Processimulator voor Windturbinebesturingen
Volume I: Probleemanalyse

T.G. van Engelen
P. Schaak

5 november 2001

Abstract

Door opschaling en minder toegankelijke offshore locaties is vooraf testen van *besturingssystemen* van windturbines steeds belangrijker om te kunnen voldoen aan hoge betrouwbaarheidseisen. Met *processimulaties* kan beoordeeld worden of het *besturingssysteem* het falen van *componenten* en (deel)*systemen* naar behoren afhandelt en of extreme bedrijfstoestanden goed worden doorstaan. In dit rapport wordt het ontwikkeltraject opgesteld voor het voorziene real-time *processimulatie*-gereedschap WINDCONTEST.

Het ontwikkeltraject is tot stand gekomen door voor een vereenvoudigde voorbeeldturbine een *simulator*-ontwikkeling daadwerkelijk uit te voeren en deze ontwikkeling tenslotte te formaliseren. Om een doeltreffende *processimulator* te ontwikkelen met de benodigde flexibiliteit ten aanzien van verschillende *turbine-realisaties*, is gestructureerd ontwerpen volgens de zogenaamde Ward-Mellor methodiek geschikt gebleken en biedt de real-time ontwikkelomgeving met MATLAB/dSpace ruime faciliteiten.

Verdere projectuitvoering zal leiden tot een geïnventariseerde set van *turbine-realisaties*, van waaruit ontwikkelde *componentmodellen* (grafisch) kunnen worden geselecteerd en geconfigureerd. De *processimulator* wordt gecreëerd door automatische conversie naar *component-modulen* voor real-time implementatie in de hiervoor opgestelde hardware voorzieningen. Het *windturbine-besturingssysteem* wordt hieraan fysiek gekoppeld.

Verantwoording

Erkentelijkheid gaat uit naar Eric van der Hooft (ECN) en Ilya Kraan (SPE) voor hun constructieve bijdragen tot het leggen van het fundament en de eerste steen van de *processimulator* voor het testen van *windturbine-besturingen*.

Distributie

NOVEM	1-5
Ministerie van EZ	6
Koninklijke Bibliotheek	7
ECN Wind Energy	8-12
F.W. Saris	13
W. Schatborn	14
G.A. Woudstra	15
H. Verkroost	16
H.J.M. Beurskens	17
L.G.J. Janssen	18
L.W.M.M. Rademakers	19
J.T.G. Pierik	20
E.L van der Hooft	21-22
T.G. van Engelen	23-24
P. Schaak	25-26

INHOUD

SAMENVATTING

In deze voorstudie is de uitvoeringswijze voor de ontwikkeling van de *processimulator* voor *windturbinebesturingen*, WINDCONTEST[1], nader gedetailleerd.

De gedetailleerde uitvoeringswijze is afgeleid door een aantal activiteiten van een *simulator*specificatie daadwerkelijk uit te voeren. Deze activiteiten zijn vervolgens functioneel omschreven en uitgebreid met beschrijvingen van de overige voorziene activiteiten.

De binnen deze probleemanalyse uitgevoerde *simulator*-specificatie is gebaseerd op een gestructureerde modelleringswijze zoals beschreven door Ward en Mellor, die het aanbrengen en wegnemen van storings- en uitzonderingstoestanden van nature in zich heeft [3]. Binnen het MATLAB-computerplatform voor technisch-wetenschappelijke berekeningen [2] kunnen uit zulke specificaties direct programma-*modulen* afgeleid worden voor geautomatiseerde verwerking tot een real-time *processimulator*.

Het vertrekpunt voor de uitgevoerde specificatie en de daaruit afgeleide detaillering van de werkzaamheden werd gevormd door:

- het uitzetten van hoofdlijnen voor het beoogde algemeen toepasbare *processimulatie*gereedschap;
- specificatie van het *besturingssysteem* voor de vereenvoudigde voorbeeld turbine waarop de daadwerkelijke *simulator*ontwikkeling was gebaseerd.

[1] **Wind** Turbine **Con**trol Systems **T**est, **E**valuation and **S**imulation **T**ool

DEFINITIES

De hieronder gedefinieerde termen hebben binnen dit project een specifieke betekenis. Ter herkenning zullen deze consequent doorgevoerde termen *cursief* worden weergegeven.

Controlproces:
Transformatie van tijd-discrete grootheden die samenhangen met conditionele voorwaarden; volgens Ward-Mellor definitie [3].

Component:
Bestanddeel in de meest gedetailleerde opdeling van het met de *processimulator* na te bootsen proces.
voorbeelden: oliedruksensor, tandwielkast, generator, kruimotor.

Componentmodel:
Model van een *component*.

Componentmodule:
Module van een *component*.

Dataproces:
Transformatie van tijd-continue grootheden die samenhangen met differentiaalvergelijkingen en algebraïsche functies; volgens Ward-Mellor definitie [3].

Medium:
Simulatie-item dat deel uitmaakt van de omgeving van een windturbine.
vooralsnog: grond, wind, golven en electriciteitsnet.

Mediummodel:
Model van een *medium*.

Medium-module:
Module van een *medium*.

Model:
Specificatie in de vorm van een *transformatie-schema*, of in de vorm van *data-* en *controlprocessen*; volgens Ward-Mellor definitie.

Module:
Computercode waarmee het gedrag van een *component, systeem, simlatie-item* of een geheel 'proces' kan worden gesimuleerd.

Processimulator (simulator):
Hardware-matig (real-time) gereedschap dat de voor *windturbine-outlines* configureerbare *simulatie-items* software-matig nabootst; dit met voldoende detail voor verificatie van een *windturbine-besturingssysteem* onder alle voorziene bedrijfssituaties.

Processimulatie (simulatie):
Simulatie met de *processimulator*.

Simulatie-item, item:
Bestanddeel in de meest globale opdeling van het met de *processimulator* na te bootsen proces, dat is samengesteld uit functie gerelateerde medium *componenten* of *windturbinedelen*. Vooralsnog: windturbine, grond, wind, golven en electriciteitsnet.

Simulatiewijze:
Beschrijving van gedrag en samenstelling van *componenten* op het voor de *processimulator* vereiste detailleringsniveau.

Systeem:
Afgebakend samenstelsel van functie gerelateerde windturbine *componenten*. Locale *systeem*hiërarchie is hierbij mogelijk.
Voorbeelden: generatorkoelsysteem, krui-onttwistsysteem, (welke respectievelijk worden omvat door) generatorsysteem, kruisysteem.

Systeemmodel:
Model van een *systeem*.

Systeemmodule:
Module van een *systeem*.

Transformatie schema:
Voorstelling van data interacties tussen *data*- en *controlprocessen*; volgens Ward-Mellor definitie [3].

Toestandsdiagram:
Voorstelling van conditionele toestandsovergangen binnen *controlprocessen*; volgens Ward-Mellor definitie.

Windturbine-besturingsstrategie (turbine-besturingsstrategie, besturingsstrategie):
Functionele beschrijving van de wijze waarop een *windturbine-realisatie* onder de voorziene bedrijfssituaties wordt beheerst.

Windturbine-besturingssysteem (turbine-besturingssysteem, besturingssysteem, windturbine-besturing, turbine-besturing, besturing):
Realisatie van een *windturbine-besturingsstrategie*.

Windturbine-concept (turbine-concept):
Klasse-indeling van windturbines naar karakteristieke kenmerken.
Indeling in termen van: active pitch to vane, variabel toerental, up-wind, direct drive, offshore.

Windturbinedeel (turbinedeel):
Afgebakend samenstelsel van functie gerelateerde *systemen*. De *windturbinedelen* vormen de meest globale opdeling van het *simulatie-item* windturbine. Vooralsnog: rotor, aandrijftrein en ondersteuning.

Windturbine-outline (turbine-outline, outline):
Functionele beschrijving van een windturbine in een omgeving.
in termen als: actieve bladverstelling, temperatuurmeting, geforceerde generatorkoeling, remote offshore

Windturbine-outline-library (turbine-outline-library, outline-library, library):
Geordende verzameling van *component-modules* die voor één *windturbine-outline* beschikbaar zijn.

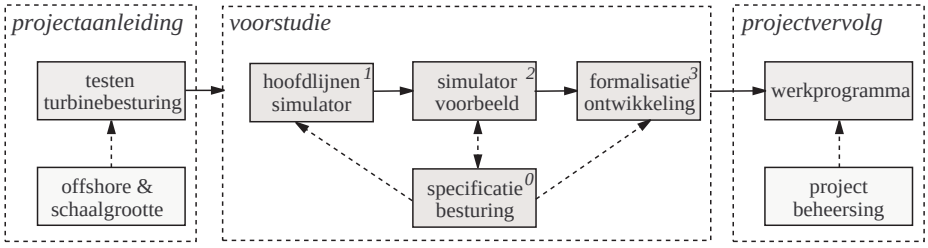
Windturbine-realisatie (turbine-realisatie, realisatie):
Beschrijving op *component*niveau van de wijze waarop (de functionaliteit van) een windturbine in een omgeving is gerealiseerd. in termen als: hydraulische bladverstelling met dubbel werkende cilinder, PT100-temperatuurmeting, generator-waterkoeling, windklasse IB, golfhoogte-klasse.

1. INLEIDING

Windturbinefabrikanten onderkennen steeds meer het belang van een goede *processimulator* voor het veilig en betrouwbaar testen van *turbinebesturingssystemen*. Met name in het geval van moeilijk bereikbare locaties zoals offshore. Cruciaal hierbij is dat een *simulator* juist de storings- en uitzonderingstoestanden in *componenten*, *systemen* en omgevingscondities moet kunnen omvatten, en met name ook de samenhang in de tijd tussen verschillende van deze toestanden.

Om deze reden is bij ECN het *processimulatie*-gereedschap WINDCONTEST[2] in ontwikkeling, waarmee een specifieke *turbine-outline* snel kan worden vertaald naar een operationele real-time *processimulator*, met inbegrip van mogelijke fout- en uitzonderingscondities. Het in het gesimuleerde proces kunnen opnemen van fysieke 'hardware in the loop' apparatuur, zal hierbij tot de mogelijkheden behoren.

Dit rapport beschrijft de voorstudie van het project, waarbinnen deze tool wordt ontwikkeld. Figuur 1.1 brengt de setting van deze voorstudie en de hierin onderscheiden uitvoeringsstappen in beeld.



Figuur 1.1 *Setting en stappen binnen probeemanalyse*

Als project-start is gekozen voor het opstellen van een vereenvoudigde voorbeeldturbine en -*besturing*, omdat daarbij veel facetten van de projectaanleiding aan de orde komen die van invloed zijn op het gehele ontwikkelproces voor het simulatiegereedschap. Deze 'stap 0' vormt echter geen onderdeel van de eigenlijke *simulator*ontwikkeling en is daarom opgenomen als appendix A. De kern van het rapport wordt gevormd door de beschrijving van stap 1 tot en met 3, wat als werkprogramma voor het projectvervolg zal dienen.

In hoofdstuk 2 worden de hoofdlijnen van de *simulator* uiteen gezet (stap 1). De context en functionele opbouw van de *processimulator* worden hier gedefinieerd. De specificatie van het *simulator*voorbeeld, stap 2, wordt beschreven in hoofdstuk 3. Om de projectvoortgang te bespoedigen wordt de specificatie van de voorbeeld*simulator* en het daarbij behorende *besturingssysteem* slechts gedeeltelijk uitgevoerd, uiteraard tot op een niveau dat toereikend is voor de detaillering van het werkprogramma. De derde stap, het formaliseren van de activiteiten die tijdens de *simulator*ontwikkeling worden doorlopen, wordt beschreven in hoofdstuk 4. Daarmee wordt de leidraad gecompleteerd, om de werkzaamheden gedurende het projectvervolg gestructureerd te kunnen uitvoeren.

[2] **Wind** Turbine **Con**trol Systems **T**est, **E**valuation and **S**imulation **T**ool

2. HOOFDLIJNEN PROCESSIMULATOR

In deze probleemanalyse wordt de uitvoeringswijze voor de ontwikkeling van de *processimulator* voor windturbinebesturingen nader beschreven. De *processimulator* is gedefinieerd als:

> Hardware-matig (real-time) gereedschap dat de voor *windturbine-outlines* configureerbare *simulatie-items* software-matig nabootst; dit met voldoende detail voor verificatie van een *windturbine-besturingssysteem* onder alle voorziene bedrijfssituaties.

Het primaire doel van dit gereedschap is *windturbine-besturingssystemen* aan Factory Acceptance Tests (FATs) kunnen onderwerpen: het *besturingssysteem* op eenvoudige wijze op alle mogelijke bedrijfssituaties kunnen toetsen, inclusief extreme - en potentieel destructieve situaties. Dit wordt mogelijk gemaakt door een opstelling te realiseren, waarbij de te toetsen *besturing* aan de *processimulator* kan worden gekoppeld, als ware de *processimulator* zijn werkelijke omgeving. De opzet staat toe delen van de windturbine als 'hardware in the loop' op te nemen (in plaats van software-matig na te bootsen).

De *simulator* zal per *windturbine-outline* worden geconfigureerd. Iedere *outline* kent een eigen *outline-library* met *componentmodulen*, waarmee de operator van de *processimulator* een *windturbine-realisatie* inclusief omgeving kan nabootsen, die binnen de beschikbare *outline* past. Met de *outline* ligt het 'werkingsprincipe' van de gesimuleerde turbine voor de operator op voorhand vast. Welke specifieke *componenten* hierbij van toepassing zijn bepaalt de operator.

Fysiek zal de *processimulator* bestaan uit 2 PC's:

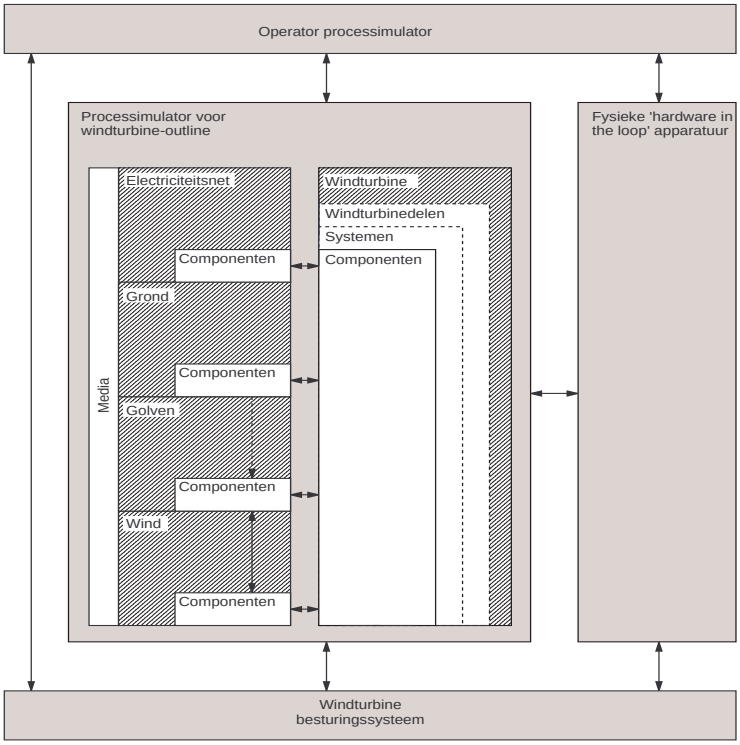
- een host-PC, waarop de *modulen* middels 'Graphical User Interfaces' (GUIs) binnen de MATLAB-omgeving worden geïmplementeerd [2];
- een target-PC, waarop de *modulen* als C-code worden 'gedownload', waarna de turbine inclusief omgeving real-time wordt nagebootst en met de te toetsen *besturing* en de fysieke 'hardware in the loop' apparatuur communiceert.

De volgende paragrafen gaan verder in op de hoofdkenmerken van het simulatiegereedschap. In §2.1 wordt de functionele opbouw van het proces*model* geschetst en komt de real-time data-uitwisseling tussen *simulator*, *besturingssysteem*, *operator* en fysieke 'hardware in the loop' apparatuur aan bod. Vervolgens wordt in §2.2 de functionele opbouw van het proces*model* verder vormgegeven door middel van transformatieprocessen, waarbij aandacht wordt besteed aan de interface-mechanismen binnen de *simulator* (ofwel aan de wisselwerking tussen de functionele bouwstenen die binnen het proces*model* worden onderscheiden). Hierbij wordt de methode om de detailspecificaties voor de *processimulator* af te leiden, in hoofdlijnen uiteengezet en toegelicht met een voorbeeld.

2.1 Functionele opbouw en externe data-uitwisseling

Figuur 2.1 geeft de voorziene context van de *processimulator* weer, met daarin opgenomen de opbouw van het proces*model*. Bij het testen van een *windturbinebesturingssysteem* communiceert de operator van de *processimulator* met de *simulator*, de daarop aangesloten *besturing* en de fysieke 'hardware in the loop' apparatuur: de operator legt de te simuleren situaties op en neemt het resulterende gedrag waar. Uiteraard wisselen de *simulator*, *besturing* en 'hardware in the loop'

apparatuur ook informatie met elkaar uit.



Figuur 2.1 *Context en functionele opbouw processimulator*

De meest globale opdeling van het proces*model* wordt gevormd door de *simulatie-items*, te weten de windturbine en de media (i.e. wind, golven, elektriciteitsnet en grond). Uiteindelijk worden alle *items* opgebouwd uit *componenten*, als zijnde de bestanddelen in de meest gedetailleerde opdeling van het proces. Hierbij bestaat voor de *media* geen tussenliggend opdelingsniveau, het *item* windturbine kent wel een gelaagde opdeling: een windturbine is opgebouwd uit *windturbinedelen*; de *windturbinedelen* zijn opgebouwd uit *systemen*; de *systemen* zijn opgebouwd uit *componenten*. Voor de *windturbinedelen* worden de standaard-delen rotor, aandrijftrein en ondersteuning voorzien. Bij de definitie van *systemen* is lokale hiërarchie toegestaan: een *systeem* kan kleinere *systemen* omvatten.

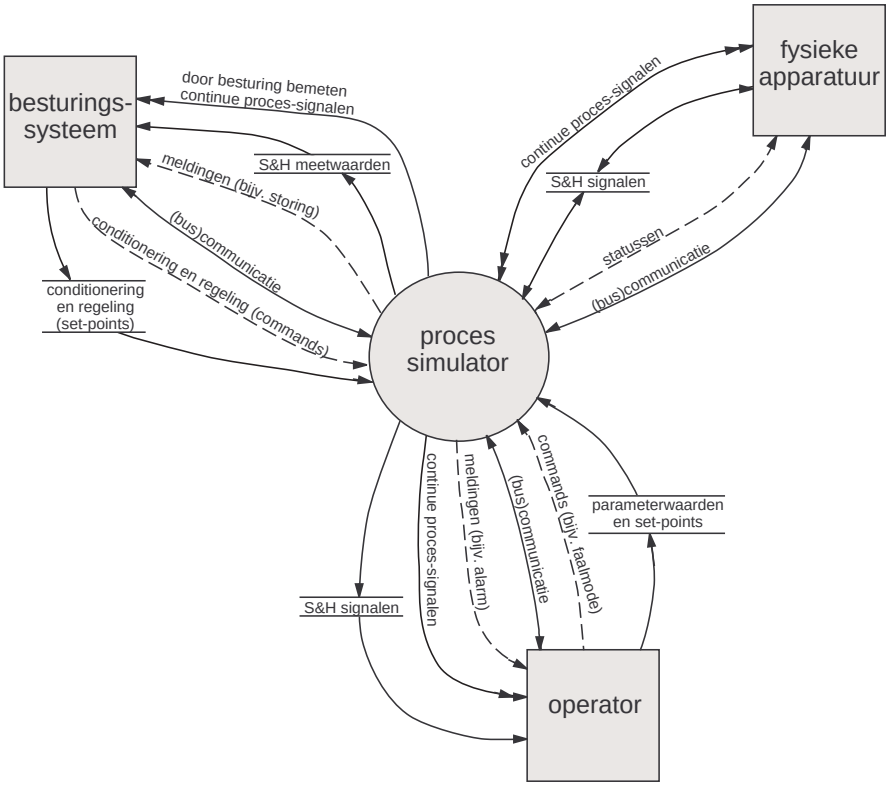
Figuur 2.2 geeft de wisselwerking weer tussen *simulator*, operator, *besturingssysteem* en 'hardware in the loop apparatuur', conform de binnen dit project gebruikte representatiewijze voor *modellen* [3].

In de figuur wordt de *processimulator* als transformatieproces weergegeven, door middel van een cirkel. De operator, *besturing* en fysieke apparatuur worden met vierkanten weergegeven, als zijnde de terminators van de *processimulator* (i.e. de omgevingsprocessen van de te ontwikkelen *simulator*, waarvoor de *modellen* worden opgezet).

De pijlen representeren data-flows die de wisselwerking tot stand zullen brengen. Er wordt onderscheidt gemaakt tussen:

- tijdcontinue signalen: doorgetrokken lijnen met dubbele pijlpunt;
- tijddiscrete signalen: doorgetrokken lijnen met enkele pijlpunt;
- events: onderbroken lijnen met enkele pijlpunt.

Een event verandert het transformatie-proces waarop het 'binnenkomt' (met betrekking tot modes; bijvoorbeeld de draairichting van een motor, het in - en uit



Figuur 2.2 *Transformatieschema van de simulator-context*

storingstoestanden gaan van een *systeem* of het stochastische gedrag van de wind).

Verder toont figuur 2.2 zogenaamde data-stores: twee horizontale lijnen, waartussen de parameter genoemd staat die door de store wordt opgeslagen. Een data-store representeert dat de waarde die op een discreet tijdstip wordt afgegeven, op de signaal-interface aanwezig blijft totdat er een nieuwe waarde wordt afgegeven.

Er is bewust voor gekozen de terminators niet rechtstreeks met elkaar te laten communiceren, maar dergelijke informatie-uitwisseling via de *processimulator* te laten verlopen. Dit biedt het voordeel dat alle signalen centraal (in de *simulator*) beschikbaar zijn en houdt de opzet van de *simulator* overzichtelijk.

Figuur 2.2 heeft betrekking op de situatie dat een *model* van een *windturbine-realisatie* inclusief omgeving reeds is samengesteld en dat hiermee een besturingssysteem middels *processimulatie* wordt getest. Het *besturingssysteem* opereert dan, zoals in werkelijkheid. De communicatie tussen het gesimuleerde proces en het *besturingssysteem* bestaat uit de volgende signalen:

- continue processignalen die door de *besturing* worden bemonsterd;
- bemonsterde meetwaarden (sample & hold meetopnemer als onderdeel van het gesimuleerde proces);
- meldingen vanuit het proces (status veranderingen);
- tijddiscrete communicatie-signalen in beide richtingen, bijvoorbeeld via een ‘bus’;
- commands van de *besturing* (regeling en conditionering van het proces);
- set-points van de *besturing* (regeling en conditionering van het proces).

Daar de *besturing* tijd-discrete apparatuur betreft, geeft deze geen tijd-continue signalen af.

Met het gebruik van fysieke 'hardware in the loop' apparatuur wordt een stukje van het proces niet gesimuleerd, maar daadwerkelijk aan het real-time gesimuleerde proces gekoppeld. De communicatie tussen de fysieke apparatuur en de *simulator* betreft dus communicatie binnen het proces. Derhalve kunnen hier in beide richtingen alle mogelijke signaalsoorten worden uitgewisseld. Hierbij zal sprake zijn van de volgende signalen:

- continue processignalen;
- bemonsterde (sample & hold) processignalen;
- statussignalen;
- tijddiscrete communicatie-signalen (bijvoorbeeld via een 'bus').

De operator heeft in de context van de figuur de taak de *simulatie* te beheersen (een voorgaande taak is het uit de *outline-library* samenstellen van het gesimuleerde proces, de figuur betreft echter de situatie dat dit reeds is voltooid). Alle mogelijke signaalsoorten die zich hierbij in het proces voordoen, kunnen worden doorgegeven aan de operator en andersom zou de operator middels alle signaalvormen gedrag kunnen opleggen aan het proces. Omdat het echter zeer onwaarschijnlijk is dat hierbij sprake zal zijn van het tijd-continu opleggen van gedrag (maar eerder het middels tijd-discrete (bus)communicatie doorgeven van een te volgen 'track'), is dit niet in de figuur opgenomen. De volgende signalen resteren voor de communicatie tussen de operator en *simulator*:

- parameterwaarden en set-points waarmee de operator het proces parametriseert en aanstuurt;
- operator-commands die proces(delen) tussen verschillende modes laten schakelen;
- tijddiscrete communicatie-signalen in beide richtingen (bijvoorbeeld via een 'bus');
- meldingen vanuit het proces (status veranderingen);
- continue processignalen;
- bemonsterde (sample & hold) signalen;

Ten tijde van een *processimulatie* zal er ook een (door de operator te bedienen) Mens-Machine-Interface (MMI) aan de *besturing* zijn gekoppeld, zoals deze ook in een werkelijke turbine gebruikt zal worden. Daar dit niet van belang is voor het opzetten van de *processimulator*, is dit niet in de figuur opgenomen.

Als het *model* van de *processimulator* in de volgende paragrafen wordt gedetailleerd, zal duidelijk worden dat hierbij ook sprake zal zijn van 'interne' events. Dit betekent dat het *processimulator-model* bestaat uit zowel tijdcontinue *dataprocessen* als uit tijddiscrete *controlprocessen*. Hiertussen wordt onderscheidt gemaakt door gebruik te maken van:

- *dataprocessen*: doorgetrokken cirkels;
- *controlprocessen*: onderbroken cirkels.

Bij het verder detailleren van een transformatieproces in 'lager gelegen' transformatieprocessen, kan een *dataproces* naast 'lager gelegen' *dataprocessen* ook *controlprocessen* blijken te omvatten. Andersom is niet mogelijk, een *controlproces* omvat geen *dataprocessen*.

