Information & Communication Technology
Anna van Buerenplein 1
2595 DA Den Haag
Postbus 96800
2509 JE Den Haag

www.tno.nl

T +31 88 866 00 00

Kansen en aandachtspunten bij het synthese-gebaseerd ontwikkelen van besturingssoftware voor infrastructurele objecten

Datum	15 december 2022
Auteur(s)	Dennis Hendriks
Versie	1.0
Aantal bijlagen	geen
Opdrachtgever	Rijkswaterstaat
Projectnaam	Onderzoek RWS mbt SBE advies
Projectnummer	060.53686

1 Inhoudsopgave

2 Management samenvatting

Rijkswaterstaat staat voor een enorme uitdaging wat betreft vervangen en renovatie (VenR) van objecten. Momenteel behelst dit voor elk object maatwerk, wat flexibiliteit biedt, maar ook bijzonder inefficiënt is. Voor het specificeren, ontwerpen, realiseren en testen van veilige besturingssoftware voor het object werkt het programma Industriële Automatisering Sourcing (IAS) daarom aan standaardisatie. Voor beweegbare bruggen en tunnels worden bouwblokken ontwikkeld, terwijl voor sluizen een lichtere vorm van standaardisatie wordt onderzocht: synthese-gebaseerd ontwikkelen (*Synthesis-Based Engineering*, SBE).

Binnen het MultiWaterWerk (MWW) project is samen met de Technische Universiteit Eindhoven (TU/e) de afgelopen jaren onderzocht in hoeverre deze werkwijze voor het ontwikkelen van besturingssoftware invulling kan geven aan de doelstellingen van Rijkswaterstaat, waaronder standaardisatie, uniformiteit, en een efficiëntere werkwijze, zonder in te boeten op betrouwbaarheid. Uit die academische samenwerking is gebleken dat SBE voor Rijkswaterstaat vele potentiële kansen biedt. Echter, het toepassen van deze nieuwe werkwijze levert ook vragen op. Rijkswaterstaat heeft tien vragen aan TNO en CGI voorgelegd. Dit TNO rapport geeft samen met het CGI rapport antwoord op die vragen.

Gesteld kan worden dat SBE in het gehele ontwikkelproces van besturingssoftware tal van garanties en andere (potentiële) voordelen biedt, zoals het vervaardigen van eenduidige, complete, consistente en up-to-date specificaties, en het sneller en goedkoper realiseren van besturingssoftware, die correct-door-constructie is, met een betere kwaliteit en veiligheid. De potentie hiervan is in de samenwerking van Rijkswaterstaat met de TU/e aangetoond, en SBE is toepasbaar op infrastructurele systemen. SBE heeft ook potentiële voordelen bij onderhoud en evolutie, en bij de gewenste introductie van productfamilies.

Na het aantonen dat SBE waarde heeft en bijdraagt aan de gestelde visie, overweegt Rijkswaterstaat nu óf en waar het wel of niet SBE wil gaan inzetten, om de geformuleerde visie te realiseren. Hiermee wordt een volgende fase ingegaan, waarbij ook andere aspecten een rol gaan spelen. Meer praktische aspecten, rondom de samenwerking met marktpartijen en de volwassenheid van SBE zijn minder aan bod gekomen in de academische samenwerking met de TU/e. Het ontbreken van commerciële ondersteuning, het beperkte ecosysteem, de beperkt aanwezige kennis en kunde bij zowel Rijkswaterstaat als marktpartijen voor het toepassen van SBE, evenals het verandermanagement en het eigenaarschap dat Rijkswaterstaat dient te pakken, zijn hierbij belangrijke aandachtspunten.

Hiertoe heeft Rijkswaterstaat een pad naar de toekomst geschetst, waarbij de aandachtspunten uit dit rapport en het rapport van CGI verder kunnen worden bekeken. TNO beschrijft in dit rapport hoe SBE verschilt van een traditionele manier van ontwikkelen, de diverse aspecten die daarbij komen kijken, en wat de meerwaarde en risico's zijn. Hiermee kan Rijkswaterstaat beter bepalen in hoeverre het SBE geschikt acht voor de gestelde doelen. Rijkswaterstaat zal uiteindelijk zelf moeten besluiten hoe zwaar het bepaalde aspecten, voordelen en risico's weegt, tot welke conclusies het daarmee komt, en dus of het SBE al dan niet breder in wil zetten. Idealiter komt Rijkswaterstaat daarbij tot een breed draakvlak voor deze conclusies en maakt het daarmee een duidelijke en weloverwogen keuze omtrent de verdere inzet van SBE.

In conclusie kan gesteld worden dat SBE met de Eclipse ESCET toolkit momenteel nog niet voldoende inzetbaar is om door Rijkswaterstaat en marktpartijen ingezet te worden voor het hele proces van het vervaardigen van de besturingssoftware. Diverse zaken moeten nog verder worden uitgezocht, moeten worden uitgewerkt, en moeten worden geregeld. Dit is echter logisch gezien de huidige fase waarin Rijkswaterstaat zich bevindt, waarbij nog niet definitief en volledig voor SBE is gekozen. Met inachtneming van de aanbevelingen uit dit TNO rapport en uit het CGI rapport, zien TNO en CGI geen fundamentele beperkingen wat betreft het toepassing van SBE met de Eclipse ESCET toolkit. Wel dienen de aanbevelingen dan gedegen geadresseerd te worden, zodat aan alle regels en richtlijnen wordt voldaan, en de vele potentiële voordelen van SBE daadwerkelijk tot hun recht kunnen komen.

3 Introductie

3.1 Achtergrond

Rijkswaterstaat is de uitvoeringsorganisatie van het ministerie van Infrastructuur en Waterstaat, en is verantwoordelijk voor onder andere de aanleg, renovatie en vervanging van tal van infrastructurele objecten in Nederland, waaronder sluizen, bruggen, tunnels en keringen. Deze objecten moeten te allen tijde juist en veilig functioneren.

Rijkswaterstaat staat hierbij voor een enorme uitdaging, waarbij in de komende 30 jaar in principe alle ruim vierhonderd objecten moeten worden gerenoveerd of vervangen, waaronder ook circa 170 sluizen [1, 2]. Rijkswaterstaat besteedt dit werk typisch uit aan marktpartijen (ingenieursbureaus, ICT-bedrijven). Echter, Rijkswaterstaat voorziet dat deze partijen met de huidige manier van werken niet voldoende capaciteit zullen hebben voor dit grote aantal renovaties. Zeker ook omdat naast Rijkswaterstaat ook gemeentes en provincies infrastructurele objecten hebben die eveneens in deze periode aan renovatie of vervanging toe zijn.

Momenteel wordt voor elk object maatwerk geleverd. Dat biedt flexibiliteit, maar is ook bijzonder inefficiënt en kost dus veel tijd en geld, omdat voor elk object telkens tal van zaken opnieuw moet worden ontworpen en gerealiseerd. Ook bij onderhoud, handmatige bediening via verschillende interfaces, en het uitrollen van centrale bediening is het brede spectrum aan gekozen oplossingen nadelig. Maatwerk wordt dan ook gezien als een belangrijke belemmering voor het schaalbaar uitvoeren van vervangings- en renovatieprojecten (VenR-projecten) [2]. Dit geldt zeker ook voor het specificeren, ontwerpen, realiseren en testen van de besturingssoftware van het object, onderdeel van de Industriële Automatisering (IA). Hoewel correcte besturingssoftware voor Rijkswaterstaat essentieel is voor de juiste en veilige werking van een object, kost het door maatwerk veel tijd en geld om alle veiligheidseisen in alle mogelijke gevallen te garanderen.

Binnen Rijkswaterstaat werkt het programma Industriële Automatisering Sourcing (IAS) daarom aan de standaardisatie van de IA voor objecten. Voor beweegbare bruggen en tunnels worden bouwblokken (configureerbare standaard hard- en software) ontwikkeld die in toekomstige projecten standaard moeten worden toegepast. Echter, IAS heeft in het plan van aanpak 2022 opgenomen dat voor sluizen een lichtere vorm van standaardisatie wordt onderzocht. Dit naar aanleiding van een advies van het Adviescollege ICT-toetsing, voorheen het Bureau ICT Toetsing (BIT), waarin expliciet wordt aanbevolen om alternatieven te bekijken [1]. Dit advies is overgenomen door de minister van infrastructuur en waterstaat [2].

Daarnaast wordt binnen Rijkswaterstaat in het MultiWaterWerk (MWW) project al jaren samengewerkt door de organisatieonderdelen Grote Projecten en Onderhoud (GPO), Centrale Informatievoorziening (CIV) en Programma's, Projecten en Onderhoud (PPO) om te komen tot gestandaardiseerde werkwijzen voor VenR-projecten op het gebied van sluizen. De standaardisatie moet leiden tot uniformer functionerende sluizen, en een voor zowel Rijkswaterstaat als voor marktpartijen efficiëntere werkwijze, zonder in te boeten op betrouwbaarheid van de objecten en de besturingssoftware van die objecten.

Specifiek wordt als lichtere vorm van standaardisatie voor het specificeren, ontwerpen, realiseren en testen van besturingssoftware gekeken naar *Synthesis-Based Engineering* (SBE), een Engelse term die in het Nederlands vertaald kan worden als synthese-gebaseerd ontwikkelen. MWW heeft samen met de Technische Universiteit Eindhoven (TU/e) de afgelopen jaren onderzocht in hoeverre deze werkwijze voor het ontwikkelen van besturingssoftware invulling kan geven aan de doelstellingen van Rijkswaterstaat.

