succesvol is afgesloten, kunnen de units met elkaar verbonden worden om te kijken in hoeverre ze met elkaar samenwerken. Dat gebeurt tijdens de **integratietestfase**. Bij deze integratietest kijk je niet meer naar de correcte werking van de units afzonderlijk (want dat heb je al in de unittestfase gedaan), maar je controleert nu vooral of de units goed met elkaar samenwerken. Dat betekent dat je test of de koppelingen tussen de units goed werken, zoals gedefinieerd in het architectuurontwerp.

Als alle units goed met elkaar samenwerken, wordt het tijd voor de **acceptatietestfase**. Hier test je de werking van het totale systeem, waarna het door de klant geaccepteerd kan worden. Dit doe je door systematisch alle klanteisen uit het programma van eisen te testen.

In het volgende gedeelte worden de fasen en de in- en outputs van elke fase nader beschreven. Dit is schematisch weergegeven in figuren 2 en 3. Bij veel fasen wordt als output een (deel van een) document opgeleverd.



Figuur 2: Inputs en outputs van de eerste drie fases van het V-Model.

3.1 Definitiefase

Input van deze fase:

- Ideeën en verwachtingen van de klant. Vaak zijn meerdere gesprekken met de klant nodig om de klanteisen (Engels: requirements) vast te kunnen stellen.

Output van deze fase:

- Programma van Eisen: een deel van het projectdocument dat een beschrijving en onderbouwing van de klanteisen bevat.
- Acceptatietestbeschrijving: een deel van het projectdocument dat alle acceptatietesten bevat die aan het einde van het project uitgevoerd zullen worden om te bewijzen dat aan de klanteisen voldaan is.

Tijdens de definitiefase wordt beschreven wat het systeem als geheel moet doen.

Er bestaan diverse methodieken om de klanteisen schematisch weer te geven. Een van deze methodieken is de methode van Yourdon [4] die gebaseerd is op functionele decompositie. Als je werkt volgens deze methode, dan teken je verschillende diagrammen waarin de functionaliteit van het systeem wordt vastgelegd. Onder andere een Context Diagram (CD), verschillende Data Flow Diagrams (DFD's) en State Diagrams (SD's). Het gedrag van de verschillende processen die door het systeem uitgevoerd worden, wordt vastgelegd in zogenoemde processpecificaties (PSPEC's). Daarnaast wordt ook een zogenaamd Data Dictionary (DD) aangemaakt waarin alle datastromen binnen het systeem beschreven worden.

Parallel aan de klanteisen kunnen de acceptatietesten opgesteld worden. De belangrijkste redenen om op dit moment de acceptatietesten te schrijven zijn:

- Verificatie van de klanteisen. Als de klanteisen testbaar zijn, moet het ook mogelijk zijn om hierbij een test te verzinnen.
- Afstemming met de klant (de opdrachtgever van het project).
- Als de klant akkoord gaat met de klanteisen en de acceptatietest, dan is voldaan aan een belangrijke voorwaarde om het project succesvol af te sluiten.

3.2 Architectuurfase

Input van deze fase:

- Programma van Eisen.

Output van deze fase:

- Architectuurontwerp: in de beschrijving van de architectuurfase in het projectdocument wordt het architectuurontwerp van het systeem vastgelegd.
- Integratietestbeschrijving: een deel van het projectdocument dat alle integratietesten bevat om tijdens de integratie te bewijzen dat alle koppelingen tussen de deelsystemen aan de eisen voldoen.

Tijdens de architectuurfase maak je een eerste opzet van het systeem en deel je dit op in deelsystemen. Vaak wordt begonnen om het systeem als een “black box” te zien. Vervolgens wordt dit systeem opgedeeld in kleinere blokjes om het overzichtelijker en gemakkelijker te maken. Zo kunnen bijvoorbeeld de taken in het project gemakkelijker verdeeld worden. Ook beschrijf je de koppelingen tussen de deelsystemen. Ieder deelsysteem en elke koppeling wordt dus apart beschreven. Elk deelsysteem heeft weer zijn eigen specificaties. Een deelsysteem kan weer verder opgedeeld worden in kleinere deelsystemen. Een deelsysteem dat in de architectuurfase niet verder wordt opgedeeld, wordt een unit genoemd.