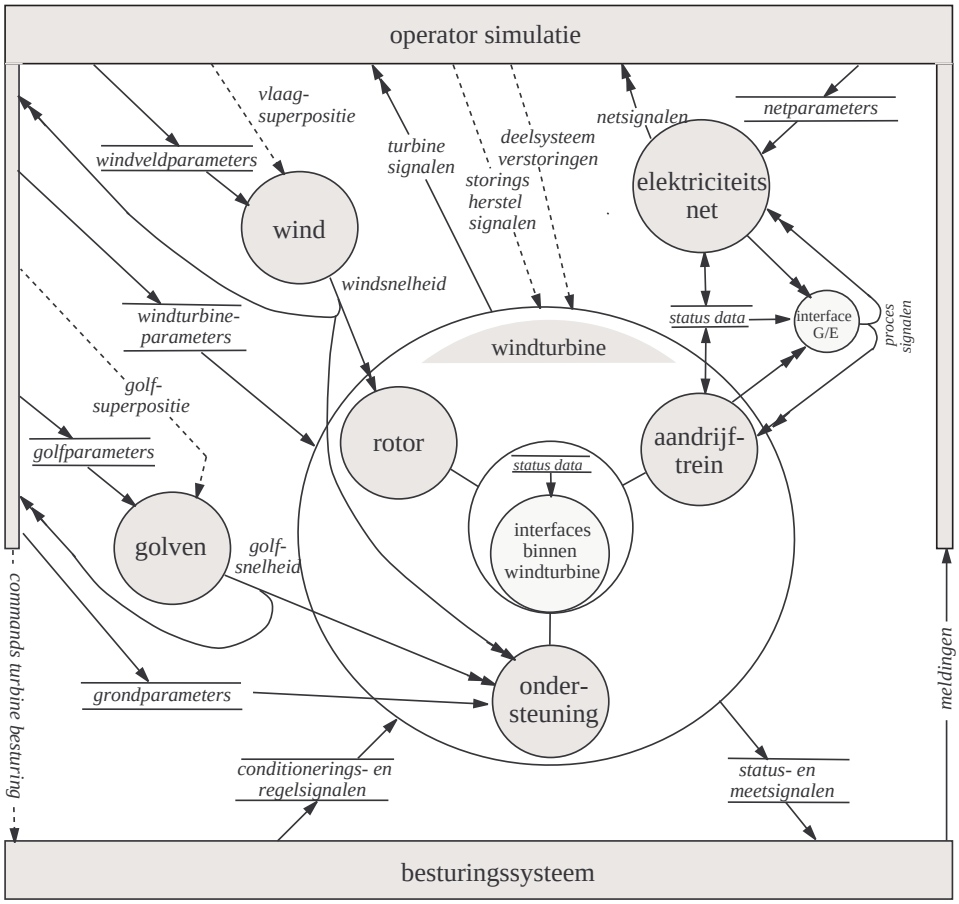
De tweedeling in *data-* en *controlprocessen* staat toe dat de modelbeschrijvingen

doorzichtig geïmplementeerd kunnen worden binnen het MATLAB-platform voor technisch-wetenschappelijke berekeningen en dynamische simulatie [2]. Binnen MATLAB zijn hiervoor zogenaamde 'Graphical User Interfaces' (GUIs) beschikbaar.

Het MATLAB-platform voorziet ook in de conversie naar de real-time simulatieomgeving. De computer met MATLAB-programmatuur, waar de modelbeschrijvingen mee zijn geïmplementeerd (i.e. de host-PC) faciliteert 'downloading' van de 'MATLAB-modellen' naar een procescomputer (i.e. de target-PC), waarop het proces dan middels real-time C-programmatuur kan worden gesimuleerd met real-time I/O naar het *besturingssysteem* en de 'hardware in the loop' apparatuur. Tijdens deze simulaties blijft de host-PC ook aangesloten voor de realisatie van overzichtelijke user I/O.

2.2 Functionele opbouw van het procesmodel

Nu wordt beschreven hoe transformatieprocessen en interfacemechanismen gestalte zullen gegeven aan de *modellen*, die voor de *processimulator* zullen worden opgesteld. Figuur 2.3 toont het top-level transformatieschema van de processimulator: het hoogst beschouwde niveau van opdeling voor het te simuleren proces, inclusief de omgeving waarmee wordt gecommuniceerd.



Figuur 2.3 *Top-level transformatieschema processimulator*

In de figuur is slechts een beperkte hoeveelheid interface-signalen (dataflows) opgenomen en de 'hardware in the loop' optie is buiten beschouwing gelaten. Tijdens de definitie-taak in het ontwikkeltraject van de *processimulator* zal de werkelijk

benodigde data-uitwisseling in beeld worden gebracht. (zie taak 2, ‘Definitie’ in hoofdstuk 4). De hier gepresenteerde opzet dient slechts ter illustratie.

In het getoonde top-level transformatieschema is het proces opgedeeld in transformatieprocessen voor de windturbine en de *media*. Voor de windturbine is bovendien de opdeling in *turbinedelen* opgenomen, waarbij het bestaan van interfaces binnen de turbine ten behoeve van de overzichtelijkheid op alternatieve wijze is aangegeven. De met terminators weergegeven omgeving van de *simulator* is een herkenbare subset uit figuur 2.2.

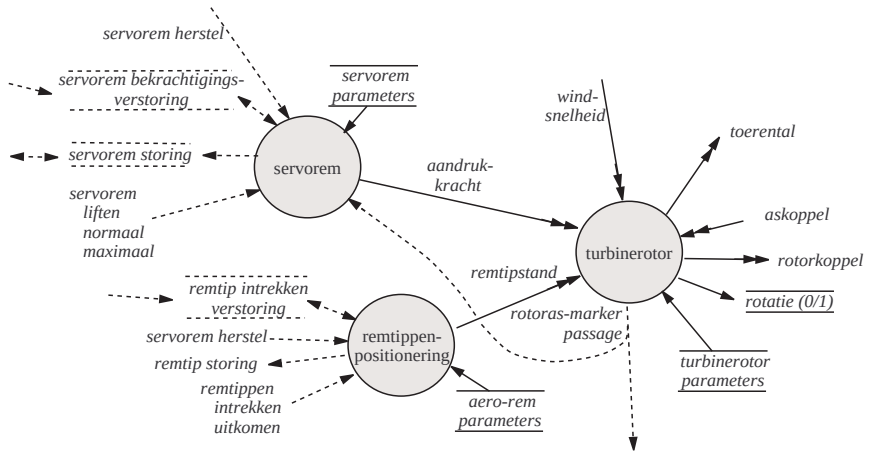
In de opzet van figuur 2.3 wordt het gedrag van de *simulatie-items* beïnvloed door:

- processignalen en status data binnen de *processimulator* (interactiesignalen);
- events en parameterwijzigingen van de operator;
- regel- en conditioneringssignalen uit het *besturingssysteem*.

Merk op dat het *medium* grond passief is en daarom volledig door parameters wordt gekarakteriseerd.

In hoofdstuk 3 worden de *modellen* voor de *media* en *turbinedelen* afgeleid voor een vereenvoudigde voorbeeldturbine. Hier wordt de daar toegepaste handelswijze inzichtelijk gemaakt via het *turbinedeel* rotor. Hierbij wordt aangenomen dat de rotor uit slechts één *systeem* bestaat, wat volstaat om de methode te illustreren. Dit *systeem* rotor zal in twee stappen tot elementaire processen wordt ontleed.

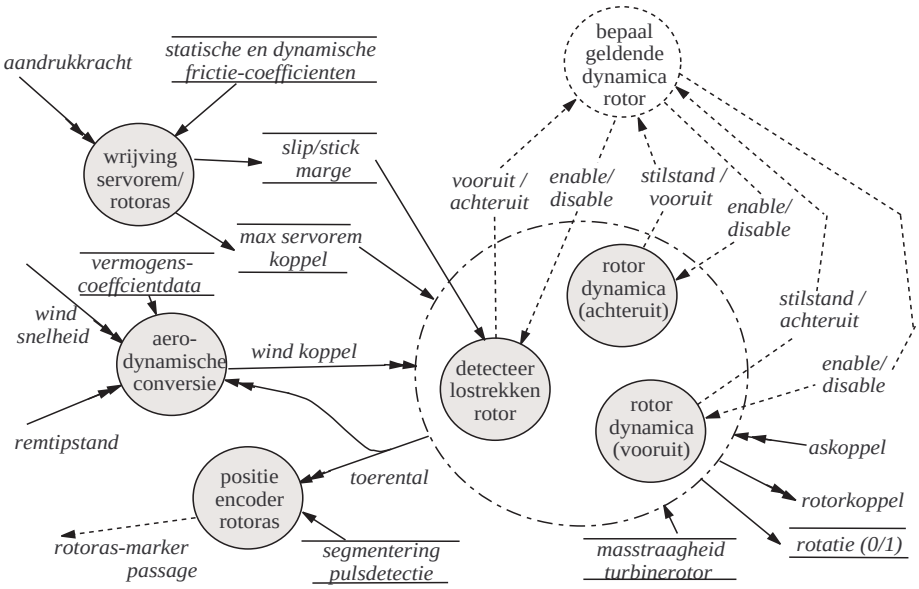
Figuur 2.4 toont het resultaat van de eerste ontledingstap voor de rotor.



Figuur 2.4 *Transformatieschema rotor*

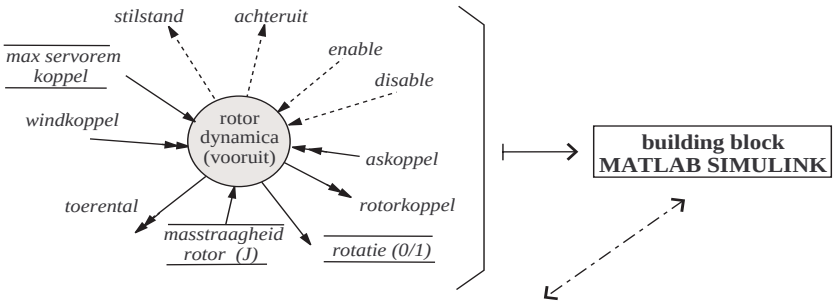
Deze figuur legt vast met welke *componentmodellen* het rotor*model* wordt samengesteld (te weten: servorem, remtippen-positionering en turbinerotor) en hoe deze *componentmodellen* zich ‘aan de grens’ kwalitatief zullen gedragen: wat er van de *modellen* door hun omgeving wordt waargenomen met betrekking tot I/O en werkingsprincipe.

Voor één *component* kunnen er meerdere *realisatie*gebonden *componentmodellen* worden opgesteld en in een *outlinelibrary* worden opgenomen, waaruit de operator van de *processimulator* dan kan kiezen. Figuur 2.5 toont een mogelijk resultaat van de tweede ontledingsstap met betrekking tot de *component* turbinerotor, waarin de elementaire processen worden onderscheiden die het *componentmodel* beschrijven.



Figuur 2.5 *Bottom-level transformatieschema turbinerotor*

Met het benoemen van de elementaire processen is het turbinerotor*model* opgedeeld tot een zogenaamd bottom-level transformatieschema: een samenstelsel van elementaire *dataprocessen* en toestandsbepalende *controlprocessen* die het *model* van een *component* belichamen. De volgende stap is de elementaire processen zodanig te specificeren, dat hieruit direct binnen MATLAB implementeerbare *componentmodules* kunnen worden opgesteld. Het dataproces 'rotor dynamica (vooruit)' is in figuur 2.6 op een dergelijke wijze gespecificeerd.

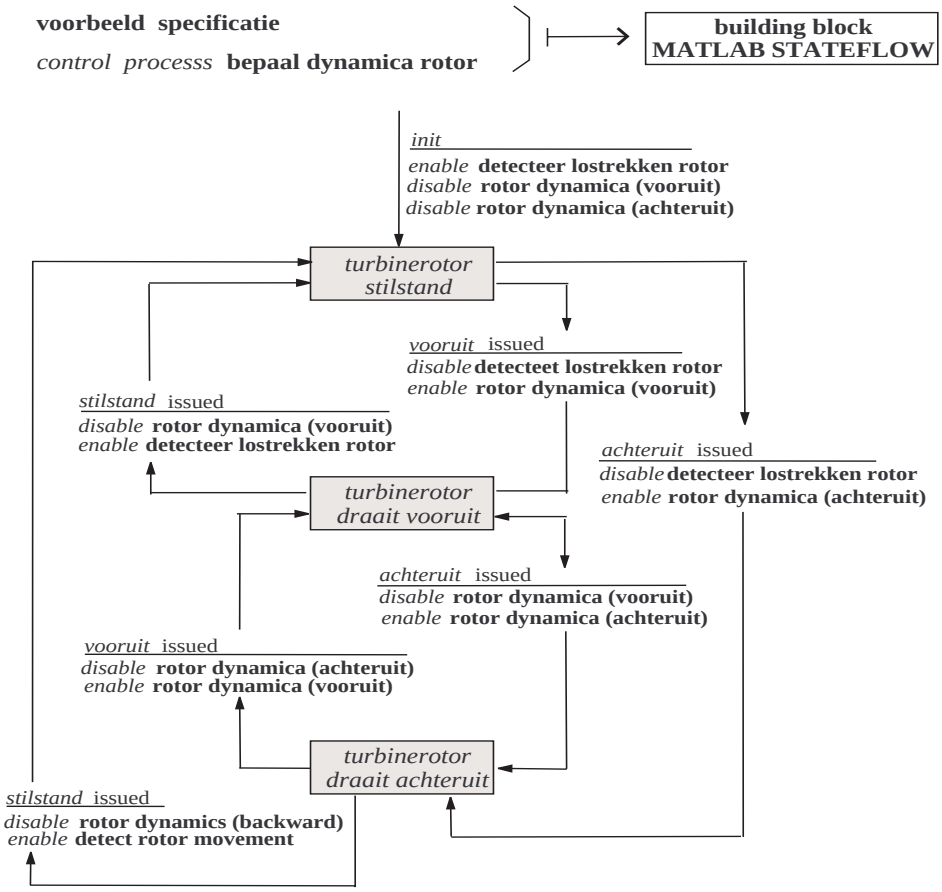



```
ALS (windkoppel - askoppel) < -( max servoremkoppel) EN toerental ~ 0 DAN
    ISSUE achteruit
ANDERS
    ALS (windkoppel - askoppel) < max servorem koppel EN toerental ~ 0 DAN
        rotatie := 0
        ISSUE stilstand
    ANDERS
        J d/dt (toerental) = windkoppel - askoppel - max servorem koppel
        rotorkoppel = windkoppel - max servorem koppel
    EINDE
EINDE
```

Figuur 2.6 *Specificatie voor dataproces 'rotor dynamica (vooruit)'*

Hieruit kan direct een *module* worden afgeleid die middels de SIMULINK-toolbox in MATLAB kan worden geïmplementeerd.

De *controlprocessen* ondergaan een soortgelijke ontleding. Figuur 2.7 toont het toestandsdiagram dat het *controlproces* 'bepaal geldende dynamica rotor' volledig specificeert.



Figuur 2.7 *Toestandsdiagram voor controlproces 'bepaal geldende dynamica rotor'*

Uit dit toestandsdiagram kan de *module* voor dit *controlproces* direct worden afgeleid en middels de STATEFLOW-toolbox in MATLAB worden geïmplementeerd.

3. SIMULATOR VOORBEELDTURBINE

In dit hoofdstuk worden de ontwikkelingsstappen uitgevoerd die van toepassing zijn voor een *processimulator* voor een sterk vereenvoudigde *turbine-layout*. Deze excercitie dient binnen deze voorstudie als opstap tot formalisatie van de *simulator*ontwikkeling in hoofstuk 4. Om de projectvoortgang te bespoedigen zijn gedeelten weggelaten waarbij niet de noodzaak bestond voor toelichting van of gedachtenvorming voor deze formalisatie.

De hier beschreven *simulatorontwikkeling* is gebaseerd op onderverdeling in vier hoofdtaken:

- inventarisatie: verkenning van de omgeving van de *processimulator*
- definitie: *component*- en *medium*eigenschappen binnen de *processimulator*
- modelvorming: toestandslogica en dynamica voor *componenten* en *media*
- implementatie: *modules* voor *componenten* en *media* en operator- en *besturings*interfaces

De 'omgeving' wordt hier beschouwd als het geheel aan factoren waaraan de eigenschappen van de te realiseren *processimulator* vooral kwalitatief getoetst kunnen worden. Hier betreft dit de conceptuele kenmerken van de turbine en de *besturingsstrategie*. (§3.1.)

In Bijlage A is de *besturingsstrategie* gedetailleerd tot een specificatie van het *besturingssysteem*. Hiermee werd beoogd om een juist gevoel te ontwikkelen voor vrijheidsgraden in *component*gedrag en interacties met de operator van de *processimulatie*. Het (potentiële) *component*- en windgedrag is dan ook vrij uitgebreid beschreven omdat alles wat binnen het *besturingssysteem* te testen valt uiteindelijk beperkt wordt door de vrijheidsgraden die hier geintroduceerd worden. (§3.2.)

Daarnaast is ook de modellering voor enkele aspecten tot in detail uitgewerkt. Dit is gedaan om duidelijk te maken wat een mogelijke weg is voor een daadwerkelijke modulaire opbouw van het simulatiegereedschap. (§3.3 t/m §3.6.)

De eigenlijke implementatie is in dit hoofdstuk niet verder uitgewerkt. De algemene werkwijze hierbij is reeds besproken in het vorige hoofdstuk (§2.2). Nadere detaillering en toepassing op de voorbeelds*imulator* wordt pas ter hand genomen bij uitvoering van de case study in het voorziene vervolgproject.

3.1 Inventarisatie: turbineconcept en besturingsstrategie

De (fictieve) turbine in deze voorstudie wordt beschreven door een zeer sterk vereenvoudigde *turbinerealisatie* met de volgende hoofdkenmerken:

- passieve overtrekregeling;
- direct drive generator;
- netkoppeling via softstarter voor één synchroon toerental;
- frictierem op rotoras en aerodynamische rem op rotorbladen.

Bij 12 m/s windsnelheid en een toerental van 25 rpm neemt de turbine 830 kW vermogen op; er wordt dan 750 kWe aan het elektriciteitsnet geleverd.

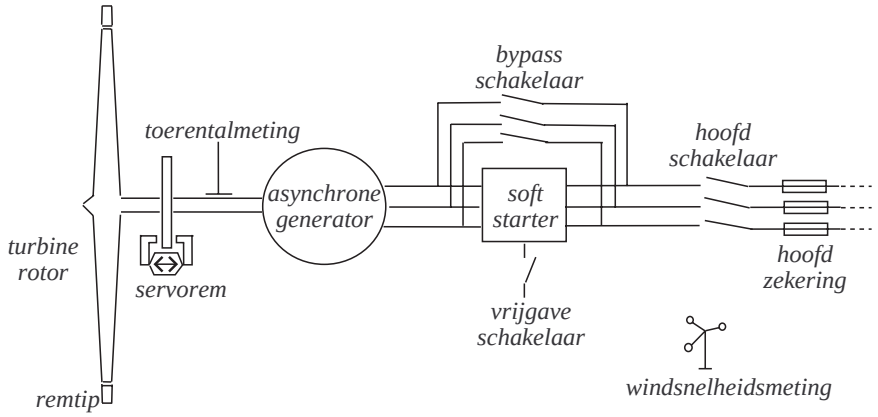
Het (fictieve) *besturingssysteem* hiervoor is gebaseerd op de volgende uitgangspunten (zie ook Bijlage A):

- automatische start-up op windsnelheidssignaal of handmatige start-up;
- gewone stop en noodprocedure op toerentalsignaal, windsignaal en/of operator;
- gewone stop op te lang duren synchroniseren;
- noodprocedure op te laat zijn / niet meer in orde zijn van netverbindingen;
- noodprocedure bij storingstoestand frictierem, aerorem of softstarter;
- laten doorbranden hoofdzekering bij niet waarneembare netontkoppeling;
- automatische start-up uitgesloten na een defectconditie.

Er worden geen *besturings*acties direct in gang gezet door mechanisch trillingen of het gedrag van het elektriciteitsnet.

3.2 Definitie: turbine-layout, componentgedrag en windgedrag

De *besturingsstrategie* en invulling van de turbine-realisatie hebben geleid tot een *turbine-outline* volgens fig. 3.1.



Figuur 3.1 *Outline voorbeeldturbine*

Aldus worden de volgende *componenten* in de *processimulator* opgenomen:

- 3-bladige turbinerotor met locaal bestuurde half-automatische remtippen;
- hydraulische servo rem met locale besturingseenheid op rotoras;
- asynchrone generator voor direct drive bedrijf;
- softstarter met bypass-schakelaar;
- hoofdschakelaar en hoofdzekering in serie;
- meetopnemers voor toerental en windsnelheid.

Daarnaast wordt natuurlijk de wind in beschouwing genomen.

Hieronder worden de wind en de *componenten* beschreven. Dit is beperkt tot het gedrag dat in de *simulator* naar voren moet komen. Alle *component*beschrijvingen zijn fictief; zij illustreren slechts de definitiefase binnen de ontwikkeling van de *processimulator*.

Het kruisysteem is weggelaten voor verlaging van de complexiteit.

3.2.1 Turbinerotor met remtippen

Samenstelling
De rotor bestaat uit 3 bladen met remtippen. De straal bedraagt 26.5 m. Het massatraagheidsmoment J_{r} om de hoofdas bedraagt $2.0{\cdot}10^{6}$ kgm^2; hierbij inbegrepen zijn de rotoras en de remschijf (zie §3.2.2).

De remtippen zijn voorzien van een locale besturingseenheid. De remtippen hebben een lengte van 1.5 m en breedte van 0.7 m.

Gedrag Rotor
De rotor kan voorwaarts draaien, stilstaan of achterwaarts draaien. De rotor inclusief rotoras wordt volledig star verondersteld. Het mechanisch gedrag wordt alleen bepaald door het massatraagheidsmoment J_{r} en de koppels door de wind, de frictierem op de rotoras en de generator.

De aerodynamische conversie met ingetrokken remtippen wordt in de basis beschreven door een kenmerkende ontwerpcurve voor de vermogensco ëfficiënt voor een overtrek geregelde turbine. Bij 25 rpm en 12 m/s neemt de rotor 830 kW aerodynamisch vermogen op. Modificaties zijn hierop aangebracht voor:

- voldoende startkoppel bij stilstand en lage toerentallen;
- achterwaartse rotatie vanaf stilstand bij scheve aanstroming.

Gedrag Remtippen
Bij uitgekomen remtippen wordt de aërodynamische conversie uitgebreid met voornamelijk weerstandskrachten aan de bladuiteinden. Zolang de tipsnelheid groter is dan de helft van de windsnelheid ($\lambda = 0.5$) ondervinden de remtippen alleen weerstandskrachten onder de aanname van het gedrag van een vlakke plaat. Bij lagere snellopendheid worden ook liftkrachten in rekening gebracht die dan de remfunctie langzaamaan overnemen.

Een veer-pal-mechanisme zorgt ervoor dat de remtippen altijd automatisch uitkomen bij 28 rpm. Dit uitkomen kan ten alle tijde bereikt worden met het signaal *remtip uitkomen*; hierbij wordt de opgaande flank in de locale besturingseenheid voor signaaldetectie gebruikt. Pas beneden 1 rpm kunnen de remtippen weer ingetrokken worden. Hiervoor is een opgaande flank in signaal *remtip intrekken* vereist. In ca. 3 seconden worden de remtippen teruggeplaatst achter de pal (reset-actie). Signalen *remtip uitkomen* en *remtip intrekken* komen vanuit het *besturingssysteem*.

De operator van de *processimulatie* kan het intrekken van de remtippen verstoren via het signaal *remtip intrekken verstoring*. Dit kan vooraf of tijdens de reset-actie opgelegd worden. De reset-actie, wanneer ook ingezet maar nog niet afgerond, wordt dan volledig ongedaan gemaakt; de locale besturingseenheid verhoogt hierbij de waarde van het signaal *remtip storing*. Na signaal *herstel remtip* van de operator vindt de reset-actie alsnog plaats en heeft de (oude) verstoring geen invloed meer. Signaaldetectie gebeurt hier ook op de opgaande flanken.

3.2.2 Hydraulische servorem

Samenstelling
De hydraulische servorem bestaat uit remklauwen, voorzieningen voor hydraulisch actuatie en een locale besturingseenheid. Verder is de remeenheid voorzien

van meetopnemers voor de aandrukkracht van de remklauwen op de remschijf en het toerental. Binnen de *besturingseenheid* is een algoritme opgenomen voor servoregeling van de aandrukkracht. Benodigde basisgegevens voor de besturing zijn het totale massatraagheidsmoment om de hoofdas en de remschijfeigenschappen straal, materiaalsoort en ruwheid.

Gedrag
Twee soorten remgedrag moeten zich kunnen doen gelden binnen de simulator:

- invariant remkoppel van 1MNm onder rotatie (3 keer nominaal);
- eenparig remmen met 1.0 rpm/s tot aan 0.5 rpm, daaronder 1MNm remkoppel.

Het eerste remgedrag (stopprocedure) wordt opgelegd via signaal *servorem normaal*; het tweede (noodprocedure) via *servorem maximaal*. Het lichten van de rem gebeurt via signaal *servorem lift*. Signaaldetectie gebeurt op de opgaande flanken.

De operator van de *processimulatie* kan de werking van de servorem verstoren via het signaal *verstoring servorem*. Dit kan vooraf of tijdens een remactie opgelegd worden. Er is vanaf dat moment geen (potentieel) remvermogen meer; de locale besturingseenheid verhoogt hierbij de waarde van het signaal *servorem storing*. De rem kan pas weer bekrachtigd worden na operatorsignaal *herstel servorem*; vanaf dan is de verstoring niet meer geldig.