3.2 Insteek van dit rapport

Uit de samenwerking met de TU/e is gebleken dat SBE voor Rijkswaterstaat vele potentiële kansen biedt, die het mogelijk maken besturingssoftware van hoge kwaliteit te maken tegen gelijke of zelfs lagere moeite

en kosten. Deze kansen komen in dit rapport uitgebreid aan bod. Echter, het toepassen van deze nieuwe werkwijze levert ook vragen op binnen de organisatie. Rijkswaterstaat heeft een aantal van deze vragen aan TNO, TU/e en CGI voorgelegd [3]. TNO belicht in dit rapport daarom ook de aandachtspunten rondom de introductie van SBE, om daarmee vanuit de expertise van TNO antwoord te geven op de gestelde vragen. De conclusies vanuit het CGI rapport [4] zijn voor een deel meegenomen in dit TNO rapport. Lezers van dit TNO rapport wordt aangeraden ook het CGI rapport te lezen, aangezien de beide rapporten complementair zijn en samen alle door Rijkswaterstaat gestelde vragen beantwoorden.

Dit rapport is als volgt opgezet. In Hoofdstuk 4 worden de aan TNO gestelde vragen globaal beantwoord, en worden ook de aanbevelingen uit dit rapport samengevat. De rest van het rapport bevat meer gedetailleerde informatie ter onderbouwing van deze antwoorden en aanbevelingen.

Als basisinformatie wordt Synthesis-Based Engineering (SBE) algemeen geïntroduceerd in Hoofdstuk 5, waarbij het ook wordt gerelateerd aan een aantal andere mogelijke ontwikkelprocessen. Daarna wordt in Hoofdstuk 6 ingegaan op de visie en plannen van Rijkswaterstaat rondom het introduceren van SBE. Daarbij komen ook de al gemaakte keuzes aan bod, de zorgen die nog leven en die tot uitdrukking komen in de aan TNO gestelde vragen, en de aanpak die TNO volgt in dit rapport. In Hoofdstuk 7 wordt vervolgens in meer detail ingegaan op zowel de potentiële kansen als de aandachtspunten bij het introduceren van SBE. Deze kansen en aandachtspunten worden, waar mogelijk en relevant, gerelateerd aan het ontwikkelproces van SBE en de gefaseerde introductie van SBE zoals Rijkswaterstaat die voor ogen heeft. Ook worden diverse aanbevelingen gedaan om de risico’s van de aandachtspunten te beperken. Ter afsluiting bevat Hoofdstuk 8 nog een verantwoording van dit rapport, en is in Hoofdstuk 9 de lijst van referenties te vinden.

4 Beantwoording van de vragen

In dit hoofdstuk worden de aan TNO gestelde vragen globaal beantwoord, en worden ook de aanbevelingen uit dit rapport samengevat. De rest van het rapport bevat meer gedetailleerde informatie ter onderbouwing van deze antwoorden en aanbevelingen.

Van de tien door Rijkswaterstaat gestelde vragen (zie ook Sectie 6.3.2), zijn er vijf aan TNO (en TU/e) gesteld, te weten vragen 1-3, 9 en 10. De overige vragen zijn gesteld aan CGI. Voor de beantwoording van vragen 4-8 verwijst TNO naar het rapport van CGI [4]. Samen beantwoorden de twee rapporten daarmee de tien door Rijkswaterstaat gestelde vragen.

De aan TNO gestelde vragen worden in principe beantwoord met dit rapport als geheel. De globale antwoorden zoals gegeven in dit hoofdstuk kunnen namelijk niet alle nuances zoals beschreven in dit gehele rapport meenemen, en dienen dus samen met de rest van het rapport te worden beschouwd.

In diverse vragen verwijst Rijkswaterstaat specifiek naar de Eclipse ESCET toolkit, een toolkit voor SBE. TNO beschouwt in dit rapport SBE in het algemeen, met Eclipse ESCET als een mogelijke toolkit om SBE toolmatig te ondersteunen (zie ook Sectie 6.4).

4.1 Vraag 1: Toepasbaarheid SBE voor Rijkswaterstaat en marktpartijen

Vraag 1

“*Is de ESCET-toolkit voor zowel RWS als marktpartijen voldoende inzetbaar om gestandaardiseerde en onderhoudbare besturingssoftware te vervaardigen voor besturingssystemen van infrastructurele objecten, zoals beweegbare bruggen en sluizen?”*

Kort antwoord

Momenteel is SBE met de Eclipse ESCET toolkit voor Rijkswaterstaat en marktpartijen nog niet voldoende inzetbaar voor het hele proces van het vervaardigen van de besturingssoftware. Diverse zaken moeten nog verder worden uitgezocht, moeten worden uitgewerkt, en moeten worden geregeld. Dit is logisch gezien de huidige fase waarin Rijkswaterstaat zich bevindt. Er zijn al tal van zaken onderzocht en aangetoond, voornamelijk in de academische samenwerking met de TU/e. Maar er is nog niet definitief en volledig voor SBE gekozen. In dit rapport zijn diverse aandachtspunten benoemd, welke zijn samengevat in het antwoord op vraag 3. Deze dienen te worden geadresseerd om SBE met de ESCET toolkit wel voldoende inzetbaar te maken voor Rijkswaterstaat en marktpartijen, zodat aan alle regels en richtlijnen wordt voldaan, en de vele potentiële voordelen van SBE daadwerkelijk tot hun recht kunnen komen. Voor een aantal van deze aandachtspunten wordt er momenteel door Rijkswaterstaat al gewerkt aan het adresseren ervan.

Vraag 2 gaat verder in op de sterke punten en kansen van SBE. Vraag 3 gaat verder in op de zwaktes en bedreigingen met betrekking tot SBE, en de daaruit volgende aanbevelingen.

Volledig antwoord

SBE kan in de diverse stappen van het ontwikkelproces van besturingssoftware tal van voordelen bieden en garanties geven. Het maakt het onder andere mogelijk om eenduidige, complete, consistente en up-to-date specificaties te vervaardigen, en daarmee sneller en goedkoper besturingssoftware te realiseren met een nog betere kwaliteit en veiligheid. De potentie hiervan is in de samenwerking van Rijkswaterstaat met de TU/e aangetoond, en SBE is toepasbaar op infrastructurele systemen. De schaalbaarheid van de formele technieken is daarbij nog niet definitief aangetoond, hoewel de vooruitzichten positief zijn.

Ook bij onderhoud en evolutie zijn deze voordelen in potentie te behalen. De gewenste voordelen op het gebied van efficiëntie en kwaliteit, die Rijkswaterstaat voor ogen staan bij de introductie van productfamilies en een configure-to-order manier van werken, zijn eveneens gebleken uit de samenwerking tussen

Rijkswaterstaat en TU/e. Hierbij dienen echter nog een aantal zaken verder uitgewerkt te worden, en moet er meer ervaring mee worden opgedaan, aangezien deze aspecten in die samenwerking tot nu toe nog wat minder aandacht hebben gekregen.

De TU/e heeft een terugblik gepubliceerd rondom recente ontwikkelingen in samenwerking met Rijkswaterstaat [5], waarin ook vooruit wordt gekeken naar de toekomst. De conclusie van de TU/e daarin is (vertaald vanuit het Engels): “De casussen zoals gepresenteerd in dit artikel, in de context van een project met Rijkswaterstaat, laten zien dat de theorie een niveau van volwassenheid heeft bereikt dat het toestaat om volledig te worden omarmd door de industrie.”

SBE moet echter niet alleen vanuit de theorie worden bekeken, maar ook vanuit de toepassing, waaronder de aspecten rondom de samenwerking met marktpartijen en de volwassenheid van SBE en de Eclipse ESCET toolkit. Deze zijn minder aan bod gekomen in de academische samenwerking met de TU/e. Als gekozen wordt voor SBE is de keuze voor Eclipse ESCET te verantwoorden, en de ontbrekende tool functionaliteit kan door Rijkswaterstaat en de TU/e binnen het Eclipse ESCET project worden toegevoegd. Het ontbreken van commerciële ondersteuning en het beperkte ecosysteem zijn hierbij belangrijke aandachtspunten. Ook de beperkt aanwezige kennis en kunde bij zowel Rijkswaterstaat als marktpartijen voor het toepassen van SBE, met een nog te beperkte visie op hoe dit verder op te bouwen, evenals het verandermanagement en het eigenaarschap dat Rijkswaterstaat dient te pakken, verdienen aanzienlijk meer aandacht dan ze momenteel krijgen.

Het is echter logisch om eerst te beginnen met het aantonen dat een nieuwe manier van werken waarde heeft, en bijdraagt aan de gestelde visie, voordat er meer in wordt geïnvesteerd. Dat heeft Rijkswaterstaat met de TU/e gedaan door middel van academisch onderzoek en tal van toepassingen daarvan door vooral studenten. Rijkswaterstaat overweegt nu óf en waar het wel of niet SBE wil gaan inzetten, om de gestelde visie te realiseren. Hiermee gaat Rijkswaterstaat een volgende fase in, waarbij ook andere aspecten een rol gaan spelen, en die aspecten voldoende aandacht moeten krijgen. Hiertoe heeft Rijkswaterstaat een pad naar de toekomst geschetst, onder andere met proefprojecten. Zo wordt stap voor stap verder gekeken, waarbij ook de aandachtspunten uit dit rapport verder kunnen worden bekeken. Elke stap wordt geëvalueerd en er wordt dan besloten of er wordt overgegaan tot bredere inzet van SBE. Rijkswaterstaat kan hierbij de aandachtspunten en adviezen uit dit rapport gebruiken.