Het overzicht van de diverse units en de koppelingen tussen die units wordt ook wel de architectuur van het systeem genoemd.

De moeilijkheid is altijd wanneer je moet stoppen met het verder detailleren van de specificaties van de units. Een vuistregel hierbij is dat je kunt stoppen wanneer het voldoende duidelijk is voor een software- of hardwareontwerper hoe het een en ander gemaakt moet worden. Probeer ontwerpbeslissingen zoveel mogelijk uit te stellen tot de detailontwerpfase zodat de ontwerper van de unit de maximale vrijheid heeft bij het ontwerpen. Wel moeten de koppelingen in detail afgesproken worden zodat los van elkaar ontworpen units wel goed samenwerken.

Het vastleggen van de architectuur is een zeer belangrijk doel van de ontwerpbeschrijving. Als deze architectuur goed in elkaar zit, kun je verschillende ontwerpers aan verschillende units laten werken en toch tot een goed totaalsysteem komen.

3.3 Detailontwerpfase

Input van deze fase:

- Architectuurontwerp.

Output van deze fase:

- Detailontwerp: in aparte delen van het projectdocument worden de gedetailleerde ontwerpen van de verschillende units vastgelegd.

- Circuit Diagram, PCB Layout, One-line diagram, etc.: alle tekeningen, schema's, berekeningen en dergelijke die nodig zijn om de hardware van het systeem te kunnen bouwen.
- State Diagram, Flowchart, etc.: alle tekeningen, schema's en dergelijke die nodig zijn om de software van het systeem te kunnen coderen.

Tijdens de detailontwerpfase worden de verschillende units die gedefinieerd zijn in de architectuurfase in detail ontworpen. Hier worden alle tekeningen en berekeningen gemaakt die nodig zijn om het systeem daadwerkelijk te kunnen implementeren.

Het kan voorkomen dat specificaties die zijn voortgekomen uit de architectuurfase of dat klanteisen die zijn voortgekomen uit de definitiefase niet gerealiseerd kunnen worden. Er moet dan een of meerdere stappen terug gedaan worden in het V-model (een nieuwe iteratie).

Aan de hand van de verschillende detailontwerpen kan vaak een BOM (Bill Of Materials) worden gemaakt waarmee er bij verschillende leveranciers alvast componenten kunnen worden besteld.

3.4 Implementatiefase

Input van deze fase:

- Circuit Diagram, PCB Layout, One-line diagrams, etc.
- State Diagram, Flowchart, etc.

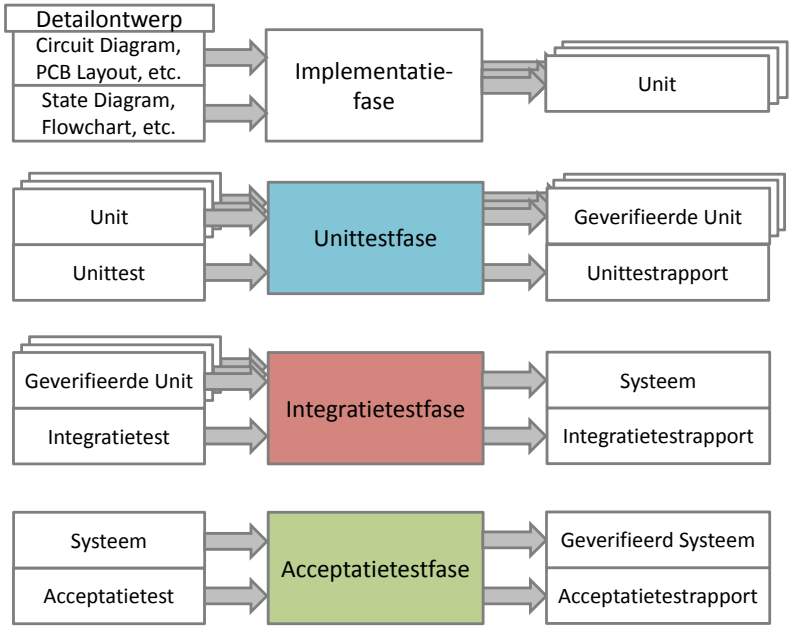
Output van deze fase:

- Units.