3.2.3 Asynchrone generator

Samenstelling
De turbine is uitgevoerd met een langzaamlopende 'direct drive' asynchrone generator. Het massatraagheidsmoment J_{g} om de hoofdas bedraagt $0.2{\cdot}10^{6}$ kgm^2.

Gedrag
Het mechanisch gedrag wordt alleen bepaald door het massatraagheidsmoment J_{g} en de koppels in de generatoras en in de luchtspleet van de generator.

Wanneer netgekoppeld is de koppel-toerencurve geldig volgens de stationaire relatie van Kloss onder een nominale slip van 0.5%. Het nominale toerental bedraagt 25.12 rpm, hetgeen een nominaal elektrisch koppel oplevert van 286 kNm bij een generatorspanning 690 V. Er is slechts één (synchroon) bedrijfstoerental.

Het koppelniveau wordt evenredig met de spanning in het kwadraat verondersteld; dit is van belang voor het gedrag bij aanlopende softstarter. Dynamisch wordt een eerste orde verband verondersteld tussen toerental- en koppelvariaties, waarbij de tijdconstante 0.1 s bedraagt.

De operator van de *processimulatie* kan in deze case-opzet de werking van de generator niet verstoren.

3.2.4 Hoofdschakelaar en bypass schakelaar

Samenstelling
Beide schakelaars zijn uitgerust met relais voor bekrachtiging en met standmelders. De relais werken direct op stuurspanningniveau's en de standmelders worden gerealiseerd door potentiaalvrije schakelcontacten die in een extern circuit zijn

genomen. De duur van de spanningsopbouw in de spoelen van de relais is niet verwaarloosbaar.

Gedrag
Het openen/sluiten van de hoofdschakelaar ('main') gebeurt via de waarde 0/1 van ingangssignaal *set main*. De hoofdschakelaar stand wordt teruggemeld via de waarde 0/1 (open/gesloten) van uitgangssignaal *main stand*. De overgang van de ene naar de andere schakelaarstand wordt beschouwd als een zuivere looptijd. Deze bedraagt 0.2 s voor sluiten en 0.1 s voor openen. Naar deze schakelaarstand-overgangen wordt verwezen als de *schakeltransiënten openen* en *sluiten*.

De bypass schakelaar werkt analoog via de signalen *zet bypass* en *bypass stand* met respectievelijke delays van 0.15 en 0.1 s voor sluiten en openen.

De operator van de simulatie kan de schakelaarwerking verstoren via signalen *schakelaar verstoring open* en *schakelaar verstoring dicht*. Zo'n storing kan vooraf of tijdens schakelacties opgelegd worden. Dit kan weer ongedaan gemaakt worden via signaal *schakelaar herstel open* of *schakelaar herstel dicht*. Deze operatorsignalen worden gedetecteerd naar stijgende flank. De specifieke effecten van de stoor- en herstelsignalen zijn als volgt:

- Signaal *schakelaar verstoring open* leidt ertoe dat een geopende schakelaar instantaan gesloten wordt (stand-waarde 1) en dat een gesloten schakelaar dicht blijft onafhankelijk van de zet-waarde. Signaal *schakelaar herstel open* leidt **bij zet-waarde 0** tot de schakeltransiënt openen.
- Signaal *schakelaar verstoring dicht* leidt ertoe dat een gesloten schakelaar instantaan geopend wordt (stand-waarde 0) en dat een geopende schakelaar open blijft onafhankelijk van de zet-waarde. Signaal *schakelaar herstel dicht* leidt **bij zet-waarde 1** tot de schakeltransiënt sluiten.

3.2.5 Hoofdzekering

Samenstelling
De set hoofdzekeringen tussen elektriciteitsnet en hoofdschakelaar brandt door bij te sterke warmte-ontwikkeling. De nominale stroom door de zekeringset bedraagt 1090 A.

Gedrag
De zekeringset geleidt of geleidt niet. Dit wordt kenbaargemaakt via de waarde 1 of 0 van van uitgangssignaal *zekering geleiding*.

Indien de stroom door de zekeringset 1.5 keer nominaal of hoger bedraagt gedurende 3 seconde of meer brandt de zekering door. De operator van de *simulatie* kan de zekeringgeleiding instantaan verstoren via signaal *zekering defect*.

Het operatorsignaal *zekering herstel* dient ertoe om de zekeringset weer in geleiding te brengen na doorbranden of opgelegde storing door de operator.

3.2.6 Softstarter

Samenstelling
De softstarter bevat vermogenselektronica voor realisatie van een basisharmonische spanning (netfrequentie) met een regelbaar effectief voltage dat lager is

dan de netspanning. Hieraan is een *besturings*algoritme toegevoegd waarmee de effectieve uitgangsspanning in een bepaalde tijd kan worden op- of afgebouwd tussen de netspanning en een ondergrens. Eigenlijk is er sprake van de netspanning **na transformatie**; de transformatoren zijn niet opgenomen in deze *simulator*-specificatie.

Gedrag
Een opgaande flank in het ingangssignaal *vrijgave* (0→1) activeert de softstarter. Zodra de softstarter de ondergrensspanning heeft gerealiseerd krijgt signaal *softstarter bedrijf* de waarde 1. Nadat de spanning opgebouwd is tot het netniveau krijgt ook signaal *koppeling volledig* de waarde 1. Het in bedrijf gaan duurt 0.1 s; de spanningsopbouw van de ondergrens van 200 Volt tot het eindniveau van 690 V duurt 5 s.

Een neerwaartse flank in het ingangssignaal *vrijgave* (1→0) initieert de spanningsafbouw en afschakeling van de softstarter. Zodra de spanning begint af te nemen krijgt signaal *koppeling volledig* weer de waarde 0; de afbouw tot de ondergrens duurt 2 s. Het daarop volgende uit bedrijf gaan van de softstarter duurt 0.1 s; daarna krijgt signaal *softstarter bedrijf* de waarde 0.

De operator van de *simulatie* kan de softstarterwerking verstoren en weer herstellen in de respectieve fases in bedrijf gaan, spanningsopbouw, spanningsafbouw en uit bedrijf gaan. De signaalbenaming hiervoor is:

- *softstarter verstoring/herstel <bedrijfsfase>*.

Elk van deze storingen kan vooraf of tijdens de bedrijfsfase waarin die effect heeft opgelegd worden. De bedrijfsfase ‘wordt opgehouden’ tot het betreffende herstelsignaal gegeven is. Deze operatorsignalen worden gedetecteerd naar stijgende flank.

Tevens kan de operator een algemeen storings/herstelsignaal geven:

- *softstarter verstoring/herstel werking algemeen*

Op het moment van deze verstoring gaat de sofstarter direct uit bedrijf. Herstel leidt ertoe dat de softstarter in de uitgangspositie voor in bedrijf gaan terecht komt: waarde 1 voor *vrijgave* leidt direct tot een nieuwe netkoppelingsprocedure.

3.2.7 Meetopnemers toerental en windsnelheid

Samenstelling
Het toerental wordt gemeten via een positie-encoder. De rotoras draait via een overbrenging van 10:1 de encoder-as aan; hiervoor wordt een klein tandwielkastje gebruikt. Op de encoder-as zit een marker ten gevolge waarvan 10000 pulsen per *rotoras*-omwenteling worden gegeneerd.

De windsnelheid wordt gemeten met een cup-anemometer.

Gedrag
Het toerental wordt bepaald door het aantal pulsen dat passeert in 0.1 s te delen door 0.1 s. Hierdoor ontstaat een toevallige fout van +/- 0.06 rpm.

De windsnelheid wordt met een nauwkeurigheid van 0.1 m/s gemeten met een bandbreedte van 5 Hz.

3.2.8 Windveld

Het windgedrag dient stochastisch te zijn en wel zo dat rotoreffectiviteit nagestreefd wordt voor het windsignaal dat via de vermogenscoefficient van de rotor gebruikt windkoppel genereert. Dit wil zeggen dat het effect van het éénpuntswindsignaal op het rotorkoppel goeddeels overeen moet komen met dat van een stochastisch *windveld op de rotor 'als ruimtelijke structuur'*.

Voor generatie van de turbulentie wordt uitgegaan van een naafhoogte van 50m. Windshear, turbulentie-intensiteit, coherentie over het rotorvlak en frequentieverdeling van de turbulentie dienen overeen te stemmen met de IEC-normen. Het effect van torenpassage dient gebaseerd te zijn op de stromingsbeschrijving rond een semi-infinite dipool volgens de potentiaaltheorie.

Naast dit rotoreffectieve windsignaal dient het hieraan verwante windsignaal ter plekke van de windsnelheidsmeter gerealiseerd te worden. Dit signaal heeft de stochastische eigenschappen van de windsnelheid op een vaste positie.

De operator van de simulator kan richting en grootte van de gemiddelde windsnelheid beinvloeden en kan een vlaag aanbrengen op het stochastische rotoreffectieve signaal in de vorm van een zogeheten Mexicaanse-hoed.

3.3 Modelvorming (1): interacties met omgeving en hoofdfuncties

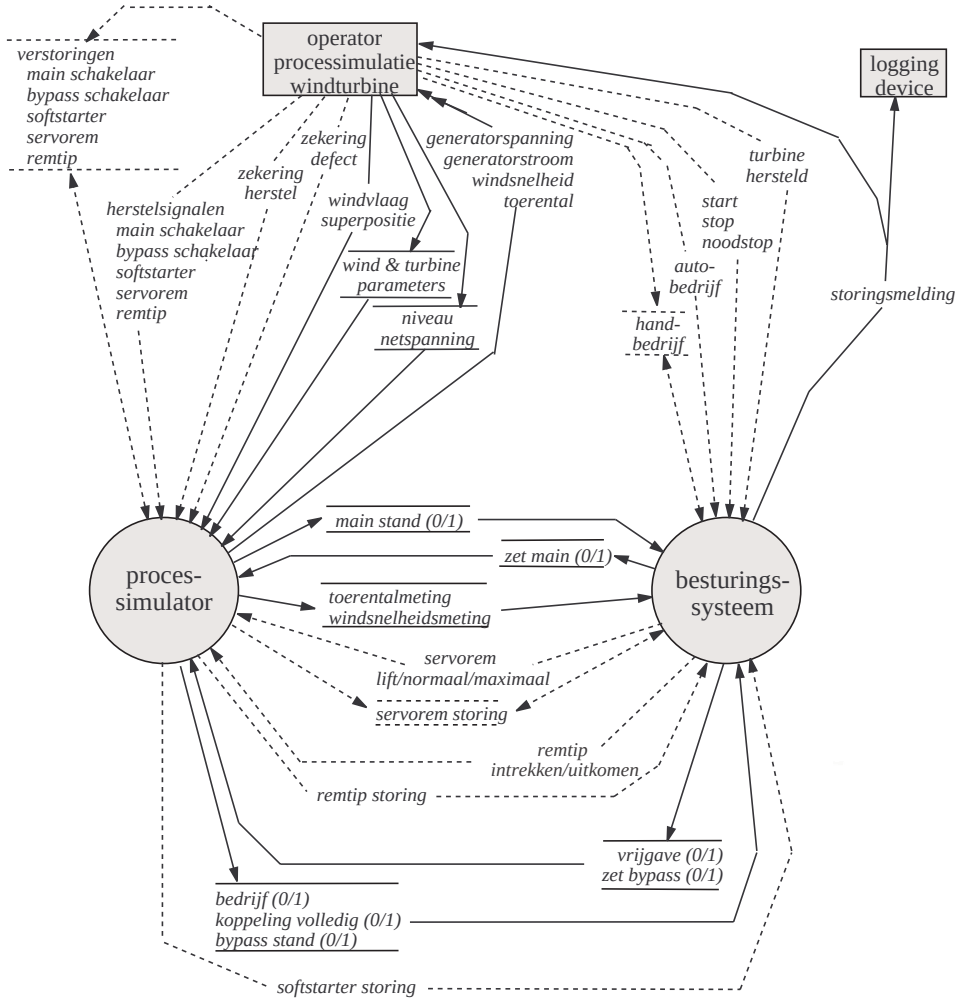
Figuur 3.2 geeft het context diagram ten behoeve van de uit te voeren *processimulaties*. Dit is samengesteld naar aanleiding van het algemene context diagram in fig. 2.2 en van alle te onderkennen signalen in de gedragsbeschrijvingen voor de *componenten* en de wind uit §3.2.1 t/m §3.2.8. De vierhoekige *terminators* vormen de 'context' in de zin van de relevante signaalgevers en -ontvangers; dit zijn dus alleen de operator van de *processimulatie* en een logging device.

De turbinebeschrijving in §3.2 en de algemene *simulator*weergave als proces in fig. 2.3 leidt tot het top level *transformatieschema* voor de processimulator in fig. 3.3.

Hierin zijn de turbine*componenten* als volgt gegroepeerd naar de hoofd*componenten* **rotorsysteem**, **generator** en **ondersteuningsconstructie**:

- **rotorsysteem**
  - turbine rotor
  - aerodynamische remtippen
  - hydraulische servorem
- **generatorsysteem**
  - asynchrone generator
  - softstarter
  - hoofdschakelaar
  - bypass schakelaar
  - hoofdzekering
- **ondersteuningsconstructie**
  - positie-encoder voor toerental
  - anemometer voor windsnelheid

Dienovereenkomstig zijn de *component*signalen verdeeld tussen deze drie hoofd*componenten* en is er sprake van interfacing hiertussen.



Figuur 3.2 *Context diagram voor processimulatie met besturingssysteem*

Het hierin voorkomende interfaceproces tussen het **rotorsysteem** en *generatorsysteem* wordt gespecificeerd in §3.3.1. Er is geen sprake van andere interfaceprocessen binnen het top level schema.

3.3.1 Interface rotorsysteem ↔ generatorsysteem

Het interface tussen rotor en generator is hieronder gespecificeerd. Dit is geldig voor een starre koppeling tussen generator en rotor.

Proces **interface rotor / generator**

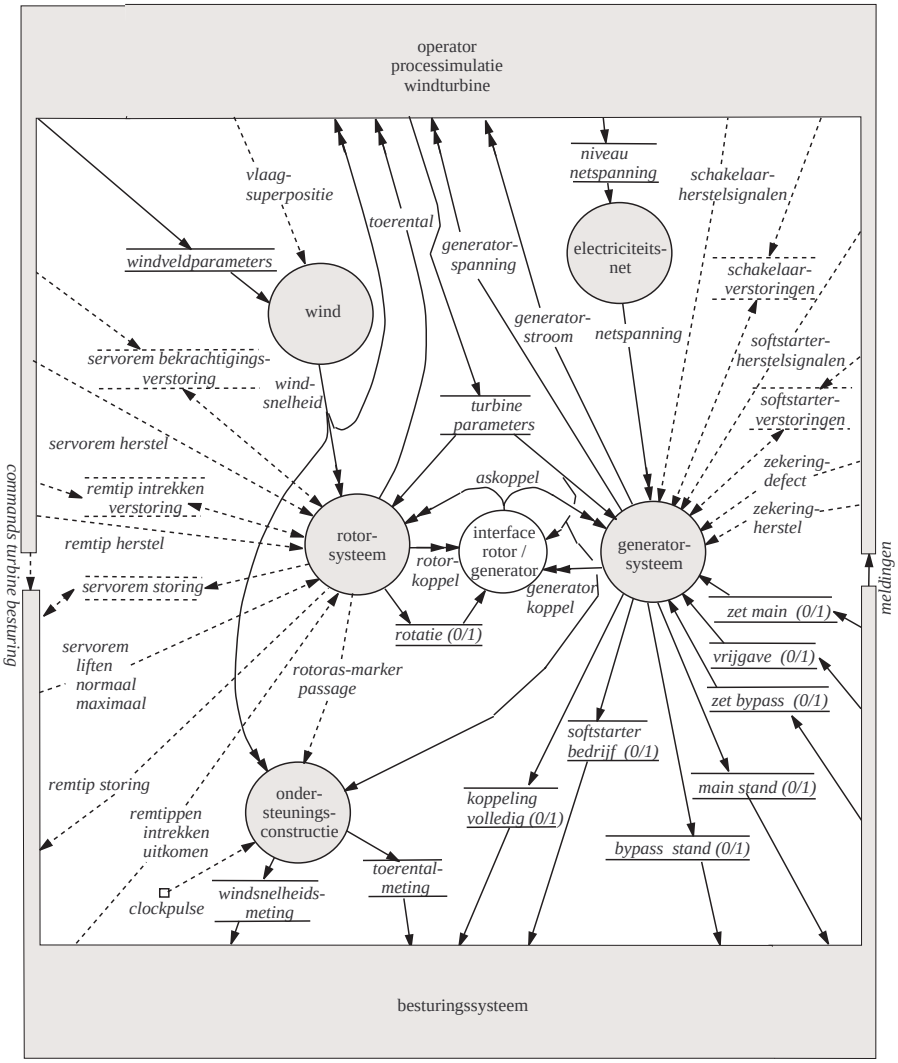
ALS *rotatie == 0* DAN

askoppel = generatorkoppel

ANDERS

*askoppel = Jg / (Jr + Jg) * rotorkoppel + Jr / (Jr + Jg) * generatorkoppel*

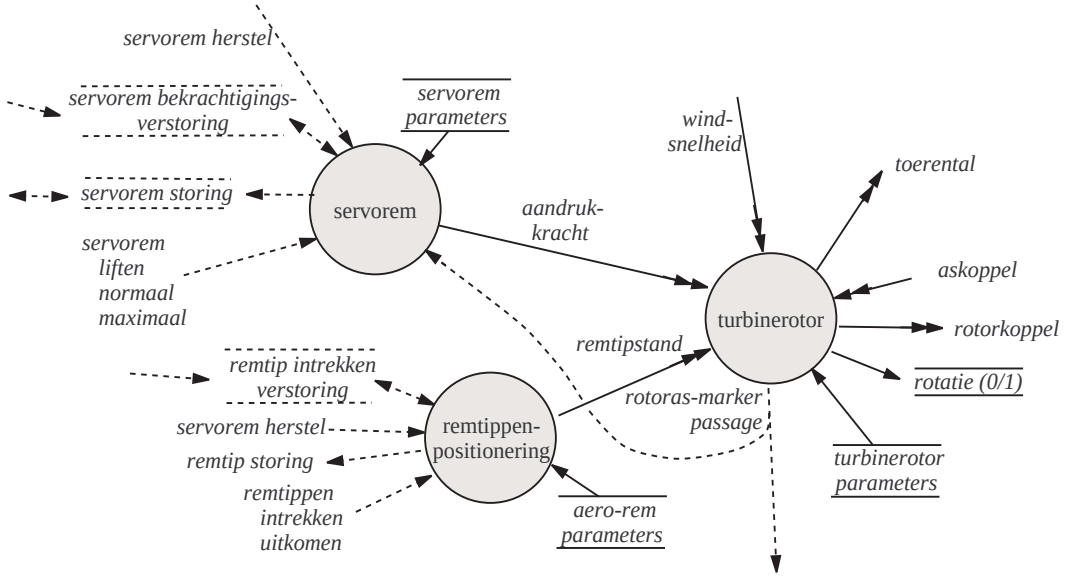
EINDE



Figuur 3.3 *Top level transformatieschema processimulator voorbeeldwindturbine*

3.4 Modellering (2a): rotorsysteem

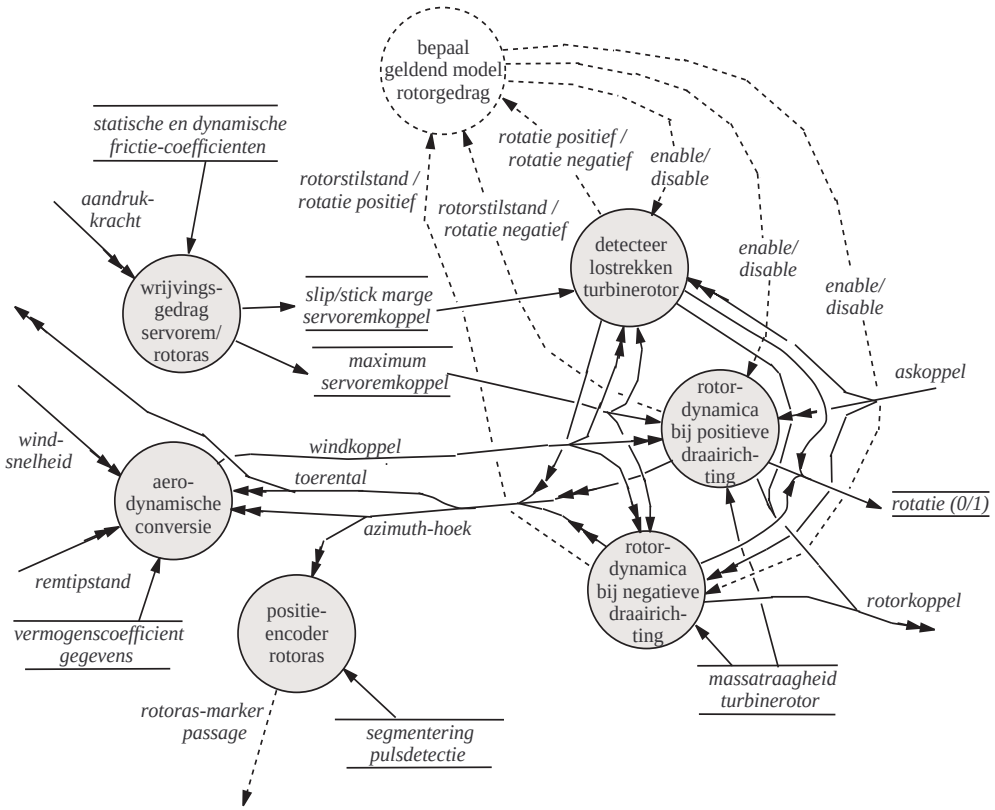
Het rotorsysteem wordt in twee stappen ontleed. Fig. 3.4 geeft het resultaat van de eerste stap waarin elke cirkel het gedrag van een bepaalde rotor*component* modelleert. De tweede stap resulteert in samenstelsels van elementaire processen voor de eigenlijke turbinerotor in §3.4.1, voor de hydraulische servorem in §3.4.2 en voor de aerodynamische remtippen in §3.4.3.



Figuur 3.4 *Transformatieschema rotorsysteem in processimulator*

3.4.1 Turbinerotor

Fig. 3.5 geeft het *transformatieschema* voor de turbinerotor zelf terwijl de overgang van bedrijfstoestanden in de turbinerotor vastgelegd wordt door het *toestandsdiagram* in fig. 3.6.



Figuur 3.5 *Transformatieschema turbinerotor binnen rotorsysteem*

Proces **aërodynamische conversie**

windkoppel = FUNCTIE*(remtipstand, windsnelheid, toerental, azimuth-hoek)*

FUNCTIE implementeert rotor-geintegreerde bladelement-impuls theorie (Cp-curves).

Proces **wrijvingsgedrag servorem/rotoras**

*servoremkoppel = wrijvingscoeficient * rotorasdiameter * aandrukkkracht*
*slip/stick marge servoremkoppel = fractie * maxium servoremkoppel*

Proces **positie-encoder rotoras**

ALS *azimuth-hoek* OVERBRUGT *rotorassegement* DAN
  ISSUE *rotorasmarker passage*
EINDE

Proces **detecteer lostrekken turbinerotor**

ALS *(windkoppel - askoppel)* $>$ *maximum wrijvingskoppel* DAN
  rotatie := 1
  ISSUE *rotatie positief*
ANDERS
  ALS *(windkoppel - askoppel)* $<$ *-(maximum wrijvingskoppel)* DAN
    rotatie := 1
    ISSUE *rotatie negatief*
  ANDERS
    ddt*(toerental) = 0*
    ddt*(azimuth-hoek) = 0*
    rotorkoppel = 0
  EINDE
EINDE

Proces **rotordynamica bij positieve draairichting**

ALS *(windkoppel - askoppel)* $<$ *-(maximum wrijvingskoppel)*
  EN *toerental* $<$ *eps* DAN
  ISSUE *rotatie negatief*
ANDERS
  ALS *(windkoppel - askoppel)* $<$ *maximum wrijvingskoppel*
    EN *toerental* $<$ *eps* DAN
    rotatie := 0
    ISSUE *rotor stilstand*
  ANDERS
    ddt*(toerental) = windkoppel - askoppel - maximum wrijvingskoppel*
    ddt*(azimuth-hoek) = toerental*
    rotorkoppel = (windkoppel - maximum wrijvingskoppel)/J_{r}
  EINDE
EINDE

Proces **rotordynamica bij negatieve draairichting**

ALS *(windkoppel - askoppel)* $>$ *+(maximum wrijvingskoppel)*
  EN *toerental* $>$ *-eps* DAN

ISSUE *rotatie positief*

ANDERS

ALS *(windkoppel - askoppel) > -(maximum wrijvingskoppel)*
EN *toerental > -eps* DAN

rotatie := 0
ISSUE *rotor stilstand*

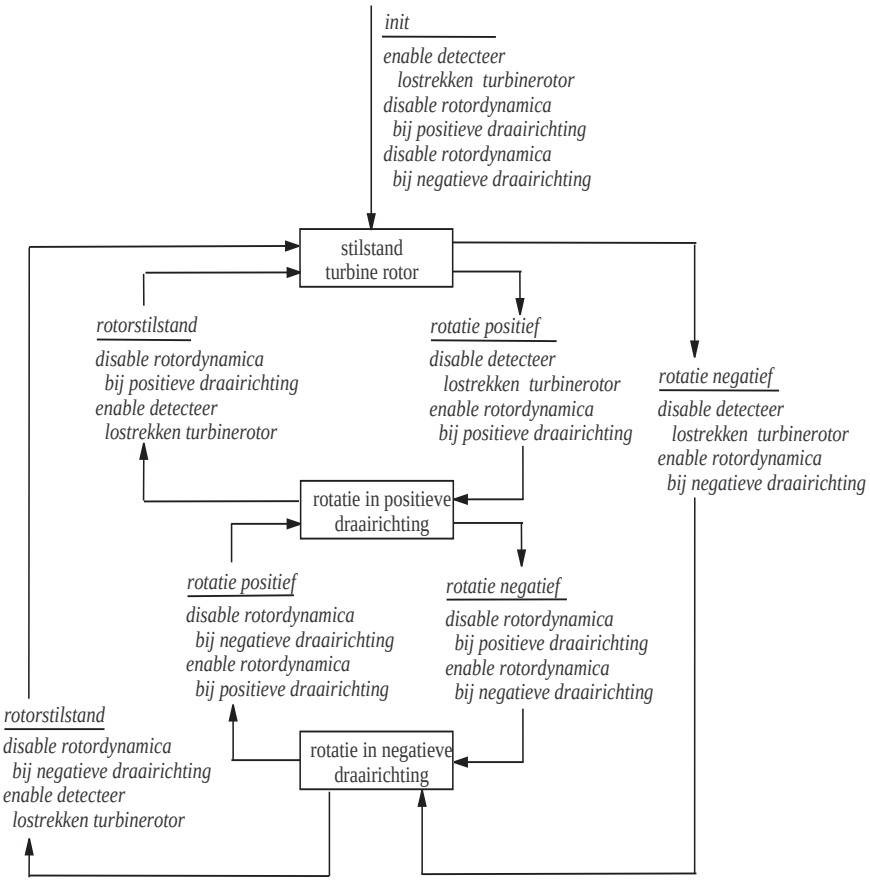
ANDERS

ddt*(toerental) = (windkoppel - askoppel - (-(maximum wrijvingskoppel)))/*J_r
ddt*(azimuth-hoek) = toerental*
rotorkoppel = windkoppel - (-(maximum wrijvingskoppel))

EINDE

EINDE

Proces **bepaal geldend model rotorgedrag**



Figuur 3.6 *Toestandsdiagram turbinerotor binnen rotorsysteem*

3.4.2 Hydraulische servorem

Het *proces* **servorem** wordt vooralsnog in het kader van dit voorbeeld niet verder uitgewerkt.