TNO beschrijft namelijk in dit rapport hoe SBE verschilt van een traditionele manier van ontwikkelen, de diverse aspecten die daarbij komen kijken, en wat de meerwaarde en risico’s zijn. Hiermee kan Rijkswaterstaat beter bepalen in hoeverre het SBE geschikt vindt voor het vervaardigen van besturingssoftware voor besturingssystemen van infrastructurele objecten. Concreet gezien kan het deze informatie gebruiken bij de proefprojecten en de evaluaties daarvan, omdat het met dit rapport beter zou moeten kunnen inschatten wat de voordelen zijn ten opzichte van de huidige manier van werken, wat de risico’s zijn, of het de risico’s voldoende denkt te kunnen beperken, en hoe het de verhouding tussen de meerwaarde en (overgebleven) risico’s ziet. Rijkswaterstaat zal uiteindelijk zelf moeten besluiten hoe zwaar het bepaalde aspecten, voordelen en risico’s weegt, tot welke conclusies het daarmee komt, en dus of het SBE al dan niet breder in wil zetten. Idealiter komt Rijkswaterstaat daarbij tot een breed draakvlak voor deze conclusies en maakt het daarmee een duidelijke en weloverwogen keuze omtrent de verdere inzet van SBE.

4.2 Vraag 2: Sterke punten en kansen van SBE

Vraag 2

“*Wat zijn sterke punten en kansen ten aanzien van de inzetbaarheid van de ESCET-toolkit voor softwaregeneratie voor besturingssystemen van infrastructurele objecten, zoals beweegbare bruggen en sluizen?”*

Antwoord

Vraag 2 betreft de verdere uitdetaillering van vraag 1 met betrekking tot de voordelen (sterke punten en kansen) van SBE. De voordelen komen kort aan bod in Hoofdstuk 5, en worden uitgebreider besproken in Hoofdstuk 7. Hier volgt een samenvatting daarvan, per relevant aspect.

Ontwikkelproces: De invloed van SBE ten opzichte van traditioneel ontwikkelen voor het ontwikkelproces van besturingssoftware is in elke stap zichtbaar (zie Sectie 7.2). Door model-gebaseerde specificaties (zie Sectie 7.2.1) wordt het mogelijk besturingseisen eenduidig vast te leggen, wat communicatie bevordert en helpt bij het voorkomen van onduidelijkheden, misinterpretaties, fouten en discussies. Het verkleint de kloof tussen de specificatie en de realisatie, en brengt daarmee verschillende disciplines dichter bij elkaar. Het ontwerp van de besturing wordt geautomatiseerd door besturingssynthese toe te passen (zie Sectie 7.2.2), wat een correct-door-constructie besturingsmodel oplevert. In tegenstelling tot mensen, die bij complexe zaken vaak fouten maken, voldoet het gesynthetiseerde besturingsmodel gegarandeerd in alle situaties aan alle gestelde besturingseisen, ook voor grote complexe objecten met veel besturingseisen die lastig zijn om te overzien, en indien het is meegemodelleerd ook voor gevallen waarbij de hardware faalt. Voor de realisatie wordt codegeneratie ingezet (zie Sectie 7.2.3), zodat automatisch foutloze code kan worden verkregen die zich precies gedraagt zoals het besturingsmodel voorschrijft. In plaats van (of in toevoeging op) tijdrovend en incompleet testen, geeft synthese hiermee tal van wiskundige garanties. Zaken die door constructie gegarandeerd zijn hoeven niet te worden geverifieerd, maar voor overige eigenschappen kan met formele verificatie, zoals *model checking*, alsnog automatisch wiskundig worden bewezen dat er in alle mogelijke situaties aan wordt voldaan (zie Sectie 7.2.4). Hoewel met verificatie dus kan worden bewezen dat de besturing aan alle gestelde besturingseisen voldoet, kan het zijn dat niet de juiste eisen zijn gespecificeerd. SBE biedt voor de validatie van de besturing(seisen) diverse mogelijkheden aan voor simulatie, op diverse niveaus waaronder modelsimulatie en *hardware-in-the-loop* simulatie, bijvoorbeeld met *digital twins* (zie Sectie 7.2.5). Dit maakt *rapid prototyping* mogelijk, wat effectief inzicht geeft in het gedrag van de besturing, en waarmee vaak zowel sneller als meer fouten en inconsistenties gevonden kunnen worden, waarmee risico's worden beperkt. Dit kan tevens plaatsvinden tijdens de eerdere stappen van het ontwikkelproces, waar de problemen goedkoper opgelost kunnen worden.

Onderhoud en evolutie: Naast de ontwikkeling van de eerste versie van de besturingssoftware van een object, heeft SBE ook invloed op het onderhoud en de evolutie ervan (zie Sectie 7.3). Zo kunnen de modellen extra mogelijkheden bieden bij het diagnosticeren van problemen (zie Sectie 7.3.1). Daarnaast staan de modellen centraal tijdens de gehele levenscyclus van het object. Alle wijzigingen, analyses en generaties vinden plaats op of vanuit de modellen, waardoor de modellen up-to-date worden gehouden en de specificatie, het ontwerp en de realisatie consistent blijven (zie Sectie 7.3.2). Omdat hetzelfde proces wordt gevolgd als voor de eerste ontwikkeling van de besturingssoftware, gelden de voordelen daarvan tot op zekere hoogte ook voor onderhoud en evolutie (zie Sectie 7.3.3).

Hergebruik en standaardisatie: SBE heeft ook invloed op het ontwikkelproces van productfamilies (zie Sectie 7.4). De door Rijkswaterstaat voorziene hergebruik en standaardisatie, en de *configure-to-order* manier van werken zijn mogelijk met SBE door het gebruik van (een bibliotheek met) modulaire plant en requirement modellen. Via hergebruik en standaardisatie kunnen veelgemaakte fouten worden voorkomen, wat de kwaliteit ten goede kan komen. Verder leveren ze efficiëntiewinst op, omdat veel werk al voor eerdere objecten is gedaan. Het configure-to-order principe kan verder de benodigde kennis en kunde reduceren.

Garanties van SBE: Daarbij heeft SBE een significante impact op het ontwikkelproces. Het geeft (extra) wiskundige garanties (zie Sectie 7.5). Het gebruik van formele modellen en technieken, zoals synthese, codegeneratie, model checking en simulatie, kan hierbij veel fouten voorkomen en daarmee de kwaliteit en veiligheid van het object nog verder te verhogen. SBE maakt vergaand gebruik van formele methoden. Die maken het volgens de auteurs van de TOPAAS methode, die Rijkswaterstaat gebruikt voor het bepalen van faalkansen, mogelijk de faalkans ten gevolge van logische softwarefouten de absolute nul te laten benaderen. Ook kan SBE efficiëntiewinst opleveren en de afhankelijkheid van leveranciers verkleinen.

Veiligheid en betrouwbaarheid: Het gebruik van SBE kan namelijk op tal van manieren de kwaliteit van de besturingseisen en de veiligheid van de besturing nog verder helpen te verhogen (zie Sectie 7.6.1), zoals door met eenduidige modellen communicatie te verbeteren en misinterpretaties, fouten en discussies te voorkomen, door tal van zaken te automatiseren en daarmee menselijk fouten te reduceren, door met efficiëntiewinst meer tijd te geven voor zaken die de veiligheid bevorderen, door met extra validatieopties meer fouten te vinden, door meer formele correctheidsgaranties te geven dan traditionele aanpakken, door met hergebruik en standaardisatie eerder gemaakte fouten te voorkomen, en door dit niet alleen bij de normale ontwikkeling maar ook bij onderhoud en evolutie mogelijk te maken. Dit verlaagt potentieel ook de faalkansen van de software, en verhoogt daarmee de betrouwbaarheid van het object (zie ook Sectie 7.6.2). Verder kan het falen van hardware worden meegenomen in het SBE ontwikkelproces, zodat de besturing ook veilig is als bijvoorbeeld sensoren of motoren stuk gaan (zie Sectie 7.6.3). CGI heeft gekeken naar de wet- en regelgeving omtrent veiligheid en gezondheid waar de objecten van Rijkswaterstaat aan moeten voldoen, en dan specifiek de Machinerichtlijn en de IEC 62061 norm (zie Sectie 7.6.4). CGI concludeert dat deze het gebruik van SBE als methode en Eclipse ESCET als toolkit niet uitsluiten, maar dat er wel randvoorwaarden worden gesteld aan het ontwerp en de wijze van risicobeheersing, hoewel deze niet uniek zijn voor SBE. CGI geeft hierbij aan dat er geen fundamentele beperking is voor SBE en de Eclipse ESCET toolkit om aan al deze eisen te kunnen voldoen, en dat SBE voor bepaalde eisen zelfs voordelen kan bieden.

Efficiëntie: Daarnaast biedt het gebruik van SBE dus ook tal van efficiëntievoordelen (zie Sectie 7.7), waaronder het met eenduidige modellen verbeteren van de communicatie om misinterpretaties, fouten en discussies te voorkomen, het vastleggen van kennis in modellen om nieuwe werknemers eerder effectief te laten zijn, het automatiseren van tal van zaken, het met korte iteraties sneller en eerder problemen vinden, het besparen van bijvoorbeeld testtijd door bepaalde formele garanties te geven, het eerder mogelijk maken van training, het met hergebruik en standaardisatie voorkomen van eerder gemaakte fouten en uitsparen van werk, en het mogelijk maken hiervan bij niet alleen de eerste ontwikkeling maar ook bij onderhoud en evolutie.