In deze fase worden bijvoorbeeld de printen gefreesd, de componenten op de print gesoldeerd, de software geschreven en de code in een microcontroller geladen.

3.5 Unittestfase

Input van deze fase:



Figuur 3: Inputs en outputs van de laatste vier fases van het V-Model.

- Units.
- Unittestbeschrijving.

Output van deze fase:

- Geverifieerde units.
- Unittestrapport: een deel van het projectdocument waarin de resultaten van de, in de unittestbeschrijving beschreven, testen worden gerapporteerd.

Voor ieder unit, zoals gedefinieerd in de ontwerpbeschrijving, wordt een unittest uitgevoerd. Deze unittesten zijn beschreven in de unittestbeschrijving. De functionaliteit van elke unit wordt in deze fase apart getest.

3.6 Integratietestfase

Input van deze fase:

- Geverifieerde units.
- Integratietestbeschrijving.

Output van deze fase:

- Systeem.
- Integratietestrapport: een deel van het projectdocument waarin de resultaten van de, in de integratietestbeschrijving beschreven, testen worden gerapporteerd.

Als een unit geverifieerd is, kan deze gekoppeld worden aan een andere geverifieerde unit. Het is dus niet nodig, en zelfs ongewenst, om alle units tegelijkertijd aan elkaar te koppelen! Het is het mooiste als je een integratietest gefaseerd kunt uitvoeren, waarbij iedere keer nieuwe units toegevoegd worden aan de reeds geïntegreerde deelsystemen. Ook kun je ervoor kiezen verschillende combinaties van units uit te proberen alvorens meerdere units te integreren. Het is verstandig om het moment waarop deze integratietesten uitgevoerd dienen te worden, vast te leggen in de planning.

Het belangrijkste doel van de integratietest is dus om te testen of de koppelingen tussen de units goed werken. De klanteisen, die in het programma van eisen zijn vastgelegd, worden in deze fase nog niet getest. De enige testen die hier uitgevoerd worden, zijn de integratietesten: werkt unit X goed samen met unit Y?

3.7 Acceptatietestfase

Input van deze fase:

- Systeem.
- Acceptatietestbeschrijving.

Output van deze fase:

- Geverifieerd systeem.
- Acceptatietestrapport: een deel van het projectdocument waarin de resultaten van de, in de acceptatietestbeschrijving beschreven, testen worden gerapporteerd.

Als de integratietest succesvol verlopen is, dan zijn alle deelsystemen aan elkaar gekoppeld en werken ze goed met elkaar samen. Het totale systeem is nu dus gereed om getest te worden. In hoeverre voldoet het aan de klanteisen?

Nu wordt de acceptatietestbeschrijving, die opgesteld werd in het begin van het project tijdens de definitiefase, zie paragraaf 3.1, weer van belang.

Het projectteam doorloopt eerst alle testen zelf en kijkt in welke mate aan de klanteisen voldaan wordt, door de acceptatietesten één voor één uit te voeren. In de praktijk wordt dit vaak de Factory Acceptance Test (FAT) genoemd. Meestal is de klant hier nog niet bij aanwezig.

Bij deze testen worden de details van de diverse deelsystemen en de koppelingen tussen deze deelsystemen niet meer opnieuw getest. Deze testen zijn immers al tijdens respectievelijk de unittestfase en de integratietestfase uitgevoerd. De acceptatietest kijkt dus voornamelijk naar de buitenkant van het systeem, dit wordt om deze reden ook wel een 'black-box test' genoemd.

Bij de officiële acceptatietest zal meestal de klant uitgenodigd worden. Soms echter is de klant ook al tevreden met alleen het acceptatietestrapport. Als een acceptatietest op locatie bij de klant zelf wordt uitgevoerd, dan wordt dit vaak de Site Acceptance Test (SAT) genoemd. Zowel de FAT als de SAT gebeuren op basis van het programma van eisen en de acceptatietestbeschrijving. Als er op dit moment nog fouten ontdekt worden, dan zijn deze vaak vrij kostbaar. Want (kijkend naar het V-model) moeten er nogal wat stappen terug gedaan worden, wat de nodige tijd zal kosten.