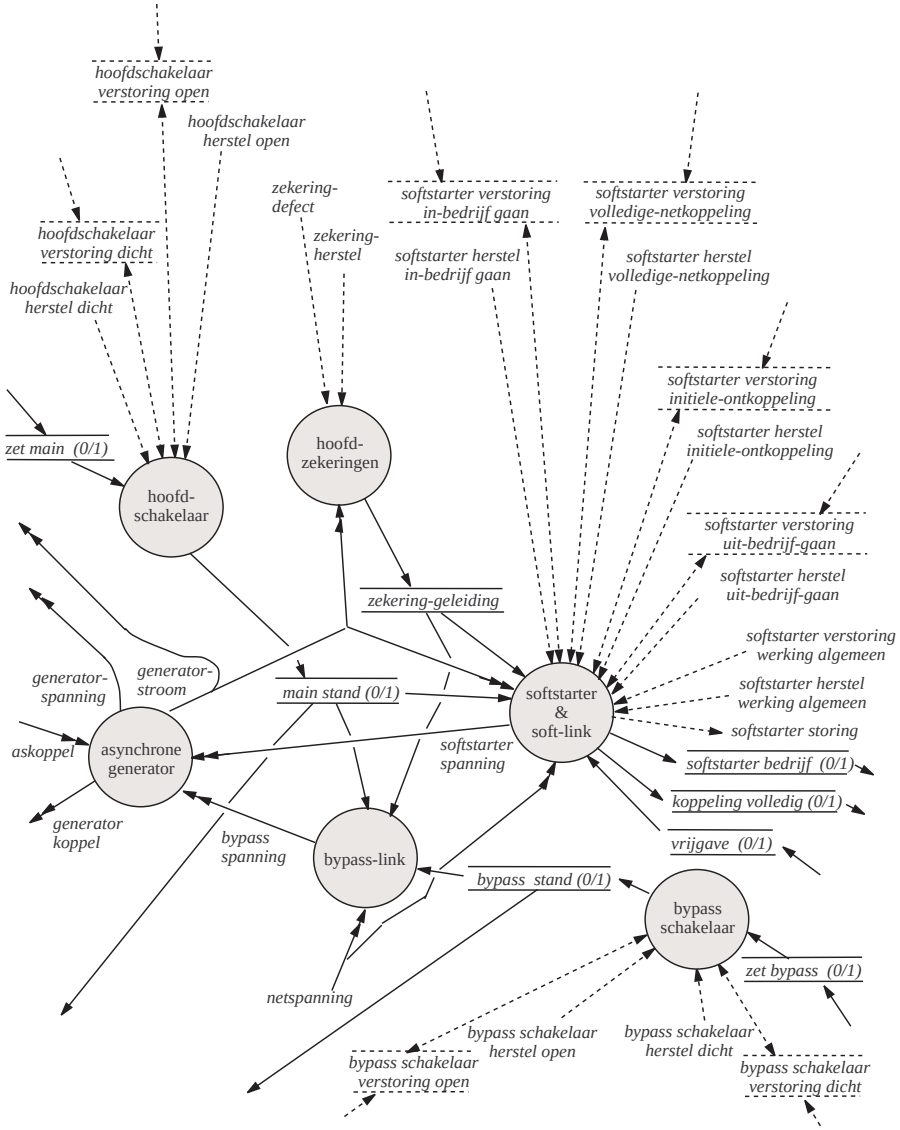
3.4.3 Aerodynamische remtippen

Een remtip-intrek-actie, ten gevolge van operatorsignaal *remtippen intrekken*, wordt eenmalig ongedaan gemaakt als het operatorsignaal *remtip intrekken verstoring* actief is geweest vooraf aan deze 'reset-actie' of actief is tijdens de reset-actie. Om deze uitwerking van het signaal *verstoring remtip* te realiseren wordt aan de *simulator*-module voor de remtippen een stoorsignaal-verwerkingeenheid toegevoegd. Deze heft een interne vlag bij een opgaande flank in het stoorsignaal van de operator (bit:=1). Na het signaal *remtip intrekken* wordt bij geheven storingsvlag deze reset-actie volledig ongedaan gemaakt, waarna de storingsvlag wordt gestreken (bit:=0).

Het *proces* **positioneren remtippen** wordt vooralsnog in het kader van dit voorbeeld niet verder uitgewerkt.

3.5 Modellering (2b): generatorsysteem

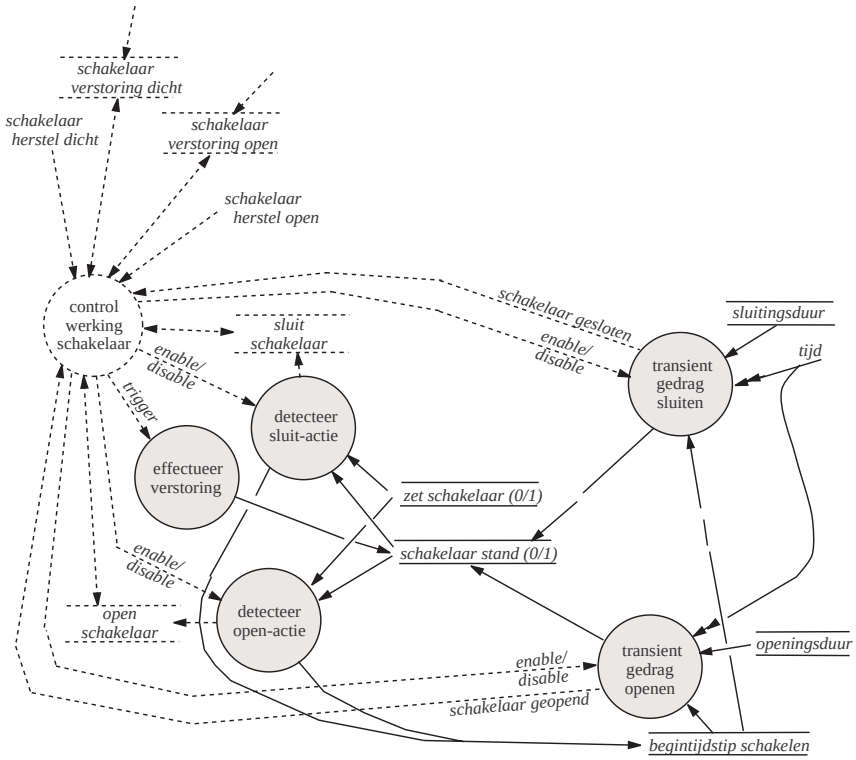
Het generatorsysteem wordt in twee stappen ontleed. Fig. 3.7 geeft het resultaat van de eerste stap. De tweede stap resulteert in samenstelsels van elementaire processen voor de hoofdschakelaar en bypass schakelaar in §3.5.1, voor de softstarter in §3.5.2 en voor de generator in §3.5.3.



Figuur 3.7 *Transformatieschema generatorsysteem in processimulator (parameter-data weggelaten)*

3.5.1 Bypass- en hoofdschakelaar

Fig. 3.8 geeft het *transformatieschema* dat zowel geldig is voor de bypass schakelaar als hoofdschakelaar.



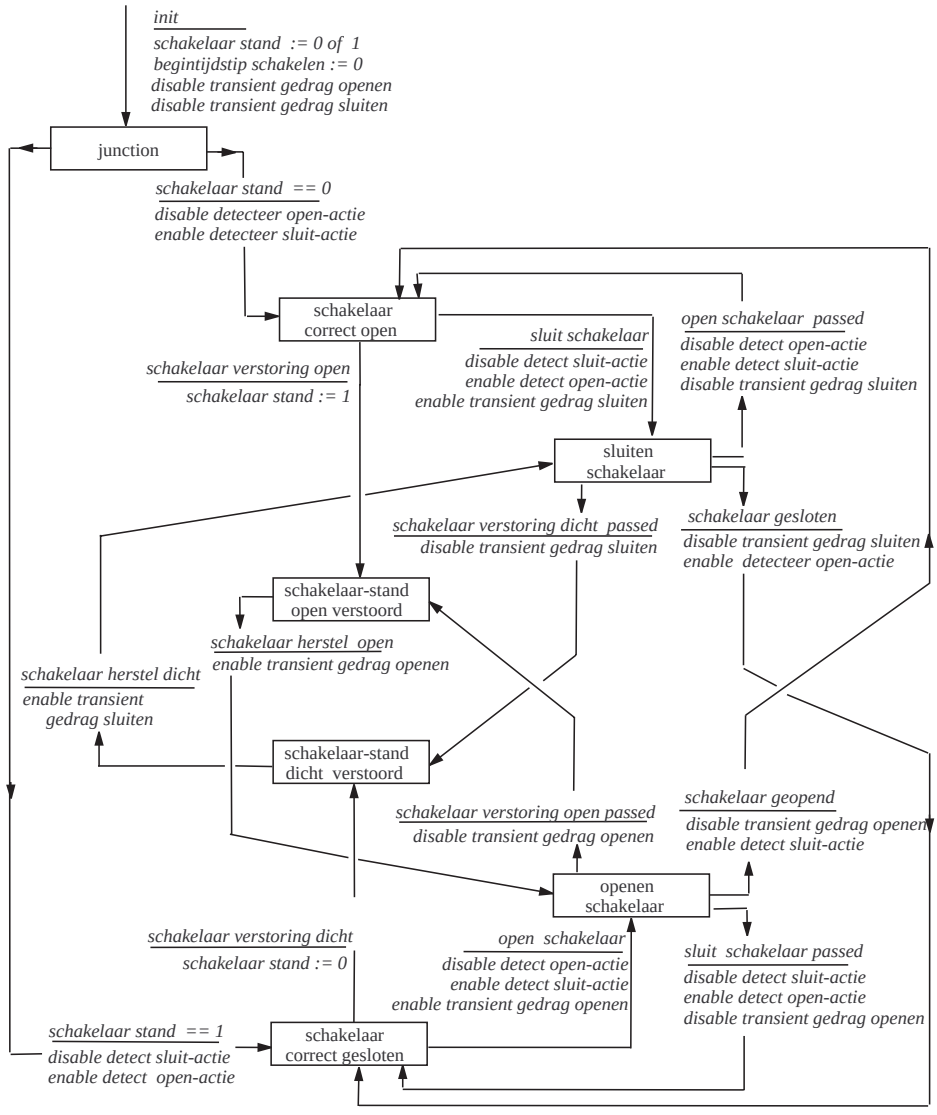
Figuur 3.8 *Transformatie schema voor simulatie hoofd- en bypassschakelaar*

De processen met enable/disable-conditionering in figuur 3.8 zijn tijdcontinue datatransformaties; dit betekent dat de ingangs/uitgangs-mapping zich doet gelden via tijd-continue relaties wanneer *enabled*. Wanneer *disabled* vindt geen I/O-mapping plaats; de uitgangen (behalve uitgangsevents) houden de waarde op het moment van disabling. Bij realisatie van de *simulator* zullen deze tijdcontinue processen toch ook tijddiscreet uitgevoerd worden maar met een rekencyclustijd die kleiner is dan de regelcyclustijd.

De processen met trigger-conditionering in figuur 3.8 zijn algoritmische datatransformaties die na een trigger precies 1 keer uitgevoerd worden.

De overgang van bedrijfstoestanden in een schakelaar wordt vastgelegd door het *toestandsdiagram* in fig. 3.9 voor het *controlproces* **control werking schakelaar**

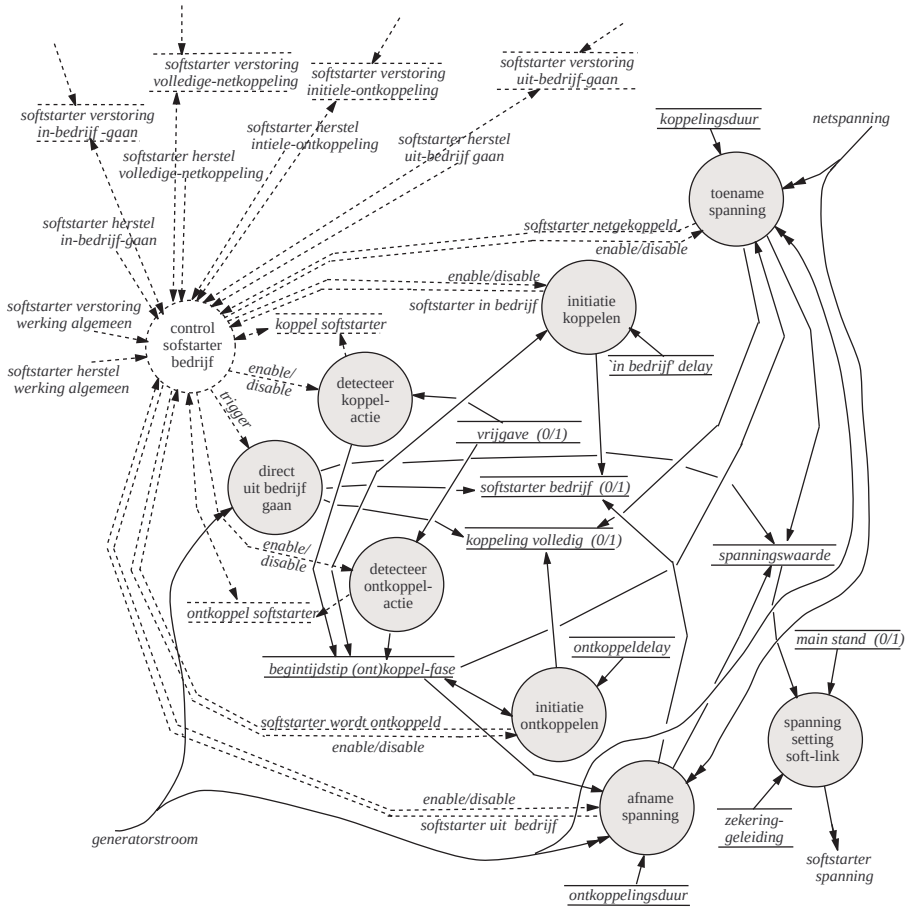
Proces **control schakelaarwerking**



Figuur 3.9 *Toestandsdiagram voor control schakelaarwerking*

3.5.2 Softstarter

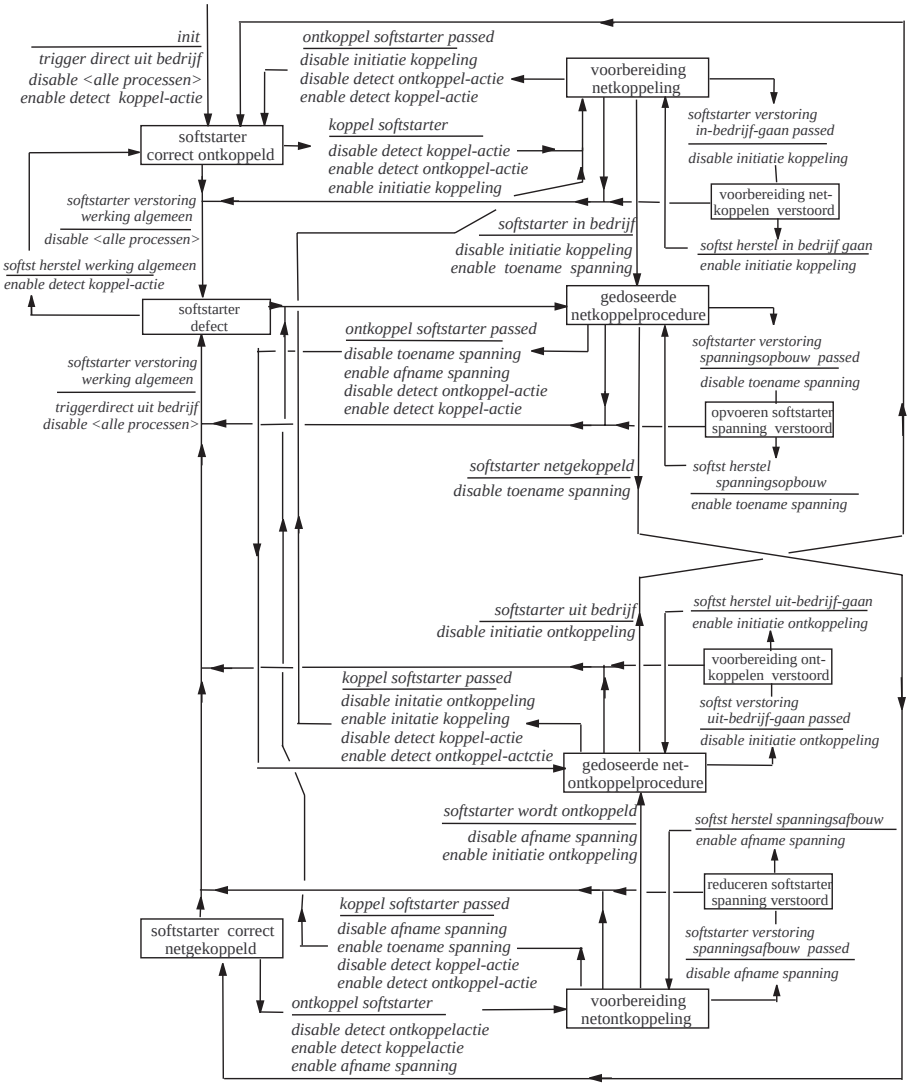
Fig. 3.10 geeft het *transformatieschema* dat geldig is voor de softstarter.



Figuur 3.10 *Transformatie schema voor simulatie softstarter*

De overgang van bedrijfstoestanden in een schakelaar wordt vastgelegd door het *toestandsdiagram* in fig. 3.11 voor het *controlproces* **control werking softstarter**

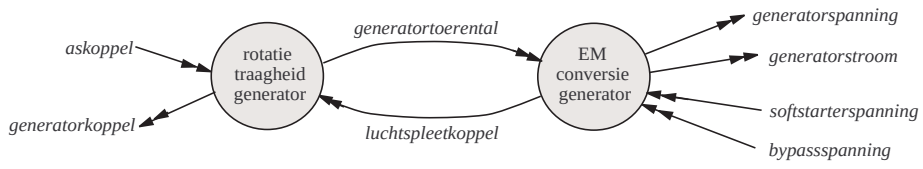
Proces **control werking softstarter**



Figuur 3.11 *Toestandsdiagram voor control softstarterwerking*

3.5.3 Asynchrone generator

Fig. 3.12 geeft het *transformatieschema* voor de asynchrone generator. Dit schema bevat geen *controlproces* in tegenstelling tot figuur 3.5 voor de turbinerotor omdat er geen mechanische wrijving is gemodelleerd wat tot verschillende rotatie-toestanden zou leiden.



Figuur 3.12 *Transformatieschema asynchrone generator (parameter-data weggelaten)*

3.6 Modellering (2c): windrealisatie

Windmodellering:

- De te meten windsnelheid betreft het windsignaal in het centrum van de rotorschijf met looptijd volgens Taylor's hypothese (afstand gedeeld door gemiddelde windsnelheid).
- Het rotoreffectieve windsignaal wordt bepaald door de (gewogen) integraal over de rotorschijf van het windveld onder in achtname van de roterende bemonstering van dat windveld door de rotorbladen; en dit gegeven de coherentie-functie tussen en de frequentieverdeling van de individuele punten in het windveld van de rotorschijf (zie [1]).

4. FORMALISATIE SIMULATORONTWIKKELING

In dit hoofdstuk worden de ontwikkelingsstappen in het voorbeeld van hoofdstuk 3 functioneel omschreven tot een uitvoerbaar ontwikkeltraject: de formalisatie van de ontwikkeling van de *processimulator*. Hoewel de formalisatie sterk gebaseerd is op de bevindingen uit hoofstuk 3, is bij het formuleren van de uiteindelijke werkwijze rekening gehouden met complexere en uitgebreidere *windturbine-realisaties* dan die van de voorbeeldturbine.

Reeds bij uitwerking van de voorbeeldsimulator werden vier hoofdtaken onderscheiden. Een vijfde hoofdtaak, real-time implementatie en case study, is hieraan toegevoegd. De case study zal gericht zijn op een *windturbine-realisatie* die past binnen de algemeen gangbare *concepten*, zoals deze tijdens de voorgaande hoofdtaken zijn geïdentificeerd en uitgewerkt. Onderstaande hoofdtaken zijn gedefinieerd:

1. Inventarisatie;
2. Definitie;
3. Modelvorming;
4. Implementatie;
5. Real-time implementatie en case study.

De verdeling in hoofdtaken geeft in principe een opeenvolging van ontwikkelingen weer, waarbij de output van iedere desbetreffende hoofdtaak de input voor de volgende vormt. Het project staat uiteraard toe vooruit te blikken naar komende hoofdtaken en de output van reeds voltooide hoofdtaken bij later gebleken wenselijkheid aan te passen.

De uitvoeringswijzen van de hoofdtaken worden in de eerste vijf paragrafen gedetailleerd, door de hierin te onderkennen activiteiten te omschrijven. In een zesde paragraaf wordt ingegaan op een gefaseerde uitvoering van de hoofdtaken.

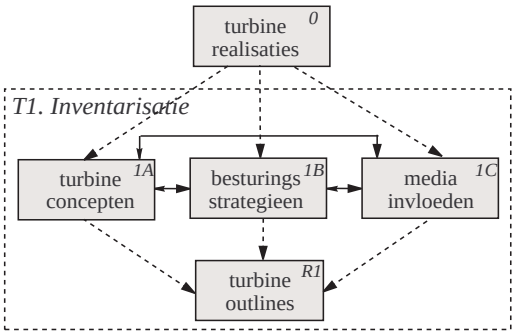
4.1 Inventarisatie (taak 1)

Tijdens de inventarisatie wordt de omgeving van de *processimulator* verkend. Het rekbare begrip 'omgeving' wordt hier gezien als het geheel aan factoren waaraan de eigenschappen van de *processimulator* vooral kwalitatief getoetst moeten kunnen worden. De voor de *processimulator* bepalende factoren, waaraan verbonden de vrijheidsgraden van de te realiseren zogenaamde *windturbine-outline-libraries*, worden vastgelegd door invulling te geven aan:

- *windturbine-concepten*;
- *besturingsstrategieën* voor de *windturbine-concepten*;
- *media*invloeden voor de combinaties van *windturbine-concepten* en *besturingsstrategieën*;

Er wordt een aantal gangbare *turbineconcepten* met bijbehorende *besturingsstrategieën* gedefinieerd, waarbinnen de realistische cases voor de *simulator* zullen worden opgezet. Dit zal aan het einde van de inventarisatie-taak leiden tot een aantal (case-gebonden) *turbine-outlines*. Dat wil zeggen dat de *componenten*, die in de *libraries* zullen worden opgenomen om *realisaties* van deze *turbine-outlines* later te kunnen modelleren, dan zijn benoemd.

Figuur 4.1 brengt de inventarisatie in beeld.



Figuur 4.1 Inventarisatie van *windturbine-outlines*

De terugwerking van vervolgactiviteiten is aangegeven met een kleine pijl. Deze wisselwerking is een gevolg van de onderlinge verbanden tussen de *windturbine-concepten*, *besturingsstrategieën* en *media*invloeden. De activiteiten worden toegelicht in de volgende drie subparagrafen.

Voor iedere uiteindelijk op te stellen *outline-library*, geldt dat deze geschikt zal zijn voor de *simulatie* van *realisaties* uit één zekere *windturbine-outline*, inclusief de relevante *media*invloeden. Zodra er sprake is van een alternatief *turbine-concept* of een andere *besturingsstrategie*, is er dus ook sprake van een andere *library*[3]. Derhalve zal een *outline-library* vaak geïdentificeerd kunnen worden met een specifieke windturbine of windturbinefabrikant. Het beschermen van vertrouwelijke gegevens bij ontwikkeling en gebruik van de *processimulator* wordt dan gemakkelijk gerealiseerd.

Het analyseren van bestaande *besturingssystemen* voor huidige *windturbine-concepten* zal een belangrijke bijdrage moeten leveren om doeltreffende case *outlines* en daaraan verbonden *libraries* te kunnen opstellen; medewerking van windturbinefabrikanten is dus niet alleen gewenst maar zelfs noodzakelijk. Ten behoeve van de genericiteit van de *processimulator* zal er naast de geboden voorbeelden echter ook met een breder spectrum aan *windturbineconcepten* en *besturingsstrategieën* rekening worden gehouden.