Afhankelijkheid van leveranciers: Verder kan het gebruik van SBE voor Rijkswaterstaat een positieve impact hebben op de afhankelijkheid van leveranciers (zie Sectie 7.8). De platform- en leveranciersonafhankelijke modellen die bij SBE worden gebruikt maken het mogelijk voor verschillende PLC-platformen van verschillende leveranciers PLC-code te genereren. Tevens geeft dit Rijkswaterstaat in potentie de mogelijkheid om te kunnen overstappen op industriële PC's, meerdere platformen tegelijk te kunnen gebruiken, en makkelijker te kunnen overstappen op een ander platform (zie Sectie 7.8.1). Wat betreft de samenwerking met marktpartijen die voor Rijkswaterstaat een besturing maken, staat Rijkswaterstaat met eenduidige, complete, consistente, uniforme en up-to-date formele specificaties sterker tegenover dergelijke leveranciers, kan het de realisatie toetsen tegen die specificaties, kan het indien gewenst dergelijke leveranciers (gedeeltelijk) overbodig maken door zelf de realisatie te verzorgen met codegeneratie, en kan het met het beheer van de standaardmodellen meer controle uitoefenen (zie Sectie 7.8.2). Rijkswaterstaat kan hiermee meer eigenaarschap nemen, en de afhankelijkheid van leveranciers beperken.

Geschiktheid voor infrastructurele systemen: SBE is toepasbaar op infrastructurele systemen, en de met het SBE ontwikkelproces te behalen voordelen zijn ook van toepassing voor dit soort systemen (zie Sectie 7.9). Dit is gebleken uit de veelvuldige toepassingen van SBE gedurende de onderzoeken in de samenwerking van Rijkswaterstaat met de TU/e. Daarbij is SBE toegepast voor de diverse stappen van het ontwikkelproces en op tal van objecten, van bruggen, sluizen en tunnels tot wegkantsystemen. Deze toepassingen zijn soortgelijk aan die van de TU/e en andere partijen in andere domeinen, en infrastructurele systemen passen eveneens in de SBE architectuur.

Ontwikkelproceskeuze MBE/VBE/SBE: Van de in dit rapport besproken ontwikkelprocessen voor het ontwikkelen van besturingssoftware (traditioneel, MBE, VBE, SBE), biedt SBE het meeste automatisering

en het meest vergaande gebruik van formele methoden en computer ondersteuning (zie Sectie 7.15). Daarmee heeft het in potentie de grootste winst in efficiëntie, kwaliteit en veiligheid.

4.3 Vraag 3: Aandachtspunten en aanbevelingen voor SBE

Vraag 3

"*Wat zijn zwaktes en bedreigingen ten aanzien van deze inzetbaarheid en welke beheersmaatregelen zijn voor hiervoor te nemen?"*

Antwoord

Vraag 3 betreft de verdere uitdetaillering van vraag 1 met betrekking tot de aandachtspunten (zwaktes en bedreigingen) van SBE, met adviezen en aanbevelingen (beheersmaatregelen). De aandachtspunten en aanbevelingen worden uitgebreider besproken in Hoofdstuk 7. Hier volgt een samenvatting daarvan, per relevant aspect, met de daaruit volgende aanbevelingen.

Onderhoud en evolutie: Hoewel SBE modellen centraal stelt gedurende de gehele levenscyclus van het object, waardoor deze doorgaans up-to-date gehouden worden, en de specificatie, het ontwerp en de realisatie consistent blijven, kan dit mogelijk wel extra inspanning vereisen, is het mogelijk geen volledige vervanging van documentatie, en moeten de ontwerpkeuzes nog altijd worden vastgelegd (zie Sectie 7.3.2). Om de voordelen van SBE die te behalen zijn bij een eerste ontwikkeling van de besturingssoftware van een object ook te behalen bij onderhoud en evolutie, moeten hiervoor diverse zaken eerst verder worden uitgewerkt, waaronder de coördinatie, uitvoering en verantwoordelijkheden, en moet hier ervaring mee op worden gedaan (zie Sectie 7.3.3).

Aanbeveling 1a: Zorg dat ook bij het gebruik van SBE alle relevante documentatie op orde blijft, en dat de ontwerpkeuzes worden vastgelegd.

Aanbeveling 1b: Werk de werkwijze voor onderhoud en evolutie verder uit, waaronder de coördinatie, uitvoering en verantwoordelijkheden, en doe hier ervaring mee op.

Hergebruik en standaardisatie: Hoewel de door Rijkswaterstaat voorziene *configure-to-order* manier van werken mogelijk is met SBE, dient dit voor sluizen nog wel nader aangetoond te worden. Ook dient het verder uitgewerkt te worden, inclusief het beheer van de modellen, en ook in combinatie met onderhoud en evolutie (zie Sectie 7.4).

Aanbeveling 2a: Toon de haalbaarheid van de configure-to-order manier van werken met SBE ook aan voor sluizen.

Aanbeveling 2b: Werk de configure-to-order manier van werken verder uit, inclusief beheer van de modellen, en in combinatie met onderhoud en evolutie.

Veiligheid en betrouwbaarheid: CGI heeft gekeken naar de wet- en regelgeving omtrent veiligheid en gezondheid waar de objecten van Rijkswaterstaat aan moeten voldoen, en dan specifiek de Machinerichtlijn en de IEC 62061 norm (zie Sectie 7.6.4). CGI ziet daarbij geen fundamentele beperkingen voor SBE en de Eclipse ESCET toolkit om aan alle eisen te kunnen voldoen, en ziet bij het gebruik van SBE voor bepaalde eisen zelfs voordelen. CGI merkt echter wel op dat nu nog niet alles in ingeregeld om daadwerkelijk aan alle eisen te voldoen, en komt dan ook met diverse aanbevelingen. Gezien dat nogal wat aandachtspunten naar boven komen die nieuw zijn ten opzichte van de samenwerking met de TU/e, is het aan te raden de aanbevelingen van CGI te adresseren met een partij die verstand heeft van en praktische ervaring heeft met, zaken als de machinerichtlijn, PLC's en PLC-code, en het daadwerkelijk realiseren van machinebesturingen.

Aanbeveling 3a: Bestudeer de aanbevelingen vanuit het CGI rapport en adresseer die aanbevelingen, samen met een partij die verstand heeft van, en praktische ervaring heeft met, zaken als de machinerichtlijn, PLC's en PLC-code, en het daadwerkelijk realiseren van machinebesturingen.

Aanbeveling 3b: Overweeg om een partij met verstand van, en praktische ervaring met, zaken als de machinerichtlijn, PLC's en PLC-code, en het daadwerkelijk realiseren van machinebesturingen, te laten aansluiten bij het Eclipse ESCET open source project, en bijvoorbeeld te laten adviseren bij het maken van de nieuwe PLC-codegenerator.

Efficiëntie: Het gebruik van SBE biedt tal van efficiëntievoordelen, onder andere door vergaande automatisering (zie Sectie 7.7). De keerzijde is dat bij vergaande automatisering er minder momenten zijn dat mensen naar bijvoorbeeld het ontwerp en de code kijken, wat potentieel minder momenten geeft waarop die personen fouten kunnen vinden. Ook vergt het een initiële investering om SBE te introduceren, waardoor de voordelen pas bij toepassing op meerdere objecten (volledig) zichtbaar zijn.

Aanbeveling 4a: Verwerk in de plannen voor het introduceren van SBE beheersmaatregelen om de keerzijdes van SBE te adresseren, waaronder dat er minder momenten zijn waarop mensen naar het ontwerp en de code wordt gekeken.

Aanbeveling 4b: Houdt bij de (deel)evaluaties rondom proefprojecten met SBE rekening met het feit dat de introductie van SBE een initiële investering vergt, en dat de potentiële voordelen van SBE niet altijd direct zichtbaar zijn bij de eerste toepassing ervan.

Afhankelijkheid van leveranciers: Het gebruik van SBE kan voor Rijkswaterstaat een positieve impact hebben op de afhankelijkheid van leveranciers. Daarbij is het dan wel van belang dat Rijkswaterstaat hier meer ervaring mee opdoet, dit verder uitwerkt, goed inricht wat betreft zaken als verantwoordelijkheden, coördinatie en begeleiding, en dat het dit eigenaarschap dan ook echt pakt (zie Sectie 7.8).

Aanbeveling 5a: Doe verdere ervaring op met het effect van SBE op de relatie met leveranciers en marktpartijen.

Aanbeveling 5b: Werk verder uit hoe de toepassing van SBE de relatie met leveranciers en marktpartijen beïnvloedt, waaronder hoe zaken als verantwoordelijkheden, coördinatie en begeleiding worden ingericht.

Aanbeveling 5c: Neem het noodzakelijke eigenaarschap, zeker ook wat betreft de verantwoordelijkheden die bij Rijkswaterstaat zelf liggen, waaronder met betrekking tot modellen van productfamilies, en coördinatie tussen en begeleiding van leveranciers en marktpartijen.