4 Ondersteunende processen

Alle processen die het projectmanagement- en/of het engineeringproces ondersteunen worden ondersteunende processen genoemd. Denk hierbij aan zaken als kwaliteitsborging (uitvoeren van inspectiesessies en audits e.d.) en configuratiemanagement (versiebeheer e.d.). In dit hoofdstuk wordt een methode voor kwaliteitsborging, namelijk inspecties, besproken. Inspecties hebben als doel, om

in een vroegtijdig stadium fouten te vinden in de klanteisen, het architectuurontwerp, de gedetailleerde ontwerpen, de verschillende testbeschrijvingen, enz. Ook wordt in dit hoofdstuk een methode voor configuratiemanagement besproken.

4.1 Inspecties

Om de kwaliteit van het eindproduct (een elektrotechnisch systeem) te waarborgen is een controle op de verschillende deelproducten noodzakelijk. In documentatie wordt bijvoorbeeld gewerkt met versiebeheer en is het gebruikelijk om een review op elk stuk tekst te laten uitvoeren door een ander persoon dan de auteur.

Vanuit de theorie is kwaliteitswaarborging controle houden over het hele proces. Indien een proces beheerst wordt is de uitkomst van het proces logisch en worden toevalligheden als uitkomst uitgesloten. Voor een projectgroep is het dan ook van groot belang dat de documentatie zorg draagt voor een eenduidige uitkomst. Omdat verschillende auteurs aan (delen van) documenten zullen werken is het verstandig de eenduidige (logische) uitkomst te controleren. Deze terugkoppeling (feedbackmoment) of inspectie wordt in het algemeen gehouden voor de oplevering van de definitieve versie.

Een inspectie is een formeel moment waarbij een geselecteerde groep een (deel van een) document inspecteert. Voor aanvang van zo'n inspectiesessie (review) wordt afgesproken hoelang er wordt gekeken naar een pagina. Tijdens het eerste deel van deze sessie wordt er onderling niet overlegd en worden alle punten waar mogelijk een vraag of fout in zit gemarkeerd.

In het tweede deel van de inspectiesessie wordt een lijst van de gemarkeerde punten gemaakt en wordt er een major of minor kwalificatie toegekend. Uiteindelijk wordt deze lijst met de auteur besproken en vindt er indien noodzakelijk een wijziging op het document plaats.

Belangrijk bij het houden van dergelijke inspectiesessies zijn de regels voor het geven van feedback. Zo moet vermeden worden dat een oordeel over de auteur gegeven wordt waardoor deze persoon zich aangevallen kan voelen. Ook geven

inspecteurs soms onterecht de voorkeur aan hun eigen formulering terwijl de originele tekst dezelfde strekking heeft.

4.2 Versiebeheer

Elke versie van een document moet voorzien zijn van een versienummer. Het is aan te bevelen om in elk document een tabel op te nemen waarin de versiehistorie is weergegeven, zoals ook in dit document is gedaan, zie pagina 2. Voor het beheren van code wordt het gebruik van het versiebeheersysteem git[5] aangeraden.

[5] Zie: https://git-scm.com/

Referenties

[1] Roel Grit. *Projectmanagement*. 7de druk. Noordhoff Uitgevers, 2015. ISBN: 978-90-01-85021-0 (geciteerd op pp. 7, 8, 9).

[2] Project Management Institute. *A Guide to the Project Management Body of Knowledge: PMBOK Guide*. PMBOK® Guide Series. Project Management Institute, 2013. ISBN: 978-1-935589-67-9. URL: http://www.pmi.org/pmbok-guide-standards/foundational/pmbok (geciteerd op p. 6).

[3] Karl E. Wiegers. “Stop Promising Miracles”. In: *Software Development* (2000). URL: http://www.uml.org.cn/softwareprocess/pdf/delphi.pdf (geciteerd op p. 10).

[4] E. Yourdon. *Modern Structured Analysis*. Yourdon Press computing series. Yourdon Press, 1989. ISBN: 978-0-13-598624-0 (geciteerd op p. 17).