4.1.1 Windturbine-concepten (activiteit 1A)

Selecteren van *windturbine-concepten* waarvoor een *outline-library* gerealiseerd zal gaan worden. Per *windturbine-concept* worden voor de *windturbinedelen* keuzes gemaakt voor een verdeling in *systemen* en *componenten*. Tevens worden de mogelijkheden verkend welke *componenten* of *systemen* in aanmerking komen, om door ‘hardware in the loop’ apparatuur te kunnen worden vervangen.

4.1.2 Besturingsstrategieën (activiteit 1B)

Uitgangspunten en kenmerken beschrijven van de bij de *windturbine-concepten* horende *besturingsstrategieën*, met inbegrip van regelstructuren en de beveiligingsvoorzieningen. Bij een bepaald *turbine-concept* kunnen meerdere *besturingsstrategieën* behoren, die niet noodzakelijkerwijs tot evenveel *outline-*

[3] een andere *besturingsstrategie* in de zin van opgenomen - en afgegeven signalen; niet in de zin van de voor de *simulator* onzichtbare interne verwerkingswijze

libraries hoeven te leiden; de *windturbine-outline* hoeft immers niet voor elke *besturingsstrategie* te verschillen.

4.1.3 Media invloeden (activiteit 1C)

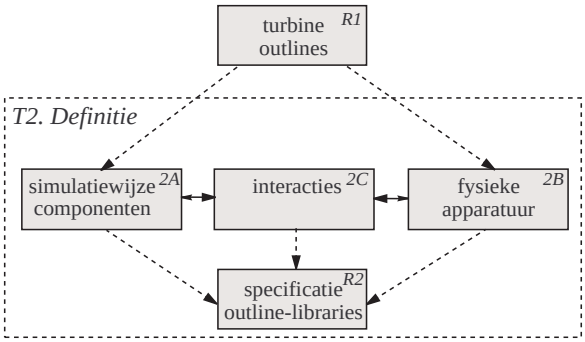
Vaststellen van de *media componenten*: in relatie tot de *windturbine-concepten* en bijbehorende *besturingsstrategieën* worden de relevante *media* eigenschappen en - fenomenen bepaald.

4.2 Definitie (taak 2)

Bij het doorlopen van de definitie-taak worden de specificaties voor de *outline-libraries* opgesteld. Hiervoor worden tijdens de definitie, aan de hand van de dan reeds geformeerde *windturbine-outlines*, vastgelegd:

- *simulatiewijzen* voor de geïdentificeerde *componenten*;
- uitvoeringsvormen van de potentiële 'hardware in the loop' apparatuur;
- interacties tussen *componenten*, *systemen* en eventuele fysieke apparatuur.

Figuur 4.2 brengt de definitie-taak in beeld.



Figuur 4.2 Definitie *turbine-libraries*: *componenten* worden gespecificeerd

Zoals de dubbele pijlen weergeven, zullen onderlinge verbanden wederom tot een iteratieve werkwijze leiden. De activiteiten 2A, B en C worden toegelicht in de volgende drie subparagrafen.

4.2.1 Simulatiewijze van componenten (activiteit 2A)

Het vastleggen van de *simulatiewijze* van een *component*, is het beschrijven van zijn gedrag en samenstelling tot het voor de *processimulator* vereiste detailleringsniveau. Gedrag betreft hier de 'veranderlijke' eigenschappen van *componenten* (de toestanden die *componenten* kunnen doorlopen), beschreven door fenomenen (tijdcontinu), events (tijddiscreet) en statussen (conditionele toestanden). Samenstelling betreft de 'constante' eigenschappen van componenten (dimensies).

Ook het feit dat de operator van de *processimulator* gedrag moet kunnen 'forceren' vereist hier aandacht. De *componenten* worden dusdanig opgebouwd dat storingen, herstelacties of superponaties bij *media* (windvlaag, golffront e.d.) kunnen worden doorgevoerd.

Aan het einde van de definitie-taak, zal voor alle geïdentificeerde *componenten* van alle *simulatie-items* de *simulatiewijze* zijn vastgelegd. Daarmee ligt dan

vast hoe de op te stellen *componentmodellen* zich ‘aan de grens’ kwalitatief zullen gedragen: wat er van een *componentmodel* door zijn omgeving wordt waargenomen met betrekking tot I/O en werkingsprincipe.

Ook ontstaat hier voor het eerst inzicht in de benodigde hardware behoeften voor de *processimulator*.

4.2.2 Uitvoeringsvormen van fysieke apparatuur (activiteit 2B)

Met de *simulatiewijze* van *componenten* als uitgangspunt, zal voor de potentieel als ‘hardware in the loop’ op te nemen apparatuur moeten worden vastgesteld hoe dit gecombineerd kan worden met de *processimulator*. Van alle potentieel op te nemen fysieke apparatuur dient het op de *processimulatie* van invloed zijnde gedrag te worden bepaald.

4.2.3 Interactie componenten/systemen en fysieke apparatuur (activiteit 2C)

Aan de hand van de eerder beschreven *simulatiewijzes* en *uitvoeringsvormen*, wordt de wisselwerking tussen *componenten*, *systemen* en voorziene fysieke apparatuur vastgesteld.

Hiermee ontstaat verderstrekkend inzicht in de benodigde hardware behoeften voor de *processimulator*.

4.3 Modelvorming (taak 3)

Tijdens de modelvorming wordt voor alle *componenten* een gedetailleerde *realisatie*gerichte specificatie opgesteld, op basis waarvan in de *processimulator* invulling gegeven gaat worden aan de *simulatiewijzen*. Dit legt vast hoe de *componenten* in de *processimulator* geïmplementeerd gaan worden: de *componentmodulen* zijn dan gespecificeerd. Dit geldt ook voor eventueel benodigde ‘interfacing’*componenten* om een specifieke interactie te realiseren of de interactie met de potentiële ‘hardware in the loop’ tot stand te kunnen brengen.

Deze taak is onderverdeeld in de twee activiteiten die nauw met elkaar samenhangen:

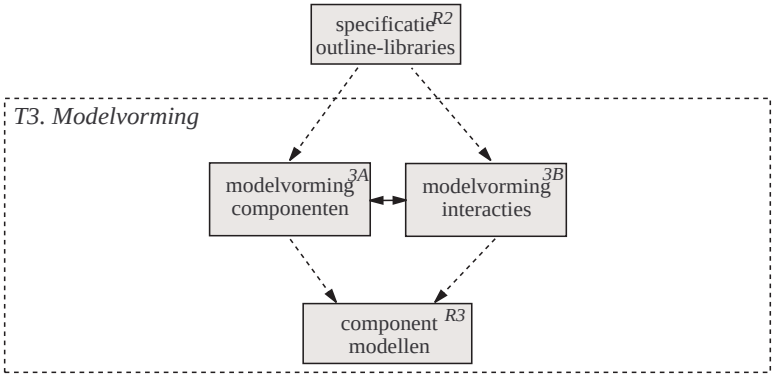
- modelvorming *componenten*;
- modelvorming interacties *componenten*, *systemen* en fysieke apparatuur.

Figuur 4.3 geeft de modelvormingstaak weer.

De dubbele pijl representeert weer de iteratieve werkwijze bij het doorlopen van de activiteiten, zoals de samenhang tussen de activiteiten dit zal vereisen. Aangezien de uit taak 3B voortvloeiende ‘interfacing’*componenten* voor de *processimulator* ook ‘gewoon’ *componenten* zijn, laat de output van deze taak zich volledig omschrijven als *componentmodellen*. De onderscheiden activiteiten worden toegelicht in de volgende 2 subparagrafen.

4.3.1 Modelvorming componenten (activiteit 3A)

Bij activiteit 3A worden de *componentmodellen* opgesteld volgens de Ward-Mellor definitie [3]. De *modellen* vormen een volledige specificatie voor de *compo-*



Figuur 4.3 *Modelvorming*

nentmodulen middels *transformatie-schema's*, waarbij zowel de *data-* als *control processen* tot in detail worden vastgelegd.

Een *dataproces* kan worden opgesplitst in een verder gedetailleerd *transformatie-schema* (met uitsluitend tijdcontinue *dataprocessen*), maar zal uiteindelijk worden gespecificeerd door een (regeltechnisch) blokschema, differentiaalvergelijkingen of algebraïsche functies. Ook een *controlproces* kan eventueel worden opgedeeld in een gedetailleerder *transformatie-schema* (maar dan met uitsluitend tijddiscrete *controlprocessen*). De uiteindelijke specificatie van een *controlproces* wordt gegeven door een *toestandsdiagram*. Met de aldus ontstane *modellen* worden de interne *component*samenstellingen en -gedragingen geheel vastgelegd, zowel kwalitatief (signalen en werkingsprincipe) als kwantitatief (limitering van signalen).

4.3.2 Modelvorming interacties (activiteit 3B)

Hier staan de interacties tussen de *componenten*, *systemen* en eventuele fysieke apparatuur centraal. Vanwege het hoge iteratieve gehalte wordt deze activiteit simultaan aan activiteit 3A uitgevoerd.

In eerste instantie worden de *modellen* zodanig opgesteld, dat de geïdentificeerde interacties bij koppeling van de *componentmodellen* (wat leidt tot *systemen* en koppeling van *systemen*) zullen worden gerealiseerd. Het is op voorhand niet uit te sluiten dat het wenselijk is een *systeem*grens zodanig te leggen dat een aanvullende 'interfacing'*component* nodig blijkt te zijn, waarvoor bij deze activiteit dan een *model* zal worden opgesteld.

Een ander gegeven dat zeker aanvullende 'interfacing'*componenten* zal vereisen, is het kunnen opnemen van 'hardware in the loop'. Het feit dat bepaalde turbine*componenten* of *systemen* niet met *simulator* interne *module(s)* hoeven worden nagebootst, maar ook via *simulator* externe apparatuur kan worden ingebracht, vereist 'interfacing' met de buitenwereld van de *simulator*. Deze 'interfacing' zal ook *modulen* vereisen, waarvoor tijdens deze activiteit *modellen* zullen worden geformeerd.

4.4 Implementatie (taak 4)

De taak implementatie voorziet in het converteren van opgestelde *modellen* in em modulen. Het betreft nu niet langer 'documentatie', maar bruikbare programma-

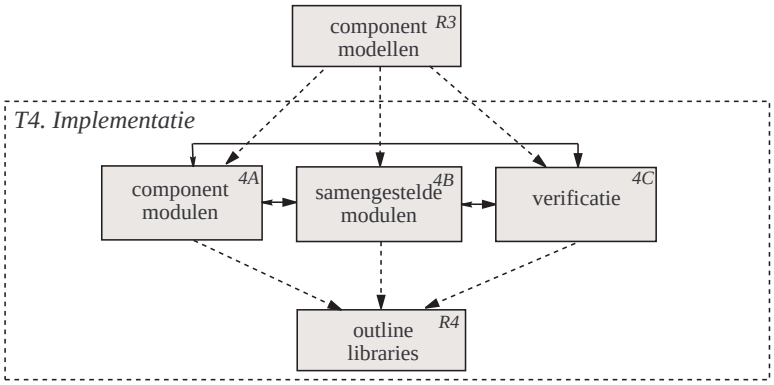
tuur die daadwerkelijk beoogd gedrag moet simuleren. In deze taak is er echter nog geen sprake van real-time simulatie, dit wordt tot stand gebracht in taak 5. De hoofdtaak implementatie bestaat uit 3 activiteiten:

- *componentmodulen* opstellen;
- *modulen* voor *systemen*, *simulatie-items* en proces samenstellen;
- module verificaties.

Een *componentmodule* is gedefinieerd als de code waarmee het gedrag van een *component* kan worden gesimuleerd en volgt direct uit het reeds opgestelde *componentmodel*. Na het opstellen van de *componentmodulen* en het hiermee samenstellen van de em modulen voor de hoger gelegen *modellen*, hebben de *outline libraries* dus gestalte gekregen.

Om een solide uitgangspunt te creeren voor hoofdtaak 5, worden alle geïmplementeerde *modulen* vanaf *component*- tot 'proces' niveau, getest op gespecificeerd gedrag.

Figuur 4.4 toont de samenhang tussen de activiteiten.



Figuur 4.4 *Implementatie van turbine libraries*

Ook hier is weer sprake van dubbele pijlen, om de wisselwerking tussen de activiteiten te benadrukken. Met name als gevolg van de verificatie-taak valt veel terugwerking te verwachten. De activiteiten worden toegelicht in de volgende drie subparagrafen.

4.4.1 Componentmodulen (activiteit 4A)

Het implementeren van de *modulen* zal plaatsvinden binnen het MATLAB-platform voor technisch-wetenschappelijke berekeningen en dynamische *simulatie* [2]. De tweedeling van de *model*beschrijving in *data*- en *controlprocessen* staat een doorzichtige implementatie toe, waarbij respectievelijk gebruik zal worden gemaakt van de toolboxen SIMULINK en STATEFLOW. Beide toolboxen beschikken over een zogenaamd 'Graphical User Interface' (GUI).

Ten behoeve van (tijddomein)*simulatie* met de SIMULINK-toolbox, kunnen de met blokschema's, differentiaalvergelijkingen en algebraïsche functies gespecificeerde *componentmodule*delen (voor de *dataprocessen*) rechtstreeks worden geïmplementeerd. Hiervoor kunnen blokschema's direct als SIMULINK-model worden ingevoerd. De differentiaalvergelijkingen en algebraïsche functies worden hiertoe geprogrammeerd als SIMULINK-functies (de zogenaamde S-functions), die dan vervolgens in de grafische representatie van een SIMULINK-model worden

opgenomen.

De door *state transistion diagrams* gespecificeerde *module*delen (voor de *control-processen*), kunnen met de STATEFLOW-toolbox direct in de gebruikte simulatie-omgeving worden geïmplementeerd. De STATEFLOW-modellen worden hierbij ten behoeve van tijddomein simulaties (grafisch) in SIMULINK-modellen opgenomen.

4.4.2 Modulen voor systemen, simulatie-items en proces (activiteit 4B)

Voor elke *turbine-outline* zullen de *componentmodulen* die zijn ingevoerd aan elkaar worden gekoppeld, opdat de modulen voor *systemen*, *simulatie-items* en het uiteindelijk 'te simuleren proces' (voor de desbetreffende *outline*) tot stand wordt gebracht. Dit gebeurt zodanig, dat de operator van de *processimulator* verschillende *turbine-realisaties* (inclusief omgeving) uit de van toepassing zijnde *outline-library* kan samenstellen: voor de *componenten* (en waarschijnlijk ook *systemen* en *simulatie-items*) waarvoor de *outline-library* meerdere alternatieven biedt, bepaalt de operator van de *processimulator* welke *componenten* (*systemen* en *simulatie-items*) in de simulatie worden opgenomen. De structuur van het gesimuleerde 'proces' (i.e. *outline* plus omgeving) kan niet door de operator worden beïnvloed.

4.4.3 Module verificaties (activiteit 4C)

Om te kunnen verifiëren of de geformeerde *modules* het in taak 3 gespecificeerde gedrag simuleren, zullen in- en outputs van de opgestelde modules worden nagebootst. Deze activiteit omvat het binnen MATLAB opzetten en doorlopen van de verificatie-tests op het niveau van *componenten*, *systemen*, *simulatie-items* en het 'gehele proces'. Het eventueel aanpassen van *modules*, betekent terugkeren naar de activiteiten 4A en 4B en mogelijk zelfs naar taak 3.

4.5 Real-time implementatie en case study (taak 5)

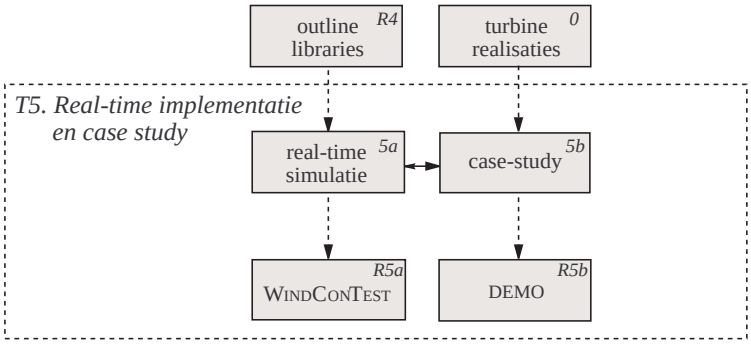
De taak real-time implementatie en case study behelst het operationeel maken van de real-time *processimulator* en het verifiëren van de werking ervan. De real-time implementatie van de *simulator* zal worden opgebouwd voor één *windturbine-realisatie*, die kan worden gesimuleerd met een *windturbine-outline* uit taak 4. Vervolgens zal hiermee een case study plaatsvinden: een Factory Acceptance Test (FAT) voor een *besturingssysteem*, dat in werkelijkheid op de (hier gesimuleerde) *turbine-realisatie* wordt aangesloten. Het gaat hierbij om fysieke *besturings*hardware (besturings-PLC's, microcomputers), die via een windturbinefabrikant zal moeten worden verkregen. De case study zal tevens dienen als basis voor de realisatie van een algemene DEMO, waarmee de *processimulator* ook voor een andere belangstellenden kan worden gedemonstreerd.

Taak 5 is opgesplitst in 2 activiteiten:

- Opstellen van de real-time *processimulator*;
- Case study.

Figuur 4.5 breng deze taken in beeld.

De dubbele pijlen staan weer voor de wisselwerking tussen de activiteiten. De volgende twee subparagrafen lichten de activiteiten toe.



Figuur 4.5 WINDCONTEST: *real-time implementatie en case-study*

4.5.1 Opstellen van de real-time processimulator (activiteit 5A)

De opstelling voor de real-time *processimulator* bestaat in de beoogde toepassing uit:

- een host-PC, waarop de programmatuur voor een te simuleren proces wordt ontwikkeld (binnen een MATLAB-omgeving);
- de target-PC, waarop deze programmatuur ten behoeve van de real-time *processimulatie* wordt 'gedownload';
- het *besturingssysteem*: hardware (bijvoorbeeld besturing-PLC('s)) met de daarop geïnstalleerde software, waarmee een *besturingsstrategie* wordt gerealiseerd;
- eventuele fysieke apparatuur die als 'hardware in the loop' wordt opgenomen.

Deze activiteit behelst het (hardwarematig) aansluiten van bovengenoemde apparatuur en het geheel (softwarematig) installeren en laten werken.

De eerste actie is het selecteren van een geschikte *windturbine-realisatie* voor de real-time implementatie, waarbij de voorkeur uiteraard uitgaat naar een *realisatie* die tijdens het inventariseren van de *outlines* reeds een belangrijke input opleverde (zie paragraaf 4.1). Voor de gekozen *realisatie* wordt het te simuleren proces op de host-PC samengesteld en gedimensioneerd, met de SIMULINK- en STATEFLOW-*modules* uit de van toepassing zijnde *outline-library*.

Vervolgens wordt de *simulator* geschikt gemaakt voor real-time simulatie, door de samenstelling van *modules* om te zetten in real-time C-code en dit op een procescomputer (de target-PC) te 'downloaden'. Door gebruik te maken van de REAL-TIME WORKSHOP-toolbox van MATLAB, zal dit geheel automatisch plaatsvinden. De REAL-TIME WORKSHOP-toolbox voorziet tevens in het kunnen opnemen van 'hardware in the loop' apparatuur, als onderdeel van het (verder gesimuleerde) proces. Middels dezelfde werkwijze wordt ook het *besturingssysteem* aan het met de target-PC gesimuleerde proces gekoppeld.

Voor het aansluiten van de fysieke 'hardware in the loop' apparatuur en de *besturing*, zullen op de target-PC commercieel verkrijgbare configureerbare I/O boards met bijbehorende drivers worden gebruikt. Merk op dat deze koppelingen tussen target-PC en fysieke apparatuur per aangesloten apparaat moeten worden geconfigureerd. De koppeling tussen host- en target-PC wordt daarentegen eenmalig opgezet en vormt daarna een vast onderdeel van de *simulator*opstelling.

Met de aldus ontstane opstelling kan het *besturingssysteem* in een reële omgeving worden beproefd. Hierbij wordt het real-time cyclisch doorlopen van de 'software' op de target-PC verzorgd door een real-time kernel (vooralsnog wordt hiervoor dSPACE voorzien, xPC TARGET is een mogelijk alternatief). Onderwijl kan

de operator interactief in het real-time gesimuleerde proces ingrijpen (‘gedrag forceren’) en het gedrag van proces, *besturing* en ‘hardware in the loop’ monitoren. Deze user-I/O zal via de host-PC verlopen.

4.5.2 Case study (activiteit 5B)

Tijdens activiteit 5B wordt de eigenlijke FAT uitgevoerd voor het bij taak 5A beschouwde *besturingssysteem*. Om dit vlot en realistisch te laten verlopen zal een testplan worden opgesteld dat vastlegt aan welke bedrijfsomstandigheden de *besturing* middels een *processimulatie* wordt onderworpen (test-cases met betrekking tot normaal turbinebedrijf, diverse faaltoestanden (van *componenten*), extreme *medium*condities en eventuele combinaties hiervan).

Als onderdeel van de case worden de prestaties van de *besturing* beoordeeld, door de testresultaten te interpreteren. De case study is hier natuurlijk voornamelijk bedoeld als demonstratie en beoordeling van de *processimulator*. Het analyseren van de mogelijkheden die de ontwikkelde *simulator* biedt met betrekking tot het uitvoeren van FAT’s voor *windturbine-besturingsystemen*, vormt dan ook een belangrijk onderdeel van deze activiteit.

Tenslotte wordt in deze activiteit nagestreefd om op basis van het beschouwde *besturingssysteem* een algemene DEMO samen te stellen, waarmee de mogelijkheden van de *processimulator* kunnen worden gedemonstreerd. Om vertrouwelijkheidsredenen (van de windturbinefabrikant die eigenaar is van het beschouwde *besturingssysteem*) of specifieke demonstratie behoeften, zou dit kunnen leiden tot gedeeltelijke her-configuratie en dimensionering van *modules* uit de *outline-library*. Voor het doorvoeren van dergelijke aanpassingen zal als gevolg van de gekozen ontwikkelingsstructuur slechts een beperkte inspanning nodig zijn.

4.6 Fasering

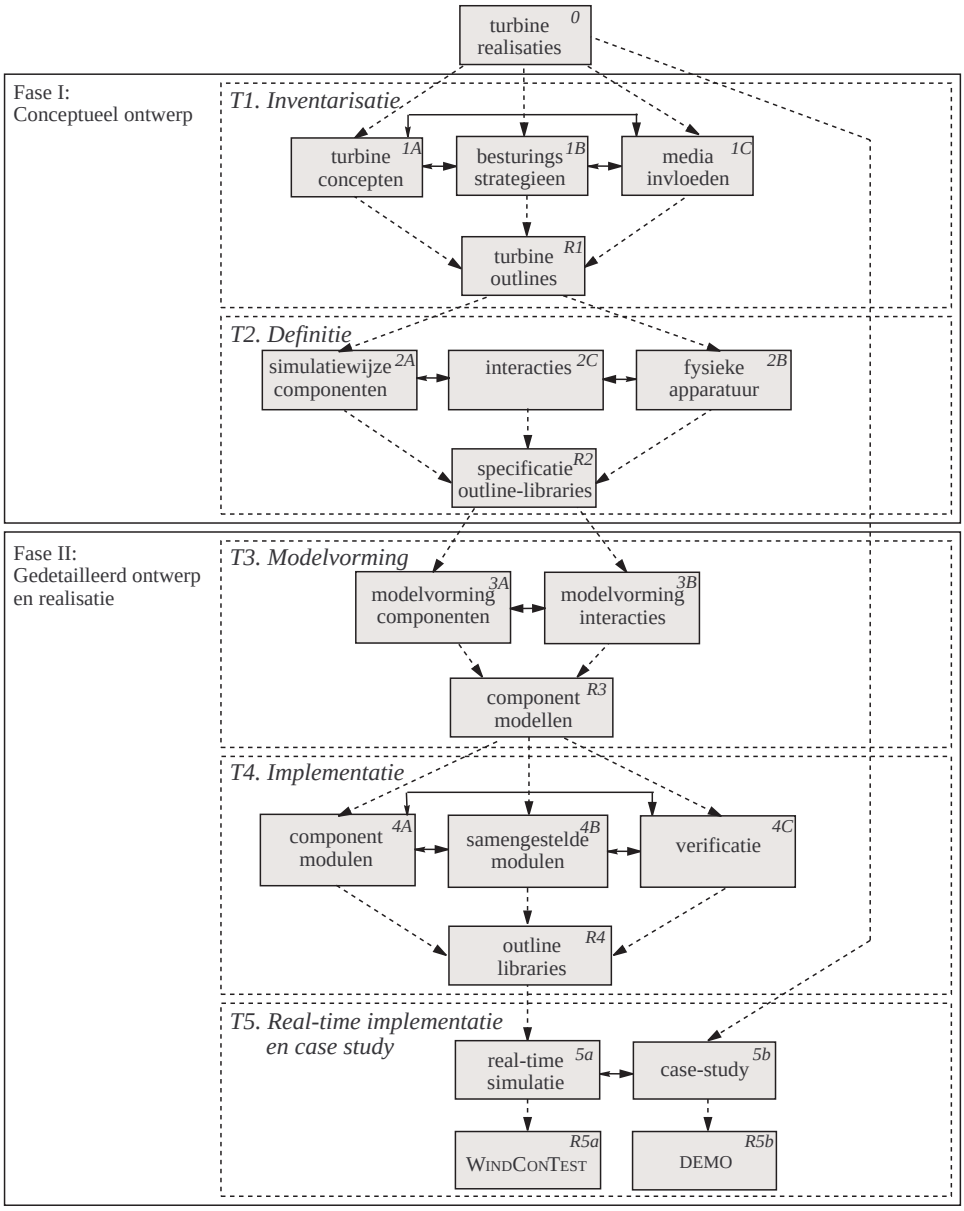
De activiteiten binnen de hoofdtaken zijn in zekere zin met elkaar verweven. Deze zijn daarom bij de uitvoering niet volledig te scheiden. Gecombineerde uitvoering van hoofdtaken 1 en 2 enerzijds en 3, 4 en 5 anderzijds lijkt zeer wel mogelijk. Op basis hiervan worden 2 fases onderscheiden:

1. Conceptueel ontwerp;
2. Gedetailleerd ontwerp en realisatie.

Figuur 4.6 brengt het ontwikkeltraject nog eens in beeld, met daarin onderscheid naar deze fases. Genoemde fases worden kort toegelicht in de volgende 2 subparagrafen.