Schaalbaarheid van formele technieken: Momenteel kan nog niet definitief worden geconcludeerd dat de schaalbaarheid van de bij SBE gebruikte formele technieken voldoende zijn voor het gebruik op infrastructurele systemen (zie Sectie 7.10). Hoewel bij het gebruik van de juiste state-of-the-art technieken voor de juiste situatie de vooruitzichten positief zijn, dient bijvoorbeeld multi-level synthese ook toegepast te worden op het complexe Algeracomplex, inclusief het synthetiseren van een fault-tolerant supervisor. Verder dienen er uitgebreidere regels en richtlijnen opgesteld te worden voor het maken van (herbruikbare) modellen die efficiënt synthetiseerbaar zijn.

Aanbeveling 6a: Toon aan dat de bij SBE gebruikte formele technieken ook schaalbaar genoeg zijn om onder andere rapid prototyping mogelijk te maken voor het Algeracomplex, inclusief het met multi-level synthese synthetiseren van een fault-tolerant supervisor.

Aanbeveling 6b: Stel uitgebreidere regels en richtlijnen op voor het maken van (herbruikbare) modellen die efficiënt synthetiseerbaar zijn.

Tooling: In hoeverre de potentiële voordelen van SBE haalbaar zijn voor infrastructurele systemen, hangt verder af van de volwassenheid van SBE. Een aspect daarbij is de tooling die voor SBE beschikbaar is,

waaronder de open source Eclipse ESCET toolkit (zie Sectie 7.11). Ondanks dat enkele voor Rijkswaterstaat essentiële algoritmen op dit moment nog niet beschikbaar zijn in de ESCET toolkit (zie Sectie 7.11.1), is het de enige toolkit die het gehele ontwikkelproces van SBE ondersteunt inclusief synthese, én dat doet via een enkele modelleertaal, én een focus heeft op industriële toepasbaarheid, én wordt ontwikkeld via een open ecosysteem (zie Sectie 7.11.2). Bij het ontwikkelen van deze toolkit wordt veel aandacht besteed aan de kwaliteit ervan. Verder bestaat de toolkit al geruime tijd. Het garanderen van volledige afwezigheid van bugs in complexe tooling is echter een utopie (zie Sectie 7.11.3). Certificering voor synthese tooling lijkt voor Rijkswaterstaat momenteel geen harde eis, maar kan dit (in de toekomst) eventueel wel overwegen (zie Sectie 7.11.4). Ook commerciële ondersteuning is voor synthese tooling nog niet voorhanden. Om risico's te beperken dient Rijkswaterstaat hier wel voor te zorgen, indien het SBE breder wil gaan toepassen. Rijkswaterstaat is bezig een partij te vinden die deze ondersteuning kan bieden (zie Sectie 7.11.5).

Aanbeveling 7a: Zorg ervoor dat alle voor toepassingen van SBE door Rijkswaterstaat en marktpartijen benodigde functionaliteit beschikbaar is in een gereleasete versie van de Eclipse ESCET toolkit.

Aanbeveling 7b: Zorg dat de functionaliteit van de Eclipse ESCET toolkit voldoende is getest en betrouwbaar is bevonden voor de toepassingen van Rijkswaterstaat en marktpartijen.

Aanbeveling 7c: Overweeg certificering van de Eclipse ESCET toolkit alleen als de voordelen hiervan opwegen tegen de grote kosten en moeite die hiermee gemoeid zijn, en in dat geval ook samen met marktpartijen die de toolkit commercieel aanbieden.

Aanbeveling 7d: Zorg dat de Eclipse ESCET toolkit commercieel ondersteund wordt voor zowel Rijkswaterstaat als door Rijkswaterstaat ingeschakelde marktpartijen.

Kennis en kunde: Om SBE en de bijbehorende tooling correct in te kunnen zetten, is de juiste kennis en kunde vereist (zie Sectie 7.12), bij Rijkswaterstaat en marktpartijen. MBE en SBE vragen specifieke kennis en een andere manier van denken en werken. Naast het onderwijsaanbod van de TU/e, is de online SBE cursus waar de TU/e aan werkt slechts een eerste stap. Er zal ook een cursus opgezet moeten worden waarbij met een trainer wordt gewerkt en vragen kunnen worden gesteld. Verder moeten de cursussen voldoende geschikt zijn om naast studenten ook marktpartijen mee op te leiden. De SBE kennis en kunde is nog onvoldoende bij marktpartijen aanwezig. Deze kan deels worden opgedaan in de door Rijkswaterstaat voorziene proefprojecten. Echter, er zal goed gekeken moeten worden hoeveel marktpartijen nodig zijn, of het aantal proefprojecten dan toereikend is, en of de plannen ter vervanging- en renovatie van alle objecten haalbaar zijn in het gewenste tijdsbestek. Daarbij dient ook een bredere visie uitgewerkt te worden, waarbij onder andere gekeken wordt naar hoeveel personen bij Rijkswaterstaat en marktpartijen expertkennis en kunde nodig hebben, en of dergelijk personeel opgeleid kan worden of vanuit de arbeidsmarkt kan worden aangenomen.

Aanbeveling 8a: Zorg voor voldoende mogelijkheden voor zowel personeel van Rijkswaterstaat zelf als voor marktpartijen om de voor een juiste toepassing van SBE benodigde kennis en kunde op te kunnen doen, waaronder voldoende geschikte opleidingsmogelijkheden en proefprojecten.

Aanbeveling 8b: Werk de verdere visie uit rondom SBE en de rolverdeling tussen Rijkswaterstaat en marktpartijen, met daarin onder andere hoeveel marktpartijen en deskundige personen nodig zijn om SBE toe te passen voor objecten van Rijkswaterstaat, binnen welk tijdsbestek deze partijen voldoende kennis en kunde dienen op te doen, en een arbeidsmarkt- en haalbaarheidsanalyse.

Veranderen van bedrijfsprocessen en bedrijfscultuur: Naast het opbouwen van de juiste kennis en kunde, moet ook de verandering van de bedrijfsprocessen en bedrijfscultuur niet onderschat worden (zie Sectie 7.13). Zeker het introduceren van SBE heeft nogal wat veranderingen ten gevolg. Het is daarom essentieel om aan goed verandermanagement te doen, waaronder het stellen van duidelijke doelen, het inbouwen van voldoende meetpunten in het proces, het hanteren van de juiste evaluatiecriteria bij (deel)evaluaties,

en het identificeren en oplossen van pijnpunten, ondersteund door goede communicatie en het nemen van voldoende tijd. Gedeeltelijk zal het een en ander duidelijk worden tijdens proefprojecten. Ook zijn realistische verwachtingen essentieel, kan een nog meer gefaseerde introductie van SBE worden overwogen dan al was voorzien, en kan samen met marktpartijen onderzocht worden hoe de adoptie van SBE het beste kan worden ingestoken. Daarnaast kan meer ervaring worden opgedaan met het veranderproces. Verder is het te overwegen om ervaringen te delen met bedrijven die overwegen om, plannen hebben voor, of al stappen hebben zet met het introduceren van bijvoorbeeld MBE of SBE, en kan worden overwegen een partij in te schakelen die is gespecialiseerd in het begeleiden van bedrijven tijdens dergelijke transities.

Aanbeveling 9a: Zorg dat zowel Rijkswaterstaat als marktpartijen aan gedegen verandermanagement doen bij de introductie van SBE, waaronder het stellen van duidelijke doelen, het hebben van realistische verwachtingen, het inbouwen van voldoende meetpunten in het proces, het hanteren van de juiste evaluatiecriteria bij (deel)evaluaties, en het identificeren en oplossen van pijnpunten, ondersteund door goede communicatie en het nemen van voldoende tijd.

Aanbeveling 9b: Overweeg een nog meer gefaseerde introductie van SBE zoals wordt voorgesteld in Sectie 7.1.3.

Aanbeveling 9c: Bekijk met marktpartijen hoe de adoptie van SBE het beste kan worden ingestoken.

Aanbeveling 9d: Overweeg om meer ervaring op te met het veranderproces voor SBE.

Aanbeveling 9e: Overweeg om ervaringen te delen met bedrijven die overwegen om, plannen hebben voor, of al stappen hebben zet met het introduceren van bijvoorbeeld MBE of SBE.

Aanbeveling 9f: Overweeg om een partij in te schakelen die is gespecialiseerd in het begeleiden van bedrijven tijdens transities als het introduceren van nieuwe manieren van werken, zoals MBE of SBE.

Ecosysteem: Ook de volwassenheid van het SBE ecosysteem is van belang (zie Sectie 7.14). SBE als onderzoeksgebied bestaat sinds de jaren ’80, en er wordt door diverse universiteiten actief onderzoek naar gedaan. Wat betreft Eclipse ESCET kan nog niet gesproken worden van een bloeiend ecosysteem, gezien het beperkt aantal ontwikkelaars en externe bijdragen. Eclipse ESCET wordt toegepast door studenten van meerdere universiteiten, zowel in onderwijs als bij projecten in de industrie. Echter, adoptie in de industrie is er nog vrijwel niet. Daarmee is sprake van een mager ecosysteem, en dient bij een bredere inzet van Eclipse ESCET ook te worden geïnvesteerd in het uitbreiden van het ecosysteem om risico’s te verkleinen.

Aanbeveling 10: Investeer bij een keuze voor toolondersteuning, zoals Eclipse ESCET, in het uitbreiden van het ecosysteem rondom die tooling.