4.6.1 Fase I: Conceptueel ontwerp

De hoofdtaken ‘inventarisatie’ en ‘definitie’ hebben voornamelijk betrekking op het vastleggen van de potentiële mogelijkheden van het *processimulatie*gereedschap. Dit is gerelateerd aan het gedrag van *componenten* en *media* zoals zich dat in de *simulator* zal moeten doen gelden. De uitvoering van deze twee hoofdtaken zal deels in wisselwerking moeten gebeuren. Wel is het zo dat ze vrijwel onafhankelijk uitgevoerd kunnen worden van de resterende hoofdtaken. Hierom worden deze taken gegroepeerd tot fase I: Conceptueel ontwerp.



Figuur 4.6 *Ontwikkeltraject processimulator windturbines*

4.6.2 Fase II: Gedetailleerd ontwerp en realisatie

De 3 hoofdtaken ‘modelvorming’, ‘implementatie’ en ‘real-time implementatie en case-study’ hebben betrekking op het operationeel maken van specificaties: respectievelijk door het uitvoeren van (papieren) analyses, het realiseren van *modulen* en het opzetten en toepassen van een real-time *processimulator*. De case study heeft hierin voornamelijk een demonstrerende en projectresultaat-controlerende taak. Omdat de voor *processimulatie* benodigde algoritmes bepaalde eisen stellen aan de *model*benadering, zullen ook deze hoofdtaken deels in wisselwerking worden uitgevoerd. De hoofdtaken worden gegroepeerd tot Fase II: Gedetailleerd ontwerp en realisatie.

5. CONCLUSIES

In deze probleemanalyse is een gestructureerde werkwijze afgeleid voor de ontwikkeling van het algemeen toepasbare real-time *processimulatie*gereedschap WINDCONTEST voor het testen van *windturbine-besturingssystemen* (zie paragraaf 4.6). Kernbegrippen hierin zijn modulariteit en fout- en uitzonderingscondities binnen *turbine-delen* en omgevingscondities van de te simuleren turbine. Bij de afleiding van dit ontwikkeltraject is gebruik gemaakt van een vereenvoudigde voorbeeld situatie, gebaseerd op een gespecificeerd *besturingssysteem*. De context en functionele opbouw van de *simulator* zijn hierbij gedefinieerd (zie figuur 2.1).

De voorgestelde werkwijze berust op zorgvuldig bepaalde begrippen en definities, en is gestoeld op de gestructureerde ontwikkeling van real-time systemen via de methode zoals beschreven door Ward & Mellor [3]. Het aanbrengen en wegnemen van storings- en uitzonderingstoestanden is hierin opgenomen via events en toestandsovergangen. Het technisch/wetenschappelijke MATLAB-computerplatform [2] voorziet middels 'Graphical User Interfaces' (GUI's) in de modulaire verwerking tot simulatie-code, van de specificaties die volgens deze methode zijn opgesteld. Het real-time toepasbaar maken van de aldus ontwikkelde programmatuur wordt hierbij door MATLAB ondersteund, alsmede de mogelijkheid om fysieke 'hardware in the loop' apparatuur in het gesimuleerde proces op te nemen.

REFERENTIES

[1] T.G. van Engelen, E.L. van der Hooft, and P. Schaak. Ontwerpgereedschappen voor de ontwikkeling van windturbines. Technical Report ECN-C—01-037, Energy research Centre of the Netherlands, Petten, The Netherlands. to appear.

[2] MathWorks. *MATLAB - The Language of Technical Computing; Using MATLAB Version 5*. The MathWorks, Inc., 24 Prime Park Way, Natick, MA 01760-1500, 1996.

[3] P.T. Ward and S.J. Mellor. *Structured Development for Real-Time Systems - Volume 1: Introduction & Tools*. Yourdan Press - Prentice Hall, Englewood Cliffs, NJ 07632, 1985.

BIJLAGE A. BESTURINGSSYSTEEM VOORBEELDTURBINE

In dit hoofdstuk wordt een besturingssysteem gespecificeerd voor de voorbeeldturbine waarvoor de simulator-ontwikkeling beschreven is in Hoofdstuk 3. In eerste instantie is dit besturingssysteem alleen gespecificeerd voor de gedachtenvorming bij simulatorontwikkeling. Dit stond toe om in verband met tijdgebrek bepaalde gedeelten oningevuld te laten. Tevens zijn in reeds wel gespecificeerde gedeelten bepaalde vereiste wijzigingen, die voort kwamen uit de simulator-ontwikkeling, nog niet aangebracht; dit echter is wel aangemerkt.

In §A.1 wordt de windturbine en de voorziene besturingsstrategie beschreven voor de gehanteerde turbine-layout volgens fig. 3.1.

Een mogelijk besturingsysteem wordt hier gespecificeerd via datatransformaties in §A.5 en toestandsovergangsdiagrammen in §A.6.

Deze specificaties worden bepaald op grond van een event/reponse model (§A.4) waarin de acties van het besturingsysteem in onderlinge samenhang beschreven zijn. De hierin vermelde acties zijn de *re*acties van het besturingssyteem op externe *events*, beschreven in §A.3). Het besturingssysteem dient ook te reagereren op het moment dat een schakelaar van de gesloten naar de open toestand overgaat of vice versa. Een gewijzigde schakelaarstand is daarom ook een extern event. In §A.2 worden dit soort events gedefinieerd voor de hoofd- en bypassschakelaar en ook voor de softstarter; tevens worden de benodigde processen voor deze event-gebaseerde conditionering en monitoring gespecificeerd. Hieruit wordt het context diagram voor het besturingssysteem afgeleid met event-gebaseerde besturing van schakelaars en softstarter.

Nog door te voeren wijzigingen (log-datum 5 november 2001)
Het softstartergedrag is gewijzigd; zie hoofdstuk 3. Het onderstaande besturingssysteem is hier nog niet op aangepast. Dit heeft consequenties voor figuren A.3, A.6 en A.7 en bijgaande text. De wijziging betreft het uit bedrijf gaan:

- De beginfase van netontkoppeling betrof voorheen het moment waarop de softstarter spanning lager werd dan de netspanning; nu is de beginfase van netontkoppeling de gehele spanningsafbouw-fase tot de ondergrens (van 200V?!) inclusief de initiatie daarvan.
- De eindfase van netonkoppeling betrof voorheen zowel de eigenlijk spanningsafbouw tot de ondergrens en het verbreken van de verbinding met de generator; nu is de eindfase alleen nog het verbreken van de verbinding.

Eea heeft tot de volgende naamswijzigingen tot gevolg in de events die de rand van het besturingssysteem genereert op grond van waarde-combinaties voor signalen *vrijgave*, *softstarter bedrijf* en *koppeling volledig*:

- *softstarter in bedrijf* wordt nu *softstarter spanningsopbouw*
- *softstarter wordt ontkoppeld* wordt nu *softstarter spanningsafbouw*
- *softstarter netontkoppeld* wordt nu *softstarter uit bedrijf*
- *softstarter initiele ontkoppeling storing* wordt nu *softstarter spanningsafbouw storing*
- *softstarter netontkoppeling storing* wordt nu *softstarter uit bedrijf storing*

- *softstarter in bedrijf storing* wordt nu *softstarter initialisatie storing*
- *softstarter netkoppeling storing* wordt nu *softstarter spanningsopbouw storing*

Daarnaast wordt het event *sofstarter netkoppeling storing* opnieuw gedefinieerd: het vindt plaats in geval van dat *softstarter initialisatie storing* of *softstarter spanningsopbouw storing* plaastvindt.

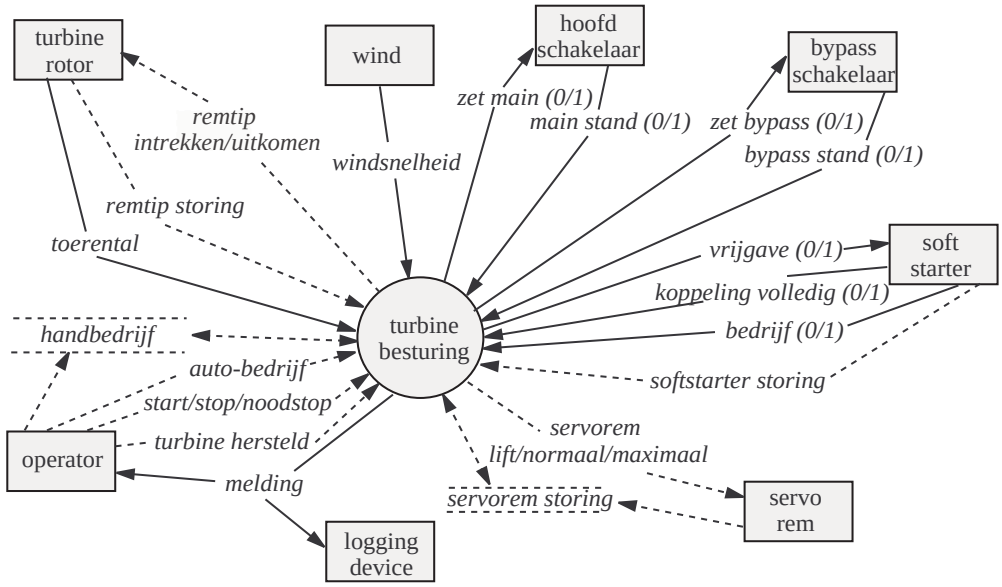
Ook wordt het event *sofstarter netontkoppeling storing* opnieuw gedefinieerd: dit vindt plaats in geval van *softstarter spanningsafbouw storing* of *softstarter uit bedrijf* plaastvindt.

Het event *softstarter netgekoppeld* blijft ongewijzigd.

A.1 Turbinebeschrijving voor ontwerp besturingssysteem

A.1.1 Turbine layout en context voor besturing

De (fictieve) turbine volgens fig. 3.1 heeft geen kruisysteem terwijl de toren volledig star verondersteld mag worden. De drie-bladige rotor heeft een straal van 25 m en neemt 830 kW op bij een windsnelheid van 12 m/s en een toerental van 25 rpm; hierbij wordt 750 kWe aan het net geleverd. Figuur A.1 geeft het context diagram ten behoeve van het besturingssysteemontwerp.



Figuur A.1 *Context diagram besturingssysteem bij voorbeeldturbine*

Hierin vormen de vierhoekige *terminators* de ‘context’ in de zin van de relevante signaalgevers en -ontvangers voor het besturinssysteem; hierin zijn de turbine-componenten te onderkennen.

Aerodynamische rem via bladtippen met locale besturingseenheid
De rotorbladen zijn voorzien van remtippen die automatisch uitkomen bij 28 rpm; daarnaast kunnen deze vanuit de besturing geactiveerd worden via het signaal *remtip uitkomen*, waarbij de opgaande flank voor signaaldetectie gebruikt wordt (pulse-gebaseerde sturing). Pas bij stilstand kunnen de remtippen weer ingetrokken worden en wel alleen vanuit de besturing via signaal-pulse *remtip intrekken*.

Iedere remtip zelf geeft het signaal *remtip storing* indien de 'reset' *remtip intrekken* niet geleid heeft tot het gewenste effect; dit gebeurt ook in de vorm van een signaal-pulse. Indien het signaal *remtip uitkomen* niet doorkomt zullen de tippen automatisch uitklappen bij overschrijding van 28 rpm.

Hydraulische servorem met locale besturingseenheid
De rotoras is voorzien van een hydraulische servorem. Het lichten van de rem gebeurt op signaal-pulse *servorem lift*. Onder normale condities tijdens shutdown realiseert deze rem, op signaal-pulse *servorem normaal*, een constante deceleratie totdat de rotor nagenoeg stilstaat; daarna wordt via de remklauwen een constante, maximale, kracht op de remschijf uitgeoefend. Onder noodcondities wordt direct maximaal geremd via signaal-pulse *servorem maximaal*. De servorem zelf geeft signaal-pulse *servorem storing* terug aan de besturing in de storingstoestand, waarbij wordt uitgegaan van het ontbreken van enig remvermogen.

Asynchrone generator en Hoofdschakelaar
De turbine is uitgevoerd met een langzaamlopende 'direct drive' asynchrone generator (synchroon toerental 25 rpm). Een noodzakelijke voorwaarde voor netkoppeling is een gesloten *hoofdschakelaar ('main')* ; openen/sluiten gebeurt via de waarde 0/1 van ingangssignaal *zet main*. De hoofdschakelaar stand wordt teruggemeld via de waarde 0/1 (open/gesloten) van uitgangssignaal *main stand*. Tussen hoofdschakelaar en electriciteitsnet is een hoofdzekering geplaatst; deze zal bij te hoge stroom doorslaan. De hoofdschakelaar werkt met signaal-*niveau's* en niet met signaal-*pulsen* zoals de servorem en remtippen. Onder productie-condities is de generator direct aan het net gekoppeld waarbij de koppel/toerencurve volgens de relatie van Kloss geldig is (nominaal koppel bij 0.5% slip). Er is slechts één (synchroon) bedrijfstoerental.

Softstarter en Bypass
Een veel ruimere band rond het synchrone toerental bij netkoppeling wordt gerealiseerd via een softstarter: deze zorgt direct na activering voor een zeer aanzienlijke verlaging van het elektrisch koppelniveau als functie van de slip ten opzichte van directe netkoppeling; daarna neemt dit niveau geleidelijk toe tot dat wat geldig is bij directe netkoppeling. Bij afschakelen wordt de omgekeerde procedure uitgevoerd. Procedures voor softstarter:

- Het ingangssignaal *vrijgave 1* activeert de softstarter. Het uitgangssignaal *softstarter bedrijf 1* wordt teruggegeven aan het besturingssysteem in de beginfase van de netkoppeling en het uitgangssignaal *koppeling volledig 1* nadat de koppel/toerental-afhankelijkheid haar eindwaarde heeft bereikt (de nominale netkoppeling). Het is zaak om hierna directe netkoppeling tot stand te brengen door sluiting van de *bypass schakelaar* via signaal *zet bypass 1*; deze is voorzien van een standmelder.
- Bij afschakelen wordt eerst de bypass schakelaar geopend (*zet bypass 0*) waarna de softstarter vrijgave ingetrokken wordt (*vrijgave 0*); dit laatste resulteert in geleidelijke afbouw van de netkoppeling, hetgeen tot uiting komt in het uitgangssignaal *koppeling volledig 0* aan het begin en *bedrijf 0* bij volledige ontkoppeling.

De softstarter en bypass schakelaar werken net als de hoofdschakelaar met signaal-*niveau's*; alleen het melden van de interne storingstoestand van de softstarter gebeurt via een signaal-*pulse*, n.l. *softstarter storing*.

A.1.2 Requirements Besturing

Condities voor start-up zijn:

- gemiddelde windsnelheid over 10 minuten tussen 4 en 24 m/s bij automatisch bedrijf;
- operator commando start bij handmatig bedrijf.

Condities voor een gewone stop zijn:

- gemiddelde windsnelheid over 10 minuten boven 25 m/s;
- gemiddelde windsnelheid over 10 minuten kleiner dan 3.5 m/s;
- operator commando stop

Condities voor een noodprocedure (noodstop, of in ieder geval afhandeling van noodcondities) zijn:

- momentane windsnelheid boven 35 m/s;
- storingstoestand servorem;
- storingstoestand remtippen;
- storingstoestand softstarter;
- hoofdschakelaar kan niet gesloten worden binnen 1 s;
- softstarter niet binnen 1 s in beginfase netkoppeling;
- softstarter niet binnen 10 s naar volledige netkoppeling;
- bypass langs softstarter kan niet gesloten worden binnen 1 s;
- bypass langs softstarter kan niet geopend worden binnen 1 s;
- softstarter niet binnen 1 s naar beginfase netontkoppeling;
- softstarter niet binnen 10 s in volledige netontkoppeling;
- hoofdschakelaar kan niet geopend worden binnen 1 s;
- hoofdschakelaar is niet gesloten;
- operator commando noodstop.

Na een noodprocecure dient de turbine altijd in de defect-toestand terecht te komen behalve als de oorzaak in de windsnelheid gelegen is. In de defect-toestand moet de hoofdschakelaar geopend zijn, anders mag deze gesloten blijven in afwachting van een herstart.

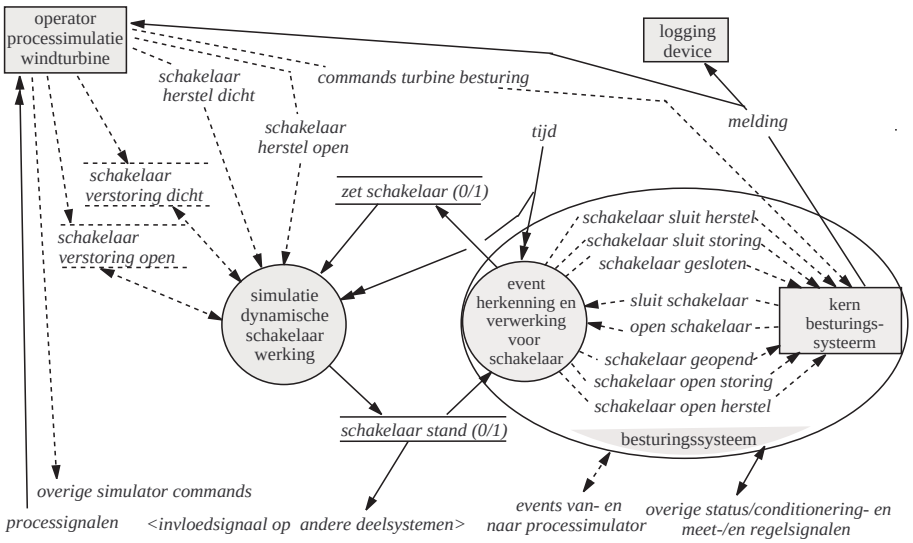
Vanuit de defect-toestand kan de turbine alleen weer operationeel worden via het operator commando *turbine hersteld*; deze komt dan in handmatig bedrijf terecht. De overgang van handmatige naar automatische bedrijfsvoering vergt het operatorcommando *auto-bedrijf*.

Het operator-commando *handbedrijf* bewerkstelligt dat de turbine vanuit de automatische start-up mode direct naar *handmatig bedrijfsvoering* gaat en dat dit gebeurt nadat de turbine gewoon gestopt is als deze op het moment van het commando in bedrijf is. Bij handbedrijf verloopt alleen een turbine-start via het besturingssysteem; alle overige handelingen met deelsystemen gebeuren dan buiten de besturing om.

A.2 Event-gebaseerde besturing van schakelaars en softstarter

Aangezien overgangen in de schakelaarstand ook externe events zijn worden in de rand van het besturingssysteem event-gebaseerde processen ingebouwd die enerzijds reageren op een veranderende schakelaar-stand (event-herkenning) en anderzijds een schakelaar-setting te weegbrengen op commando van het eigenlijke besturingssysteem (event-verwerking). Dit geldt ook voor de verschillende

toestanden die de softstarter doorloopt of kan doorlopen bij in bedrijf gaan, tijdens bedrijf en bij uit bedrijf gaan. Figuur A.2 geeft het gedeelte van het context diagram van de processimulatie met besturing dat toegespitst is op event-gebaseerde conditionering en statusmonitoring van een schakelaar.



Figuur A.2 *Event-gebaseerde schakelaarbesturing in rand van het besturingssysteem*

De processen voor schakelaar- en softstarterbesturing reageren op de volgende, voor zich sprekende, events vanuit (de kern van) het besturingssysteem:

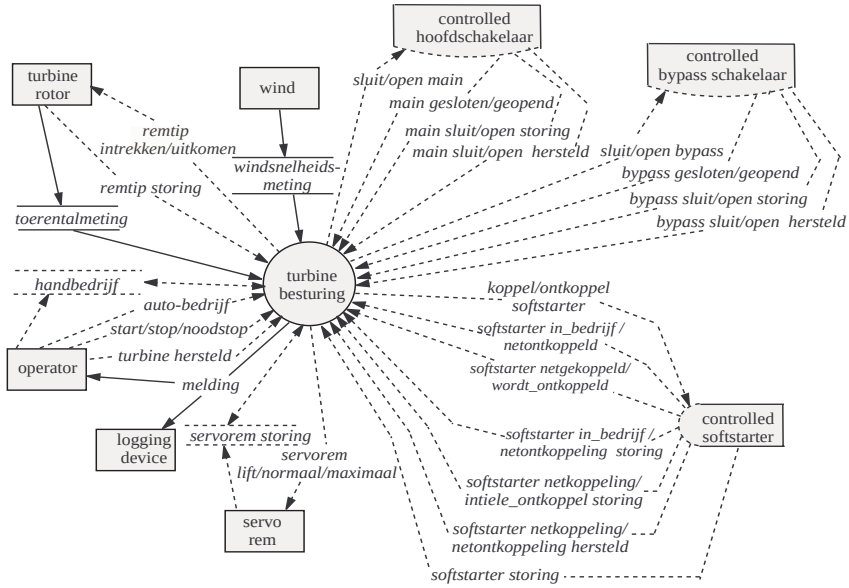
- *sluit hoofdschakelaar*
- *open hoofdschakelaar*
- *sluit bypassschakelaar*
- *open bypassschakelaar*
- *koppel softstarter*
- *ontkoppel softstarter*

Het verloop in de tijd van de ‘zet’- en ‘stand’-signalen voor de softstarter en de hoofd- en bypassschakelaar kan leiden tot de volgende events waarop het besturingssysteem dient te reageren.

- *hoofdschakelaar gesloten*: verbinding met elektriciteitsnet succesvol
- *hoofdschakelaar open*: ontkoppeling van elektriciteitsnet succesvol
- *bypass open*: bypass langs softstarter succesvol geopend
- *bypass gesloten*: bypass langs softstarter succesvol gesloten
- *softstarter netgekoppeld*: beginfase netkoppeling via softstarter en volledige netkoppeling op tijd.
- *softstarter netontkoppeld*: beginfase netontkoppeling en volledige netontkoppeling op tijd.
- *hoofdschakelaar open storing*: ontkoppeling van elektriciteitsnet niet tot stand gekomen
- *hoofdschakelaar sluit storing*: verbinding met elektriciteitsnet niet tot stand gekomen
- *bypass open storing*: bypass langs softstarter blijft gesloten
- *bypass sluit storing*: bypass langs softstarter blijft geopend
- *softstarter netkoppeling storing*: beginfase netkoppeling via softstarter dan wel volledige netkoppeling niet op tijd.
- *softstarter netontkoppeling storing*: beginfase netontkoppeling danwel volle-

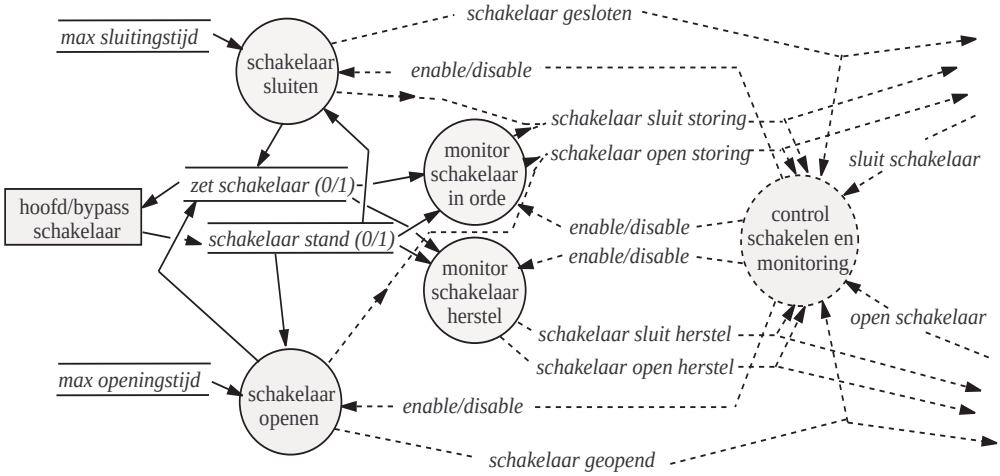
dige netontkoppeling niet op tijd.

De aldus gedefinieerde ‘event-verwerkende’ en ‘event-genererende’ schakelaars en softstarter worden gerealiseerd via programmatuur in de rand van het besturingssysteem. Aldus communiceert het eigenlijke besturingssysteem volledig via events met de (‘controlled’) schakelaars en softstarter. Fig. A.3 geeft het hierop aangepaste context diagram.



Figuur A.3 *Context diagram besturingssysteem met ‘controlled schakelaars en softstarter’*

Het transformatie-schema voor event-herkenning en -verwerking ten behoeve van de besturing van de hoofd- en bypass schakelaar is gegeven in fig. A.4.



Figuur A.4 *Transformatieschema event-herkenning en -verwerking tbv hoofd- en bypass schakelaar*

De (tijd-discrete) *data-transformaties* in figuur A.4 (gesloten cirkels) worden elke regelcyclus uitgevoerd wanneer *enabled*. Wanneer *disabled* vindt geen ingangs/uitgangs-mapping plaats. Hieronder volgende de specificaties voor deze vier processen. Deze geven aan hoe enerzijds ‘enable-events’ vertaald worden naar ‘zet’-signalen en anderzijds ‘uitgangsevents’ gegenereerd worden uit ‘stand’-

signalen.