Ontwikkelproceskeuze MBE/VBE/SBE: SBE heeft van de in dit rapport besproken ontwikkelprocessen voor het ontwikkelen van besturingssoftware (traditioneel, MBE, VBE, SBE) in potentie de grootste winst in efficiëntie, kwaliteit en veiligheid. Het verschilt daarmee echter ook het meest van traditioneel ontwikkelen, is het minst gemeengoed van deze alternatieven, en heeft het kleinste ecosysteem, waarbij ook commerciële ondersteuning momenteel nog ontbreekt. Mochten Rijkswaterstaat en/of marktpartijen, ondanks een gefaseerde introductie en de aanbevelingen uit dit rapport, er niet in slagen SBE en Eclipse ESCET (volledig) te introduceren, dan kan worden overwogen (gedeeltelijk) terug te vallen op (een mix met) VBE en/of MBE, en de daarvoor beschikbaar tools.

Aanbeveling 11a: Heb bij de introductie van SBE oog voor de grote verschillen met de huidige manier van werken, de nadelen en aandachtspunten met betrekking tot deze methode, en neem beheersmaatregelen om deze zo goed mogelijk te adresseren.

Aanbeveling 11b: Mocht het (volledig) introduceren van SBE en Eclipse ESCET niet slagen, overweeg dan om (gedeeltelijk) terug te vallen op (een mix met) VBE en/of MBE, en de daarvoor beschikbaar tools.

4.4 Vragen 4-8: Wet- en regelgeving, veiligheid en betrouwbaarheid

Vraag 4

"Is het mogelijk middels de ESCET-toolkit besturingssoftware te vervaardigen die in overeenstemming is met de machinerichtlijn en de relevante geharmoniseerde normen en is het toepassen van ESCET een belemmering om te voldoen aan SIL of PL levels?"

Vraag 5

"Zouden de modellen en de softwarecode die in de Proof of Concept worden vervaardigd voor de Oisterwijksebaanbrug in de praktijk van een op te zetten renovatieproject veilig te maken zijn, in overeenstemming met de machinerichtlijn en met eventueel relevante geharmoniseerde normen?"

Vraag 6

"Indien het antwoord op vraag 5 negatief is, welke aanpassingen aan de modellen en tools in ESCET moeten dan plaatsvinden om softwarecode te verkrijgen die in overeenstemming is met de machinerichtlijn en de relevante geharmoniseerde normen?"

Vraag 7

"Kan de gegenereerde code door compilers in de praktijk van een op te zetten renovatieproject betrouwbaar worden omgezet naar machinecode, zowel van veel gebruikte PLC-fabrikanten zoals Siemens, ABB en Schneider, als van veel gebruikte industriële pc's?"

Vraag 8

"Kan de verkregen code in de praktijk van een op te zetten renovatieproject betrouwbaar op een PLC dan wel industriële pc werkend gemaakt worden, als gebruik gemaakt wordt van de modellen op basis waarvan de software is gegenereerd?"

Antwoord

Vragen 4-8 zijn door CGI beantwoord in het CGI rapport [4]. De aanbevelingen uit het CGI rapport zijn meegenomen in dit TNO rapport.

4.5 Vragen 9 en 10: Garanties en betrouwbaarheid bij gebruik SBE

Vraag 9

"*Voldoet de softwarecode die wordt gegenereerd middels ESCET-toolkit altijd aan de ingevoerde plant- en eisenmodellen? De beantwoording van deze vraag kwalitatief/theoretisch onderbouwen door beschouwing van het algoritme."*

Vraag 10

"*Wordt door het toepassen van SBE middels de ESCET-toolkit betrouwbare software gegenereerd? Hiermee wordt bedoeld dat deze altijd op dezelfde wijze functioneert als de aangeleverde modellen en de besturing niet in ongedefinieerde toestanden terecht kan komen. De beantwoording van deze vraag kwalitatief/theoretisch onderbouwen."*

Antwoord

Vragen 9 en 10 relateren sterk aan het SBE ontwikkelproces (zie Secties 5.6 en 7.2) en de theoretische garanties die SBE geeft tijdens het ontwikkelproces (Sectie 7.5).

Bij SBE wordt via besturingssynthese een correct-door-constructie model van de *supervisor* gegenereerd uit de gespecificeerde plant- en eisenmodellen. Vanuit de wiskundige theorie is onder andere bewezen dat deze in alle situaties, voor alle toestanden, aan alle gespecificeerde besturingseisen voldoet, en dat het bestuurde systeem niet blokkeert. Het supervisormodel wordt vervolgens automatisch gecontroleerd op een aantal wiskundige eigenschappen die moeten gelden om een *implementeerbare supervisor* te zijn, waaronder dat de supervisor altijd de juiste keuzes maakt, dat het die keuzes binnen eindige tijd maakt, en dat de theoretische garanties vanuit de besturingssynthese ook gelden als de supervisor op een PLC wordt uitgevoerd. Gelden deze eigenschappen, dan kan de supervisor automatisch worden omgezet in een correct-door-constructie model van de *controller*, en kan uit het controllermodel automatisch de correct-door-constructie PLC-code van de besturingssoftware worden gegenereerd Codegeneratie garandeert dan dat de gegenereerde code zich precies zo gedraagt als het controller model, en via de gecontroleerde eigenschappen de garanties van besturingssynthese, dat deze zich ook precies zo gedraagt als is vastgelegd in de plant- en eisenmodellen.

Deze theoretische garanties gelden uiteraard alleen als de algoritmes ook juist zijn geïmplementeerd in de toolondersteuning, zoals de Eclipse ESCET toolkit (zie Sectie 7.11), en als ze juist worden ingezet (zie Sectie 7.12). De veiligheid en betrouwbaarheid van de besturingssoftware wordt in iets algemenere zin, zowel theoretisch als praktisch, uitgebreider besproken in Sectie 7.6.

5 Synthesis-Based Engineering (SBE)

In dit hoofdstuk wordt synthese-gebaseerd ontwikkelen (*Synthesis-Based Engineering* of *SBE* in het Engels) geïntroduceerd als achtergrondinformatie. Eerst wordt besturingssoftware zoals dat bij Rijkswaterstaat wordt gebruikt kort toegelicht (Sectie 5.1). Daarna wordt SBE in algemene zin gerelateerd aan een aantal andere processen voor het ontwikkelen van besturingssoftware (Sectie 5.2). Vervolgens worden die ontwikkelprocessen in iets meer detail toegelicht, te weten traditioneel ontwikkelen (Sectie 5.3), model-gebaseerd ontwikkelen (Sectie 5.4), verificatie-gebaseerd ontwikkelen (Sectie 5.5) en synthese-gebaseerd ontwikkelen (Sectie 5.6). Uiteindelijk wordt een aantal belangrijke begrippen nog een keer samengevat (Sectie 5.7).

Het doel van dit hoofdstuk is niet om compleet te zijn, maar om wat intuïtie te geven over wat SBE ongeveer inhoudt. Daarmee kan de rest van dit rapport beter worden begrepen, zeker voor diegenen zonder enige kennis van SBE. De tekst van dit hoofdstuk is hoofdzakelijk gebaseerd op de documentatie van de Eclipse ESCET website [6], en gaat met name in op de voordelen van alternatieve ontwikkelprocessen ten opzichte van de traditionele manier van werken. In Hoofdstuk 7 wordt in meer detail ingegaan op wat SBE inhoudt. Daar wordt SBE zowel uitgebreider als genuanceerder vergeleken met vooral een traditionele manier van werken, zowel qua kansen als aandachtspunten.

5.1 Besturingssoftware

Infrastructurele objecten bestaan uit tal van onderdelen, die moeten worden aangestuurd om de juiste functionaliteit te bewerkstelligen. Denk bijvoorbeeld bij een sluis/brug-combinatie aan de sluisdeuren die open en dicht moeten kunnen, slagbomen voor het wegverkeer die omhoog en omlaag moeten kunnen, verkeerslichten voor zowel het varende als rijdende verkeer die de juiste lichtsignalen moeten geven, en bedienpanelen met knoppen voor de handmatige besturing van de sluis, bijvoorbeeld voor het aansturen van die sluisdeuren en slagbomen.

Dit alles moet worden aangestuurd door besturingssoftware. Deze software moet bijvoorbeeld voorkomen dat een verkeerslicht voor het wegverkeer groen licht kan geven terwijl de brug nog open staat, of dat de sluis dicht gaat terwijl er nog een schip door de sluis vaart. De besturingssoftware moet er dus voor zorgen dat een object te allen tijde veilig is.

Bij objecten van Rijkswaterstaat wordt de software vaak op een *Programmable Logic Controller* (PLC) gezet. PLC's worden veel gebruikt voor industriële automatisering. Ze kenmerken zich door hun robuustheid en grote betrouwbaarheid. Dit komt de langdurige veilige operatie van infrastructurele systemen ten goede. Een mogelijk alternatief is industriële PC's, waarvoor het ontwerpproces van de besturingssoftware grotendeels soortgelijk is als voor PLC's. In dit rapport wordt voor de overzichtelijkheid voornamelijk gesproken over het gebruik van PLC's.

In de basis is de besturingssoftware simpel. De machinebesturing kan signalen binnen krijgen, zoals bijvoorbeeld een signaal vanuit een centrale aanstuurlocatie elders in het land, een signaal van een lokaal ingedrukte knop, of een signaal van een van de vele sensoren die in het object aanwezig zijn. Voor elk binnenkomend signaal bepaalt de besturingssoftware wat er moet gebeuren, afhankelijk van de huidige situatie. Wordt er bijvoorbeeld gevraagd om een sluisdeur te sluiten of een brug te openen, dan bepaalt de besturingssoftware of dit veilig kan, en als dat veilig kan, dan zal de besturingssoftware door het activeren van de juiste uitvoersignalen er voor zorgen dat dit daadwerkelijk gebeurt. Dit echter wel pas nadat het, bijvoorbeeld, de verkeerslichten op de juiste kleur heeft gezet, of de slagbomen voor het wegverkeer heeft neergelaten. Wordt via de handmatige bediening van het object gevraagd om een sluisdeur te sluiten terwijl dit al in gang is gezet, dan wordt de aanvraag bijvoorbeeld afgewezen of genegeerd. En als een verzoek niet veilig kan worden uitgevoerd, bijvoorbeeld het sluiten van de sluisdeuren terwijl er een schip door de sluis vaart, dan wordt het verzoek afgewezen.

Het grote aantal functionele en veiligheidseisen die nodig zijn voor een juiste en veilige werking van de sluis, maken dat de besturingssoftware toch bijzonder complex kan worden. Bij slechts 10 sensoren bijvoorbeeld, zijn er tot 1.024 gecombineerde situaties (als elke sensor onafhankelijk uit of aan kan zijn). Maar bij 20 sensoren is dit al meer dan een miljoen, en bij 30 sensoren stijgt dit naar meer dan een miljard situaties. Het aantal situaties wordt alleen maar groter als ook onverwacht gedrag wordt meegenomen, zoals de mogelijkheid dat een sensor kapot kan gaan. Ook in dat soort situaties moet het systeem zich veilig blijven gedragen. Om deze complexiteit goed onder controle te hebben is het dan ook zaak de besturingssoftware op een gedegen manier te ontwikkelen.

5.2 Verschillende ontwikkelprocessen

Besturingssoftware kan op diverse manieren worden ontwikkeld. Figuur 1 geeft een sterk versimpeld overzicht van een aantal van deze mogelijkheden, om de essentiële verschillen tussen een traditionele manier van ontwikkelen en een manier van werken volgens SBE op hoofdlijnen duidelijk te maken.

Ontwikkelproces → / ↓ Ontwikkelstap	Traditioneel Ontwikkelen	Model-gebaseerd Ontwikkelen	Verificatie-gebaseerd Ontwikkelen	Synthese-gebaseerd Ontwikkelen (*SBE*)
Ontwerp besturingseisen	Document-gebaseerd	Document-gebaseerd	**Model-gebaseerd (formeel)**	Model-gebaseerd (formeel)
Ontwerp besturing	Document-gebaseerd	**Model-gebaseerd (formeel)**	Model-gebaseerd (formeel)	**Computer-ondersteund (formeel)**
Realisatie besturing (implementatie in code)	Traditionele ontwikkeling (software programmeren)	**Code generatie (fout-vrije code)**	Code generatie (fout-vrije code)	Code generatie (fout-vrije code)
Verificatie besturing (tegen de besturingseisen)	Testen	Testen + **Model-gebaseerd testen**	**Formele verificatie (‘*model checking*’)**	**Correct-door-constructie (gegarandeerd)**
Validatie besturing (en de besturingseisen)	Testen	Testen + **Simulatie**	Testen + Simulatie	Testen + Simulatie

Legenda: Handmatig werk / Focus | (Semi-)automatisch

Figuur 1 – Sterk versimpeld overzicht van een aantal verschillende ontwikkelprocessen voor besturingssoftware.

De kolommen geven verschillende ontwikkelprocessen aan. Van links naar rechts gebruiken ze steeds meer automatisering en worden ze steeds meer ondersteund door automatisering via computers. De rijen geven typische stappen aan die moeten worden doorlopen tijdens de ontwikkeling van besturingssoftware. De cellen geven aan wat een bepaalde stap voor een proces inhoudt. De lichtgroene gekleurde cellen zijn voornamelijk, of ten minste voor een significant gedeelte, handwerk voor dat bepaalde proces. De grijs gekleurde cellen daarentegen geven aan dat een bepaalde stap voor een bepaald proces (grotendeels) is geautomatiseerd. Aangezien de stappen met meer handmatig werk typisch meer tijd kosten, geven de groene cellen ook aan waar de focus voor een bepaald proces ligt. Dikgedrukte tekst in cellen geeft een wijziging aan ten opzichte van de voorgaande kolom.

Typische stappen die worden doorlopen bij de ontwikkeling van besturingssoftware zijn, zoals gerepresenteerd door de rijen van boven naar onder:

- Bij het **ontwerp van de besturingseisen** wordt bepaald wat de besturing voor het systeem moet doen. Functionele en veiligheidseisen worden gespecificeerd, bijvoorbeeld dat bij een druk op de noodknop alle motoren onmiddellijk moeten stoppen (met een bepaalde betrouwbaarheid).

- Bij het **ontwerp van de besturing** wordt gekeken hoe de besturing de besturingseisen gaat garanderen, om effectief en veilig het systeem aan te sturen. Er wordt bijvoorbeeld voor elke situatie aangegeven wat het effect is van inkomende signalen en hoe daarop moet worden gereageerd.

- De **besturing wordt gerealiseerd** in software, de besturingssoftware. Bij het maken van de besturingssoftware wordt gebruik gemaakt het ontwerp van de besturing en bijbehorende besturingseisen.

- De **verificatie van de besturing** bestaat uit het controleren dat de gerealiseerde software voldoet aan alle opgestelde besturingseisen. Dat deze dus voldoet aan het ontwerp en juist is gerealiseerd.

- De **validatie van de besturing** bestaat uit het controleren van de gerealiseerde besturing en het ontwerp daarvan, om er zeker van te zijn dat de juiste besturing is gemaakt. Zo moeten de besturingseisen compleet en correct zijn, zodat bijvoorbeeld de sluis onder alle omstandigheden juist en veilig werkt.

De ontwikkelprocessen, zoals gerepresenteerd door de kolommen van links naar rechts, kunnen als volgt sterk versimpeld worden gekarakteriseerd:

- **Traditioneel ontwikkelen** is document-gebaseerd. Besturingseisen worden opgeschreven in documenten. Daaruit wordt het besturingsontwerp bepaald en eveneens in een document opgeschreven. Deze documenten dienen als basis voor de implementatie. Programmeurs, vaak van een andere afdeling of een marktpartij, ontwikkelen de softwarecode. Verificatie en validatie bestaan voornamelijk uit diverse vormen van testen, wat pas in een laat stadium kan plaatsvinden. Traditioneel wordt dit allemaal handmatig gedaan, wat zowel bewerkelijk als foutgevoelig is.

- **Model-gebaseerd ontwikkelen** (*Model-Based Engineering* of *Model-Driven Engineering* in het Engels) plaatst modellen centraal. De werking van de besturing wordt vastgelegd in formele modellen, zodat wiskundig gezien eenduidig vastligt wat de besturing in elke situatie moet doen. De realisatie van de besturing wordt vervolgens dikwijls geautomatiseerd, door met een computer vanuit de modellen automatisch softwarecode te genereren. Dit zorgt er voor dat de code in principe fout-vrij is en zich precies zo gedraagt als is vastgelegd in het model. Het model kan ook worden gebruikt voor computer ondersteuning bij verificatie en validatie, bijvoorbeeld door gebruik te maken van model-gebaseerd testen en simulatie. Hierbij kan de computer bepaalde zaken automatiseren, wat tijd scheelt bij het uitvoeren van deze activiteiten. Verder kan dit virtueel gebeuren, en al in een vroeg stadium tijdens de ontwikkeling, ook als bijvoorbeeld de echte sluis nog niet is gerealiseerd.

- **Verificatie-gebaseerd ontwikkelen** is een manier van model-gebaseerd ontwikkelen waarbij met computer-ondersteuning de verificatiestap wordt geautomatiseerd. Door het gebruik van formele verificatie (zoals *model checking*), kan het besturingsmodel automatisch wiskundig worden vergeleken met de besturingseisen. Hiervoor moeten zowel het besturingsmodel als de besturingseisen worden gemodelleerd, zodat een computer deze kan interpreteren. Een computerprogramma bewijst dan ofwel dat in het model aan de gestelde eisen wordt voldaan, of dat er situaties zijn waarin dit niet kan worden gegarandeerd. In tegenstelling tot testen, waar vaak slechts een eindig aantal situaties kan worden bekeken, rekent de computer bij formele verificatie automatisch alle mogelijke situaties in het model door. Bij deze manier van ontwikkelen kan iteratief het besturingsmodel worden aangepast totdat deze gegarandeerd aan alle gestelde eisen voldoet.

- **Synthese-gebaseerd ontwikkelen** (*Synthesis-Based Engineering* of *SBE* in het Engels) is een vorm van model-gebaseerd ontwikkelen, waarbij het ontwerp van de besturing wordt geautomatiseerd met behulp van computer-ondersteuning. Door het automatisch genereren (synthetiseren) van het besturingsmodel kan wiskundig worden gegarandeerd dat dit gesynthetiseerde besturingsmodel aan alle gestelde besturingseisen voldoet. Daarmee is verificatie van dit besturingsmodel tegen de besturingseisen overbodig, omdat deze gegarandeerd correct-door-constructie is vervaardigd. Door het automatiseren van het verkrijgen van het

besturingsmodel en de realisatie van de besturing, en het gedeeltelijk overbodig maken van de verificatie ervan, blijven voornamelijk het ontwerpen en valideren van de besturingseisen (en daarmee de besturing zelf) over. Hiermee kan de volledige aandacht komen te liggen op wat de besturing moet doen, in plaats van hoe die dat moet bewerkstelligen.