Proces **monitor schakelaar in orde**:

als *zet schakelaar == 1* en *schakelaar stand == 0* dan

issue event *schakelaar sluit storing*

als *zet schakelaar == 0* en *schakelaar stand == 1* dan

issue event *schakelaar open storing*

Proces **monitor schakelaar herstel**:

als *zet schakelaar == 1* en *schakelaar stand == 1* dan

issue event *schakelaar sluit herstel*

als *zet schakelaar == 0* en *schakelaar stand == 0* dan

issue event *schakelaar open herstel*

Proces **schakelaar sluiten**:

zet schakelaar = 1
timer sluiten := timer sluiten + regelcyclus tijd
als *timer sluiten* $\leq$ *max sluitingstijd* en *schakelaar stand == 1*

issue event *schakelaar gesloten*

anders

als *timer sluiten* $>$ *max sluitingstijd*

issue event *schakelaar sluit storing*

Proces **schakelaar openen**:

zet schakelaar = 0
timer openen := timer openen + regelcyclus tijd
als *timer openen* $\leq$ *max openingstijd* en *schakelaar stand == 0*

issue event *schakelaar geopend*

anders

als *timer sluiten* $>$ *max openingstijd*

issue event *schakelaar open storing*

Het toestandsdiagram in figuur A.5 specificeert volledig de *control-transformatie* uit A.7 voor de schakelaars (streeplijn cirkel).

Proces **control schakelen en monitoring**:

Het transformatie-schema voor event-herkenning en -verwerking ten behoeve van de besturing van de softstarter is gegeven in fig. A.6.

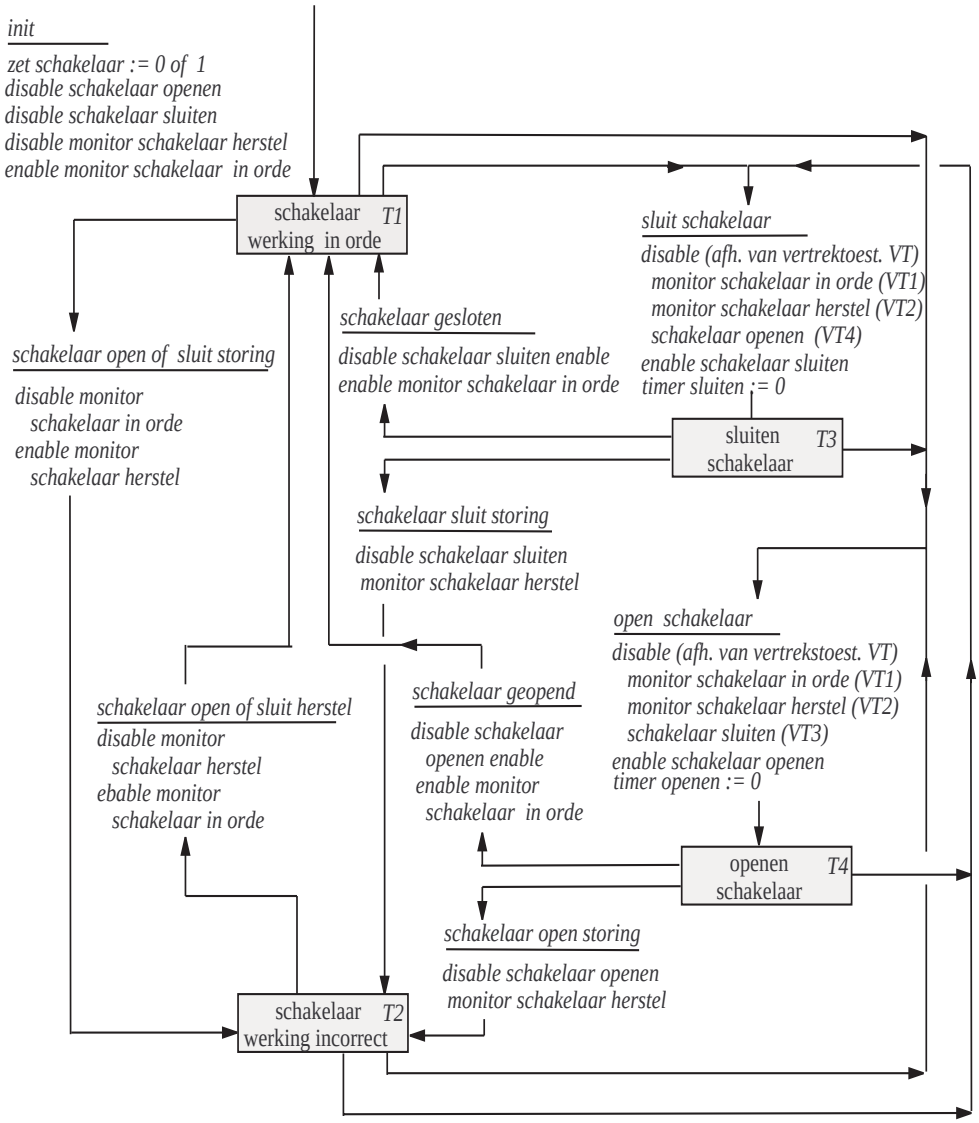
De *data-transformaties* in figuur A.6 (gesloten cirkels) worden elke regelcyclus uitgevoerd wanneer *enabled*. Wanneer *disabled* vindt geen ingangs/uitgangs-mapping plaats. Hieronder volgende specifcaties voor deze zes processen. Deze geven aan hoe enerzijds ‘enable-events’ vertaald worden naar het signaal *vrijgave* en anderzijds ‘uitgangsevents’ gegenereerd worden uit de signalen *bedrijf* en *koppeling volledig*.

Proces **monitor softstarter in orde**:

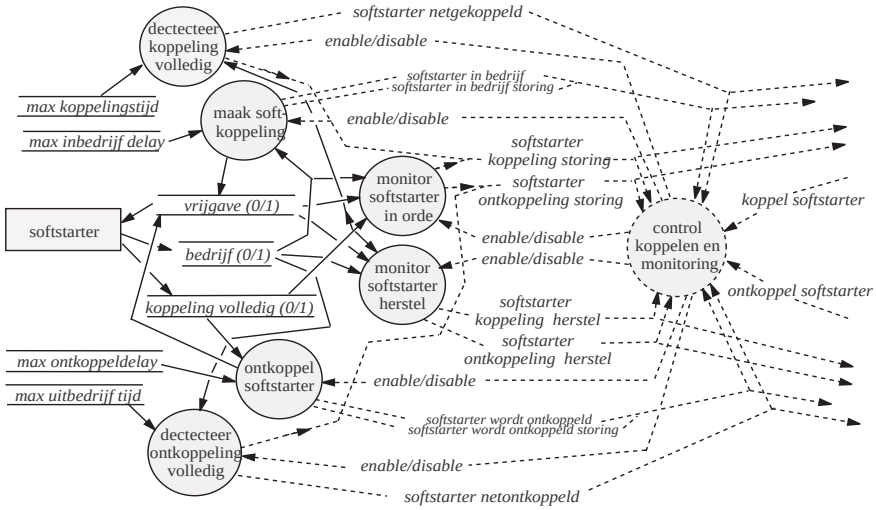
als *vrijgave == 1* en *volledige netkoppeling == 0* dan

issue event *softstarter netkoppeling storing*

als *vrijgave == 0* en $\{$*bedrijf ==1*$\}$ dan



Figuur A.5 *Toestandsdiagram event-controlled hoofd- en bypassschakelaar*



Figuur A.6 *Transformatie schema event-herkenning en -verwerking tbv softstarter*

issue event *softstarter netontkoppeling storing*

Proces **monitor softstarter herstel**:

als *vrijgave == 1* en *volledige netkoppeling == 1 en bedrijf == 1* dan

issue event *softstarter netkoppeling herstel*

als *vrijgave == 0* en {*bedrijf == 0* } dan

issue event *softstarter netontkoppeling herstel*

Proces **maak soft-koppeling**:

vrijgave = 1
timer koppeling begin := timer koppeling begin + regelcyclus tijd
als *timer koppeling begin* $\leq$ *max inbedrijf delay* en *bedrijf == 1*

issue event *softstarter in bedrijf*

anders

als *timer koppeling begin* $>$ *max inbedrijf delay*

issue event *softstarter in bedrijf storing*

Proces **detecteer koppelling volledig**:

vrijgave = 1
timer koppeling volledig := timer koppeling volledig + regelcyclus tijd
als *timer koppeling volledig* $\leq$ *max koppelingstijd* en *koppeling volledig == 1*

issue event *softstarter netgekoppeld*

anders

als *timer koppeling volledig* $>$ *max koppelingstijd*

issue event *softstarter netkoppeling storing*

Proces **ontkoppel softstarter**:

vrijgave = 0
timer ontkoppeling begin := timer ontkoppeling begin + regelcyclus tijd
als *timer ontkoppeling begin* $\leq$ *max ontkoppeldelay* en *koppeling volledig == 0*

issue event *softstarter wordt ontgekoppeld*

anders

als *timer ontkoppeling begin* $>$ *max ontkoppeldelay*

issue event *softstarter wordt ontkoppeld storing*

Proces **detecteer ontkoppelling volledig**:

vrijgave = 0
timer ontkoppeling volledig := timer ontkoppeling volledig + regelcyclus tijd
als *timer koppeling ontvolledig* $\leq$ *max uitbedrijf tijd* en *bedrijf == 0*

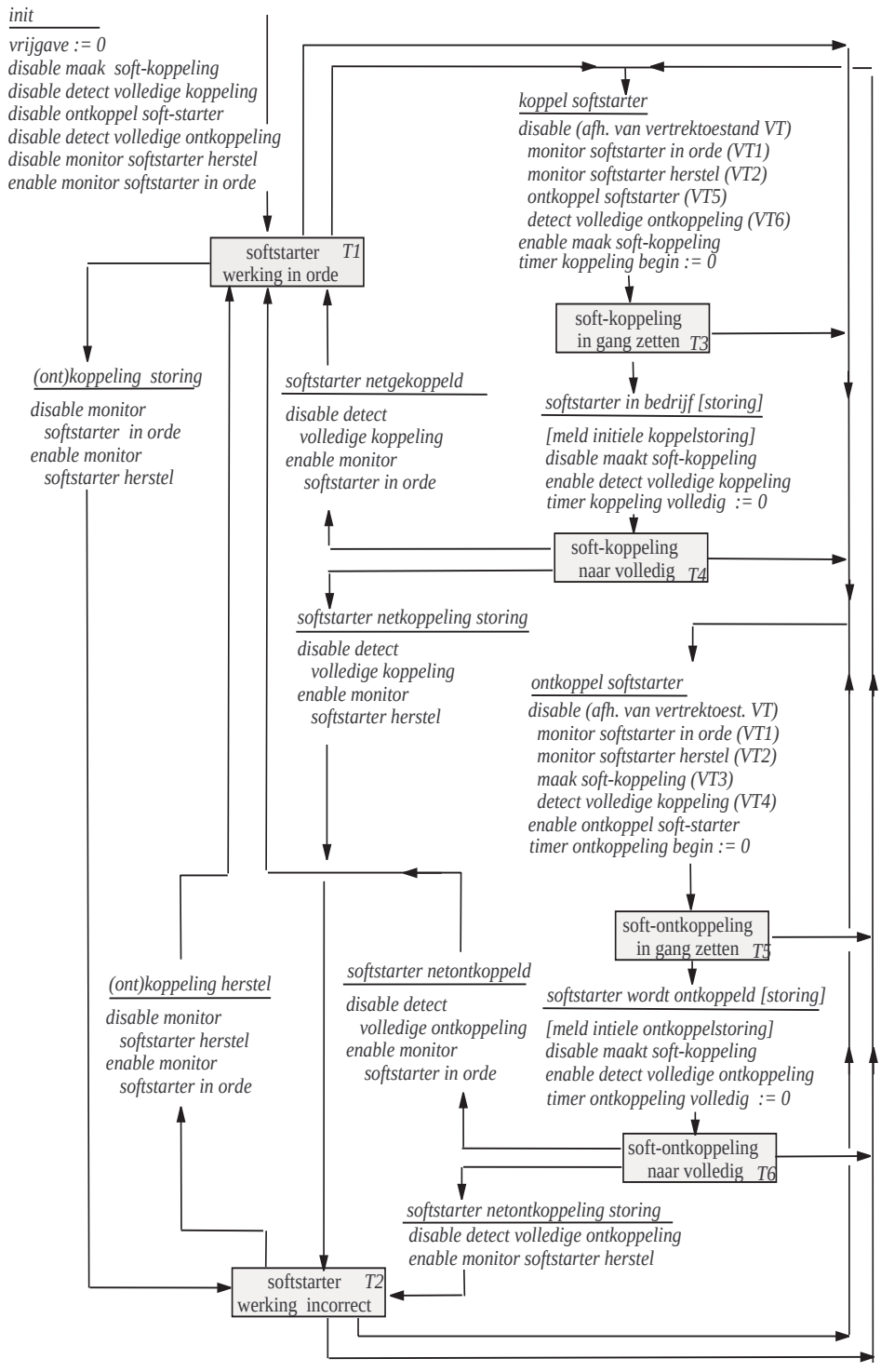
issue event *softstarter netontkoppeld*

anders

als *timer ontkoppeling volledig* $>$ *max uitbedrijf tijd*

issue event *softstarter netontkoppeling storing*

Het toestandsdiagram in figuur A.7 specificeert volledig de *control-transformatie* uit A.7 voor de softstarter (streeplijn cirkel).



Figuur A.7 *Toestandsdiagram event-controlled softstarter*

A.3 Event list

A.3.1 Events onder normale condities

windsnelheid start: gemiddelde windsnelheid over 10 minuten boven 4 m/s en

beneden 24 m/s bij turbine gereed voor start-up.

start: turbine-startup door operator

hoofdschakelaar gesloten: verbinding met elektriciteitsnet succesvol

softstarter netgekoppeld: beginfase netkoppeling via softstarter en volledige netkoppeling op tijd.

bypass open: bypass langs softstarter succesvol geopend

windsnelheid stop: gemiddelde windsnelheid over 10 minuten boven 25 m/s of onder 3.5 m/s

toerental stop: toerental overschrijdt 25.5 rpm gedurende 3 s

stop: turbine-shutdown volgens gewone procedure door operator

bypass gesloten: bypass langs softstarter succesvol gesloten

softstarter netontkoppeld: beginfase netontkoppeling en volledige netontkoppeling op tijd.

hoofdschakelaar open: ontkoppeling van elektriciteitsnet succesvol

automatisch bedrijf: overgang van handmatige naar automatsiche bedrijfsvoering door operator

handmatig bedrijf: overgang van automatische naar handmatige bedrijfsvoering door operator

A.3.2 Events onder uitzonderingscondities

remtip storing: aerorem reset (‘intrekken’) niet geslaagd

windsnelheid noodstop: momentane windsnelheid boven 35 m/s

toerental noodstop: toerental overschrijdt 27 rpm

noodstop: turbine-shutdown volgens noodprocedure door operator

softstarter netkoppeling storing: beginfase netkoppeling via softstarter dan wel volledige netkoppeling niet op tijd.

softstarter netontkoppeling storing: beginfase netontkoppeling danwel volledige netontkoppeling niet op tijd.

softstarter werking storing: storing softstarter in interne werking

servorem storing: servorem is niet in staat tot levering remvermogen

bypass open storing: bypass langs softstarter blijft gesloten

bypass sluit storing: bypass langs softstarter blijft geopend

hoofdschakelaar open storing: ontkoppeling van elektriciteitsnet niet tot stand gekomen

hoofdschakelaar sluit storing: verbinding met elektriciteitsnet niet tot stand gekomen

turbine hersteld: turbine na storing weer in orde gemeld door operator

A.4 Event/response model

Op grond van de requirements aan de besturing worden de volgende acties voorzien binnen het besturingssysteem:

Start-up procedure

- intrekken van remtippen
- netverbinding maken via hoofdschakelaar (‘main’)
- lichten servorem
- vrijgave softstarter bij synchroon toerental
- sluiten hoofdleiding parallel aan softstarter (‘bypass’)

Productie-klasse bedrijf

- <geen acties>

Shutdown-procedures

- stopprocedure (shutdown niveau 0)
- 4 noodprocedures (shutdown niveau’s 1 t/m 5)

Beschrijving van de shutdown niveau’s 0 t/m 5:

0 stopprocedure op grond van windsnelheid, toerental of operator:
- netontkoppeling door bypass openen en vrijgave softstarter intrekken
- servorem normaal
- turbine klaarzetten voor start-up

1 noodprocedure 1 op grond van windsnelheid [en storingscondities 0]
- netontkoppeling door bypass en softstarter
- servorem maximaal
- turbine klaarzetten voor start-up

2 noodprocedure 2 op grond van toerental, operator of bij netkoppeling storing of remtip-intrekken storing, of niet-fatale netontkoppeling storing [en storingscondities 0,1]:
- netontkoppeling via mogelijk zowel bypass en softstarter als hoodschakelaar
- servorem maximaal
- turbine buiten bedrijf stellen

3 noodprocedure 3 bij servorem storing [en storingscondities 0,1,2];
- netontkoppeling via mogelijk zowel bypass en softstarter als hoodschakelaar
- remtip laten uitklappen en servorem lift
- turbine buiten bedrijf stellen

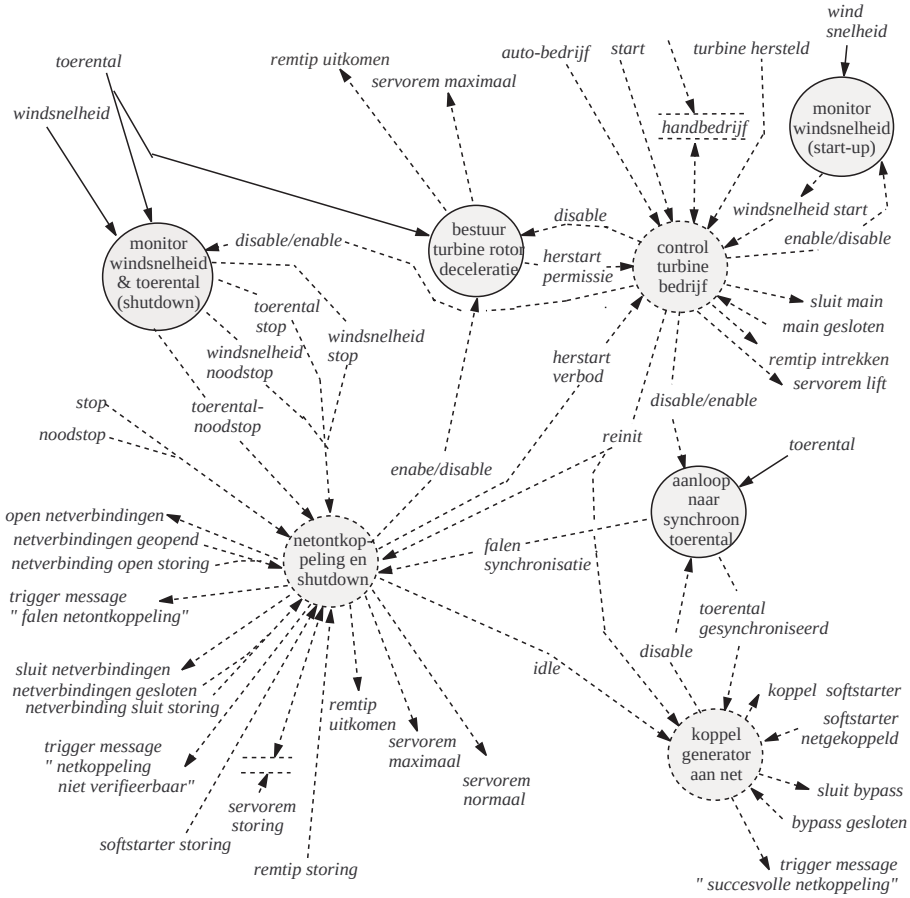
4 noodprocedure 4 bij fatale netontkoppeling storing [en storingscondities 0,1,2];
- sluiten main en bypass (of softstarternetkoppeling) voor doorslaan hoofdzekering
- remtip laten uitklappen en servorem maximaal
- turbine buiten bedrijf stellen
- melding gevaarsituatie door niet geslaagde netontkoppeling

5 noodprocedure 5 bij fatale netontkoppeling storing en storingsconditie 3 [en 0,1,2];
- sluiten main en bypass (of softstarternetkoppeling) voor doorslaan hoofdzekering
- remtip laten uitklappen
- turbine buiten bedrijf stellen

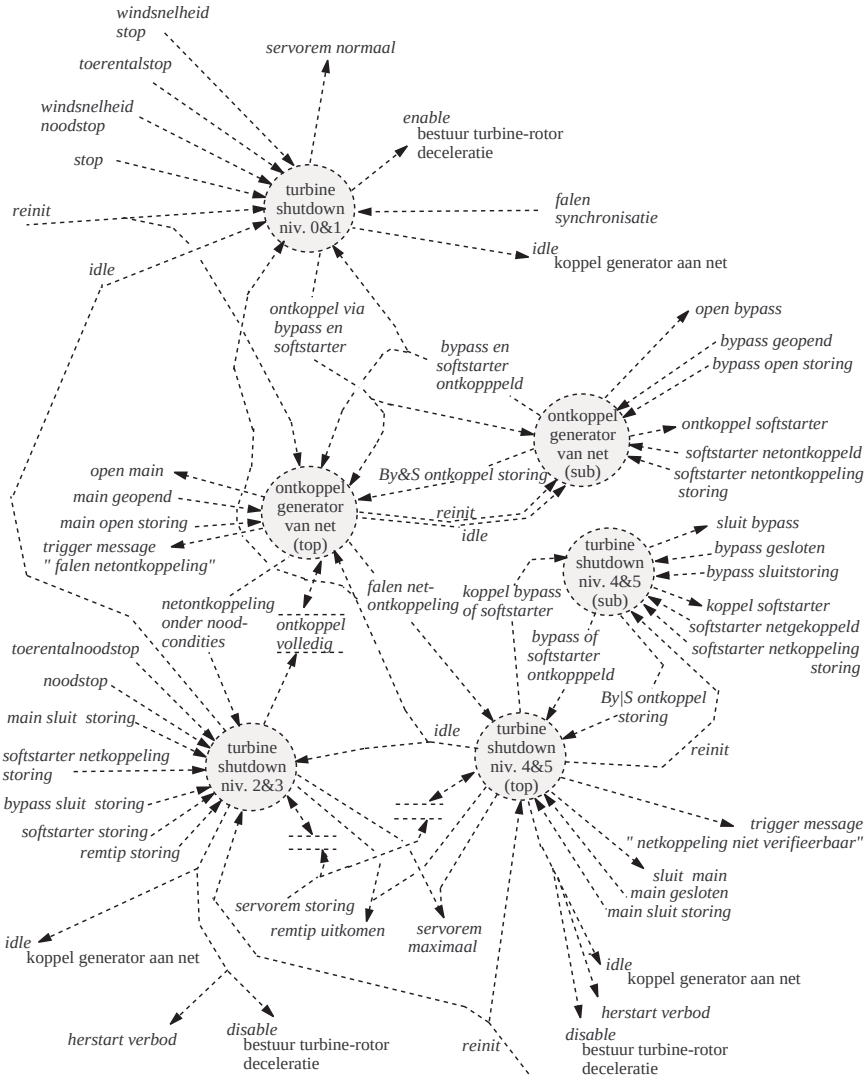
- melding gevaarsituatie door niet geslaagde netontkoppeling

Fig. A.8 geeft het zogeheten ‘preliminary transformation scheme’ waarin de complete layout van de besturing gegeven is, en wel zodanig dat de event/response-werking inzichtelijk gemaakt wordt; de tranformatie ‘netontkoppeling en shutdown’ is onderverdeeld in transformaties voor shutdown procedures op verschillende niveau’s en voor netontkoppeling.

Deze schema’s zijn afgeleid uit de besturingsrequirements en -acties waarbij de externe events gebruikt zijn volgens §A.3. Dit laatste impliceert dat de acties in de rand van het besturingssysteem voor realisatie van event-gebaseerde schakelaar- en softstarterwerking niet deze schema’s zijn opgenomen.



Figuur A.8 *Preliminary transformation scheme besturingssysteem (top)*



Figuur A.9 *Preliminary transformation scheme besturingssysteem (sub netonkoppeling en shutdown)*

A.5 Data-transformaties

Proces **Monitor windsnelheid (start-up)**

Proces **Aanloop naar synchroon toerental**

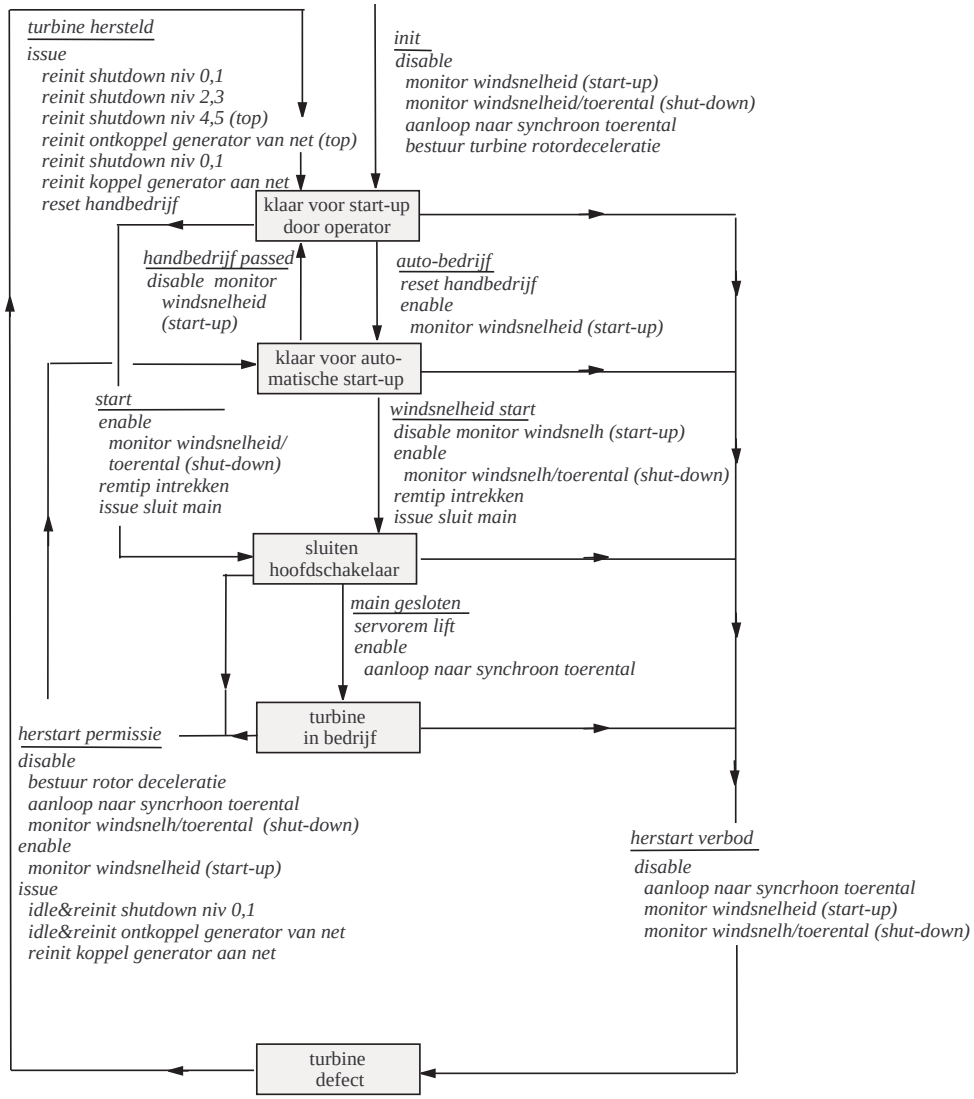
timer synchronisatie := timer synchronisatie + regelcyclus tijd
als *timer synchronisatie* $\leq$ *max synchronisatie-duur* en
$\{$*1.005*synchroon toerental* $\geq$ *toerental* $\geq$ *0.995*synchroon toerental* terwijl $|\text{ddt}($*toerental*$)| \leq$ *0.01*synchroon toerental per seconde*$\}$
issue event *toerental gesynchroniseerd*
anders
als *timer syncrhonisatie* $>$ *max synchronisatie-duur*
issue event *falen syncrhonisatie*

Proces **Monitor windsnelheid/toerental (shut-down)**

Proces **Bestuur turbine rotor deceleratie**

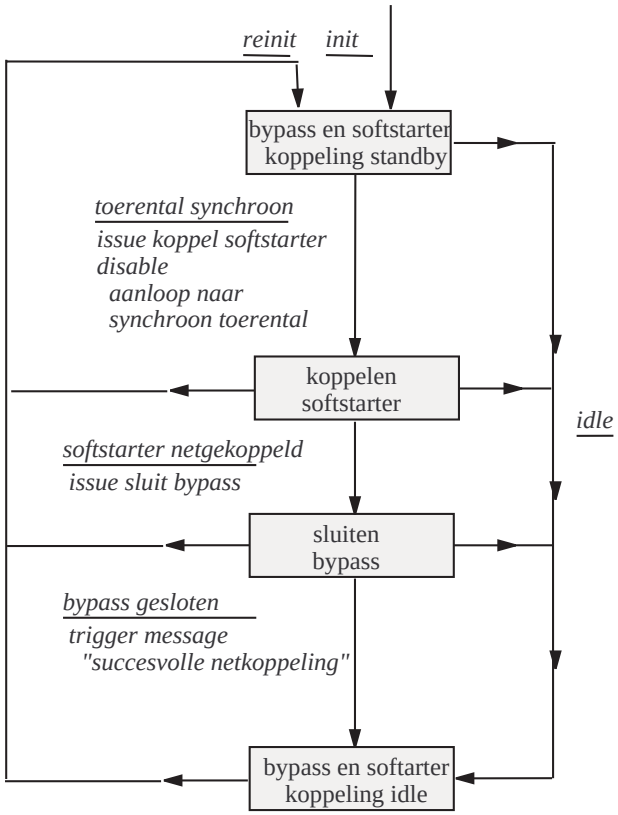
A.6 Control-transformaties (toestandsovergangen)

Proces **Control turbine bedrijf**



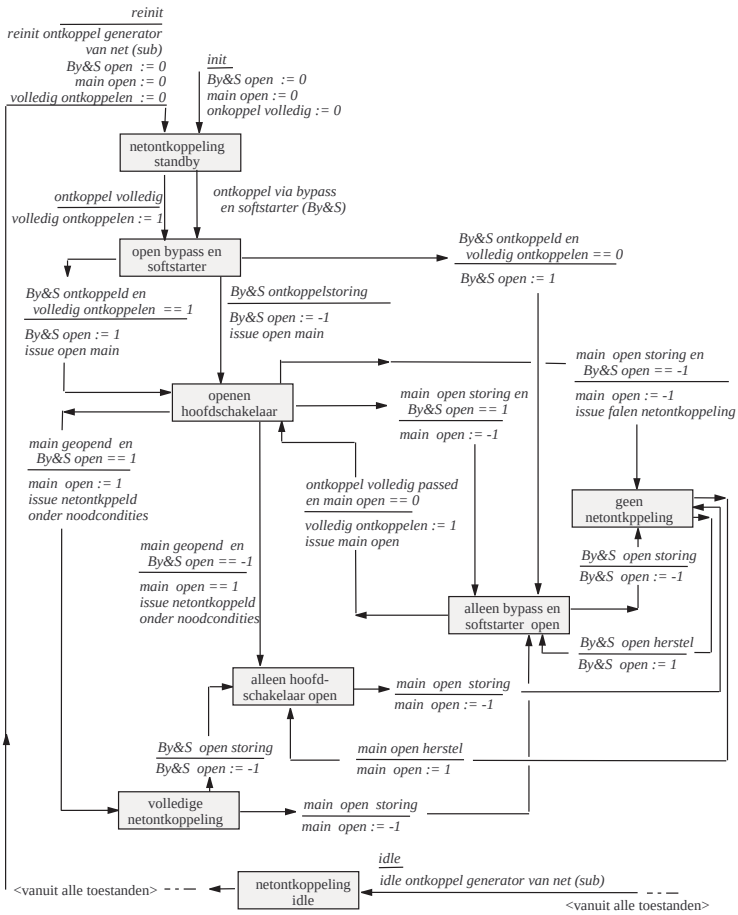
Figuur A.10 *Toestandsdiagram* **Control turbine bedrijf**

Proces **Koppel generator aan net**



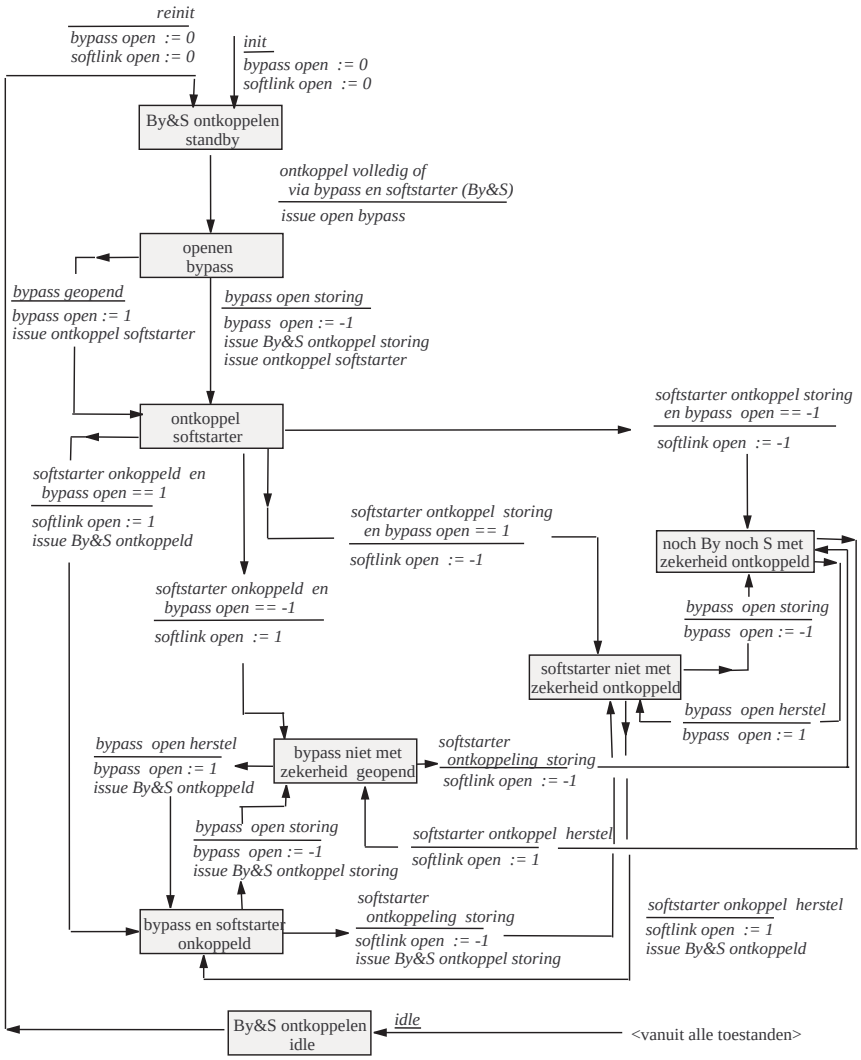
Figuur A.11 *Toestandsdiagram* **Koppel generator aan net**

Proces **Ontkoppel generator van net (top)**



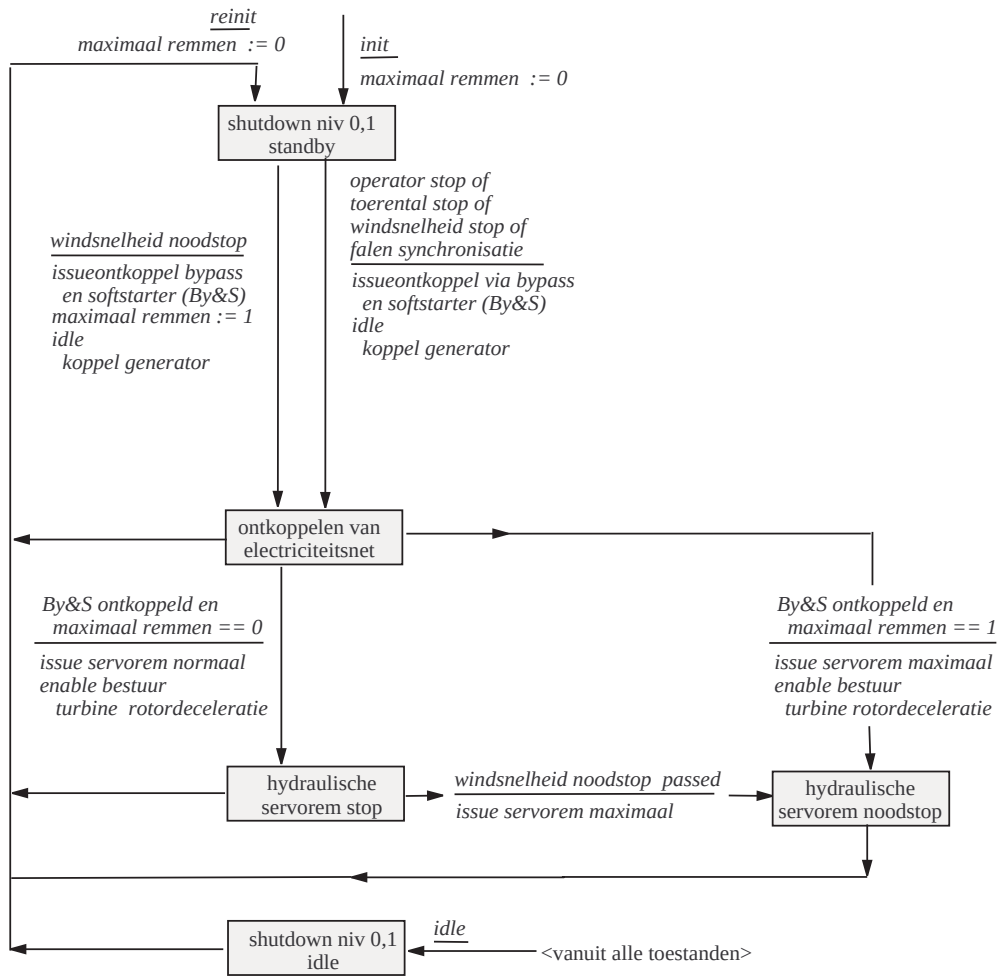
Figuur A.12 *Toestandsdiagram* **Ontkoppel generator van net (top)**

Proces **Ontkoppel generator van net (sub)**



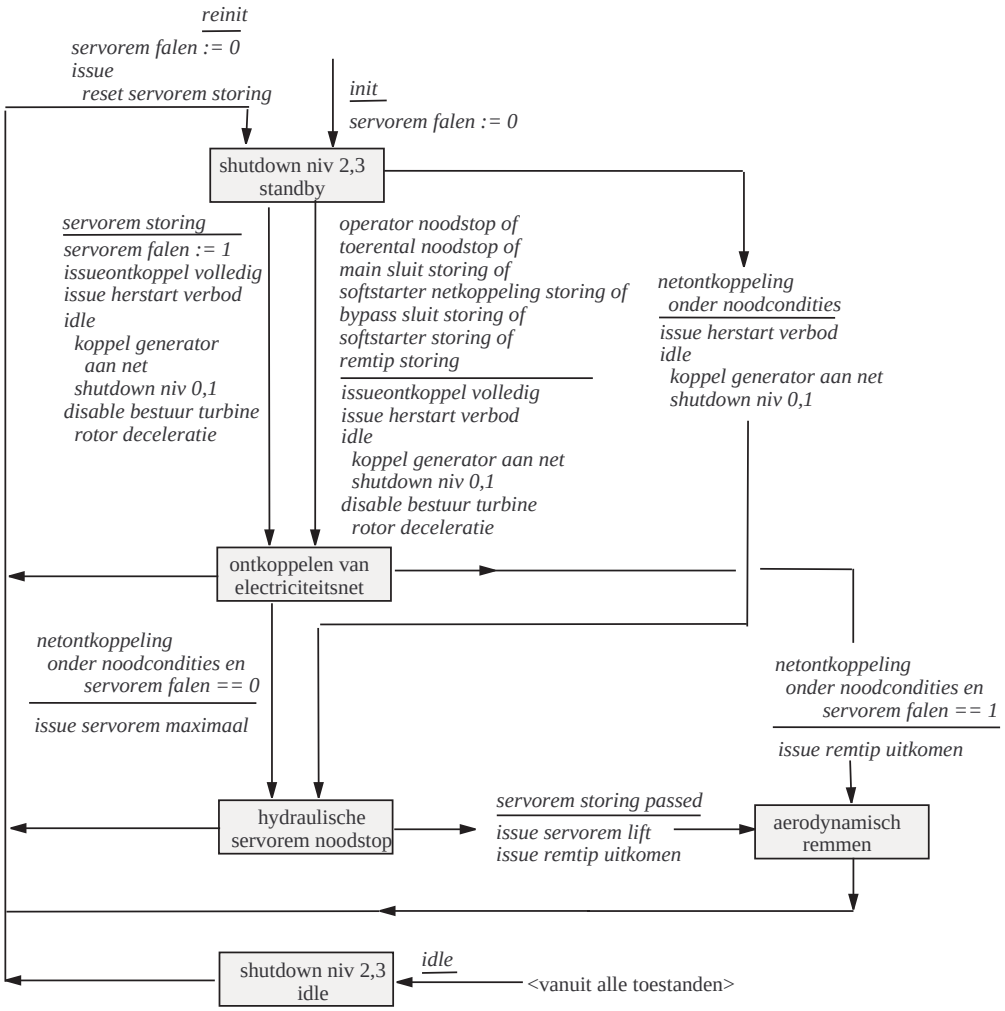
Figuur A.13 *Toestandsdiagram* **Ontkoppel generator van net (sub)**

Proces **Turbine shut-down niveau 0&1**



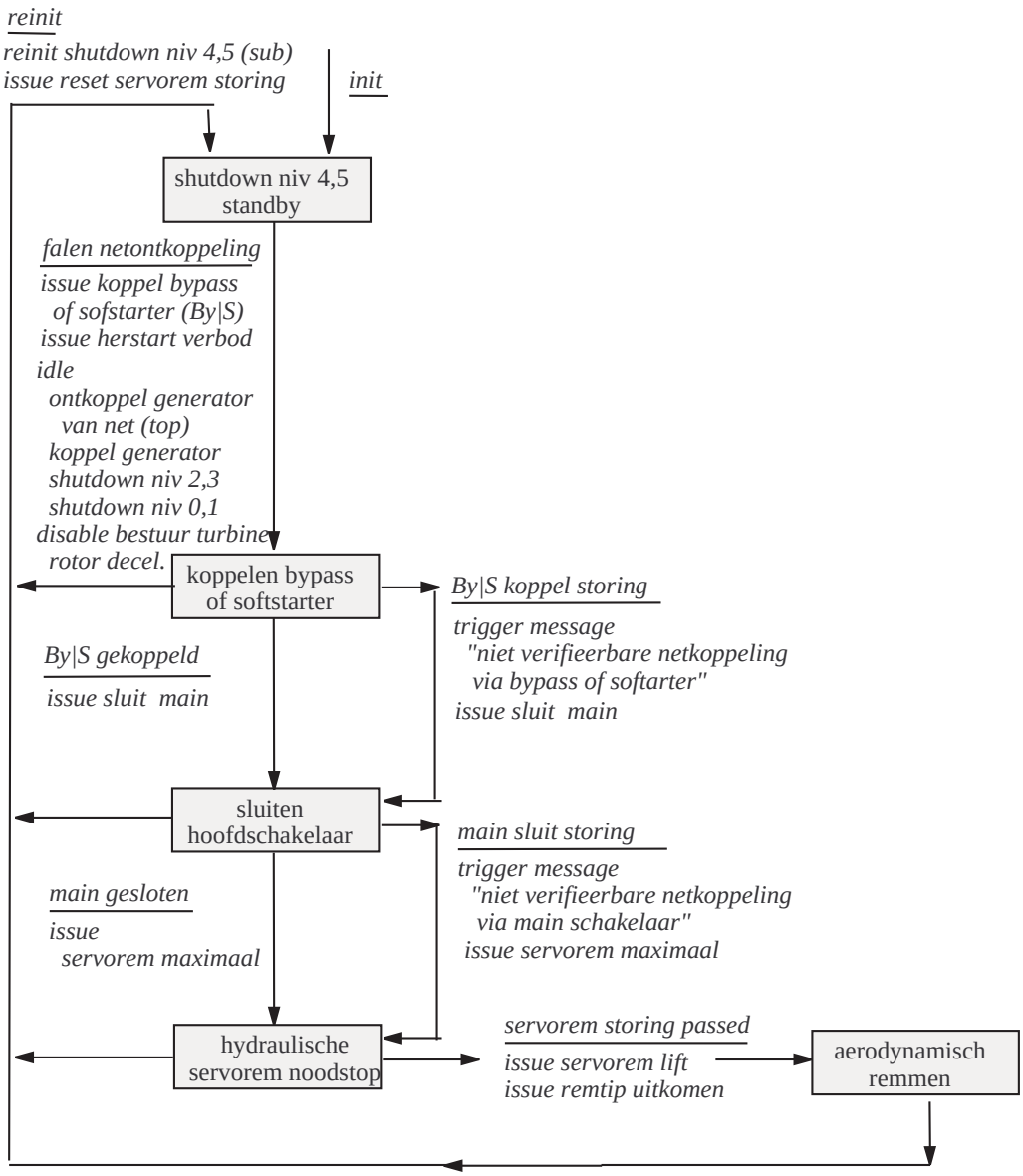
Figuur A.14 *Toestandsdiagram* **Turbine shut-down niveau 0&1**

Proces **Turbine shut-down niveau 2&3**



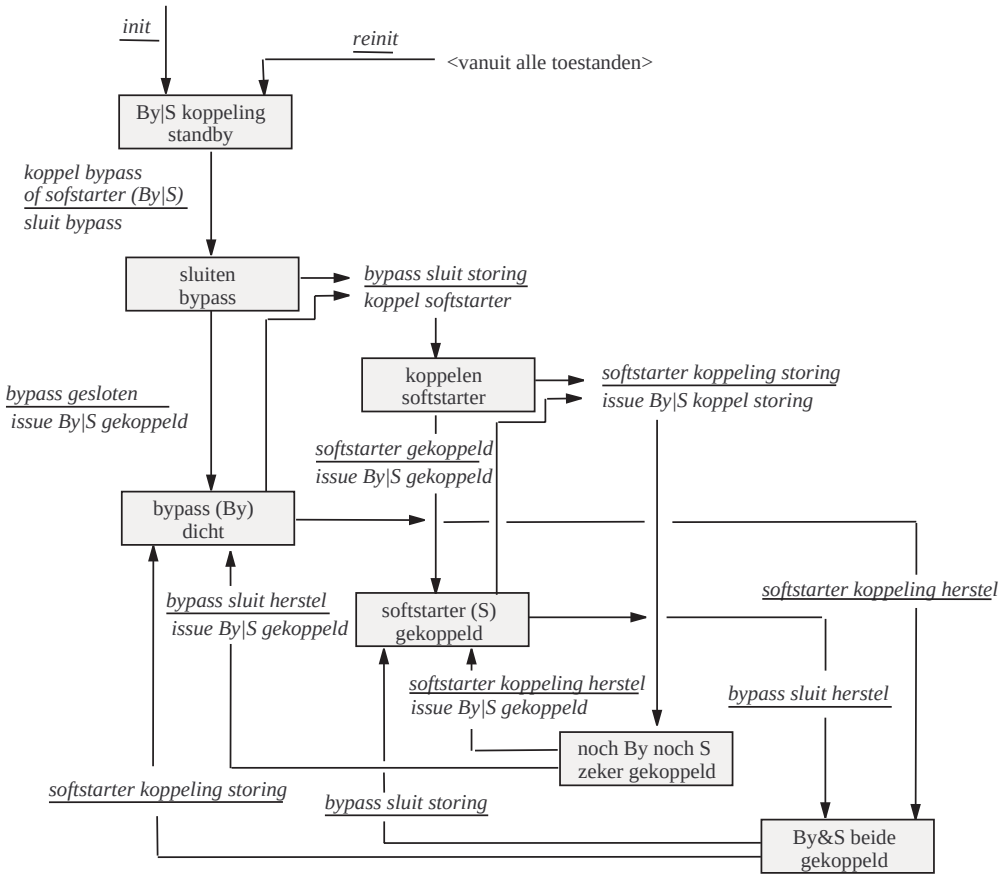
Figuur A.15 *Toestandsdiagram* **Turbine shut-down niveau 2&3**

Proces **Turbine shut-down niveau 4&5 (top)**



Figuur A.16 *Toestandsdiagram* **Turbine shut-down niveau 4&5 (top)**

Proces **Turbine shut-down niveau 4&5 (sub)**



Figuur A.17 *Toestandsdiagram* **Turbine shut-down niveau 4&5 (sub)**

	Datum: 5 november 2001	**Rapport No.:** ECN-C—01-112
Titel	Processimulator voor Windturbinebesturingen Volume I: Probleemanalyse	
Auteur(s)	T.G. van Engelen, P. Schaak	
Opdrachtgever(s)	NOVEM, Ministerie van EZ	
ECN projectnummer **Opdrachtgever ordernummer**	7.4097 224.721-0017	
Programma('s)	Duurzame energie, offhore windenergie (NOVEM), ARB Samenwerkingsfinanciering (EZ)	

Abstract

Door opschaling en minder toegankelijke offshore locaties is vooraf testen van *besturingssystemen* van windturbines steeds belangrijker om te kunnen voldoen aan hoge betrouwbaarheidseisen. Met *processimulaties* kan beoordeeld worden of het *besturingssysteem* het falen van *componenten* en (deel)*systemen* naar behoren afhandelt en of extreme bedrijfstoestanden goed worden doorstaan. In dit rapport wordt het ontwikkeltraject opgesteld voor het voorziene real-time *processimulatie*-gereedschap WINDCONTEST.

Het ontwikkeltraject is tot stand gekomen door voor een vereenvoudigde voorbeeldturbine een *simulator*-ontwikkeling daadwerkelijk uit te voeren en deze ontwikkeling tenslotte te formaliseren. Om een doeltreffende *processimulator* te ontwikkelen met de benodigde flexibiliteit ten aanzien van verschillende *turbine-realisaties*, is gestructureerd ontwerpen volgens de zogenaamde Ward-Mellor methodiek geschikt gebleken en biedt de real-time ontwikkelomgeving met MATLAB/dSpace ruime faciliteiten.

Verdere projectuitvoering zal leiden tot een geïnventariseerde set van *turbine-realisaties*, van waaruit ontwikkelde *componentmodellen* (grafisch) kunnen worden geselecteerd en geconfigureerd. De *processimulator* wordt gecreëerd door automatische conversie naar *component-modulen* voor real-time implementatie in de hiervoor opgestelde hardware voorzieningen. Het *windturbine-besturingssysteem* wordt hieraan fysiek gekoppeld.

Sleutelwoorden

processimulatie, windturbine, besturingsysteem, testen, offshore, factory acceptance test (FAT)

Authorisatie	**Naam**	**Handtekening**	**Datum**
Gecontroleerd			
Goedgekeurd			
Geauthoriseerd			