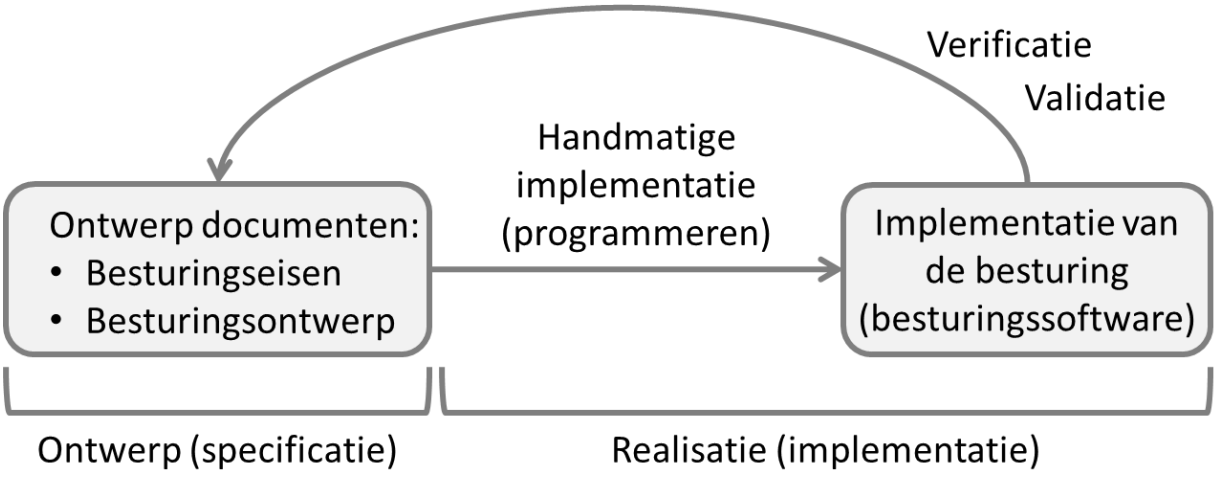
Het gebruik van model-gebaseerd ontwikkelen in combinatie met computer ondersteuning, heeft veel potentiële voordelen, waaronder het vervaardigen van eenduidige, complete, consistente en up-to-date specificaties, waarmee sneller en goedkoper besturingssoftware kan worden gerealiseerd met een hoge(re) kwaliteit. Hierbij zijn er echter wel diverse aandachtspunten, die in Hoofdstuk 7 uitgebreid worden besproken.

In bepaalde andere domeinen is het gebruik van modellen en de achter formele methoden liggende principes al lang gemeengoed. Zo wordt bij het ontwerp van complexe computerchips veelvuldig gebruikt gemaakt van modellen en automatisering met behulp van computers (*electronic design automation* in het Engels) [7], om bijvoorbeeld alternatieve ontwerpen door te rekenen of de computer automatisch ontwerpen te laten synthetiseren en optimaliseren [8]. Ook bij software ontwikkeling in bredere zin worden steeds meer modellen gebruikt, bijvoorbeeld UML modellen. Wat verder van software verwijderd is het gebruik van notenbalken voor muziekstukken, wat gezien kan worden als een vorm van gestandaardiseerde domein specifieke notatie. En ook bij de bouw van een huis zal een architect eerst bouwtekeningen maken, en daarbij gestandaardiseerde domein specifieke notaties gebruiken. Dit kan worden gezien als modelleren, en zorgt voor tekeningen die door alle uitvoerende partijen begrepen kunnen worden. Vervolgens zullen er, vaak gebruikmakend van computers, constructie-berekeningen worden uitgevoerd, alvorens het huis daadwerkelijk wordt gebouwd.

In de volgende secties worden de verschillende ontwikkelprocessen in meer detail besproken: traditioneel ontwikkelen in Sectie 5.3, model-gebaseerd ontwikkelen (MBE) in Sectie 5.4, verificatie-gebaseerd ontwikkelen (VBE) in Sectie 5.5 en synthese-gebaseerd ontwikkelen (SBE) in Sectie 5.6. Hierbij ligt de nadruk vooral op de nadelen van traditioneel ontwikkelen, en de voordelen die opeenvolgende alternatieve ontwerpprocessen bieden. Een diepere analyse van MBE, VBE en SBE, maar vooral SBE, met naast de potentiële voordelen ook ruime aandacht voor aandachtspunten wat betreft de introductie ervan, is te vinden in Hoofdstuk 7.

5.3 Traditioneel ontwikkelen

Figuur 2 toont een zeer versimpelde weergave van het ontwikkelproces voor traditionele ontwikkeling van besturingssoftware.



Figuur 2 – Zeer versimpelde weergave van het proces van traditioneel ontwikkelen.

Traditioneel worden besturingen eerst beschreven en gespecificeerd in documenten. Die beschrijven bijvoorbeeld de functionele en veiligheidseisen (*requirements*), en hoe voor elke toestand van het systeem de besturing moet reageren door bepaalde actuatoren aan of uit te zetten afhankelijk van de waardes van sensor signalen, en de veranderingen daarvan.

Vervolgens wordt de besturingssoftware handmatig geïmplementeerd met behulp van een programmeertaal, zoals bijvoorbeeld PLC-code voor een PLC-platform, of Java of C++ code voor een industriële PC.

Tenslotte wordt de implementatie gevalideerd en geverifieerd, meestal met behulp van testen. Verificatie behelst het controleren en uiteindelijk garanderen dat de besturing aan de gestelde eisen (requirements) voldoet. Validatie behelst het controleren dat de besturing het gewenste gedrag vertoont, en dus dat het de gewenste besturing is. Omdat een besturing aan de gespecificeerde eisen moet voldoen, bevat dit dus de validatie van de eisen, om te zorgen dat het de juiste en gewenste eisen zijn.

Ook Rijkswaterstaat werkt veelal op deze manier, waarbij de documenten dienen als opdracht en contract voor een externe partij om (de software voor) de besturing te ontwikkelen, en deze dan al dan niet samen met Rijkswaterstaat te verifiëren en valideren.

Traditioneel ontwikkelen bestaat, zoals de naam al aangeeft, al geruime tijd. Bedrijven weten vaak goed wat werkt en wat niet, en hoe om te gaan met de verschillende uitdagende aspecten ervan. Deze manier van werken kan prima geschikt zijn, zeker voor kleinere en simpelere systemen die worden ontwikkeld door een goed geleid en niet te groot team. Echter, er zijn ook nadelen bij deze manier van werken. Deze nadelen zijn fundamenteel aanwezig, maar nog meer van toepassing als het gaat om het ontwikkelen van besturingssoftware voor grotere en complexere systemen, als die worden ontwikkeld door meerdere teams, of als bepaalde ontwikkelactiviteiten worden uitbesteed aan marktpartijen.

Hieronder wordt verder ingegaan op een aantal van de nadelen van traditioneel ontwikkelen.

5.3.1 Ambiguïteit

Het eenduidig opschrijven van besturingseisen in een document is bijzonder lastig. Vaak zijn tekstuele beschrijvingen in natuurlijke taal voor meerdere interpretaties vatbaar.

Diegene die de eisen opstelt heeft daar een bepaald beeld bij. Een ander persoon, die de eisen vervolgens in software moet implementeren, kan deze anders interpreteren en vormt daarbij een eigen mentaal beeld. Hierbij is sprake van een kloof tussen de specificatie en de implementatie.

Documenten kunnen ook dienen als startpunt voor een leverancier of marktpartij om de besturingssoftware te ontwikkelen. Dan zijn de impact en kosten van ambiguïteit vaak groot, groter dan wanneer de implementatie in het eigen bedrijf wordt ontwikkeld, vanwege bijvoorbeeld de fysieke afstand, andere bedrijfscultuur en ander taalgebruik.

5.3.2 Incompleetheid en inconsistentie

Naast de interpretatie van besturingseisen is ook de compleetheid van besturingseisen een belangrijk punt om over na te denken. Vaak worden de in normale situaties voorkomende gevallen wel afgedekt (ook wel *happy flow* genoemd). Echter, de randgevallen en uitzonderingen zijn zeker zo belangrijk, gezien ook dat veiligheid voor Rijkswaterstaat een belangrijke rol speelt.

Denk bijvoorbeeld aan veiligheidseisen bij falende hardware, zoals een kapotte kabel of defecte sensor. Dit soort gevallen zijn vaak nog complexer en het aantal combinaties/interacties waarover moet worden nagedacht is vaak erg groot. De kans dat de beschrijving van al die gevallen tot inconsistenties (tegenspraken) leidt, is dan ook vaak niet uit te sluiten.

Een goede expert zal het aantal fouten (missende eisen en tegenstrijdigheden) in de specificaties weten te beperken, onder andere door goed na te denken, te overleggen, en gebruik te maken van reviews, maar fouten uitsluiten is vaak erg lastig. Een goede software/PLC-engineer zal zeker een aantal van de overgebleven fouten vinden tijdens de implementatie en het testen van de besturing.

Echter, gemiddeld bevat iedere 1.000 regels geteste en opgeleverde code in de industrie nog altijd 1 tot 25 fouten [9]. Bovendien zal, als de specificatie incompleet is, een software engineer hier vaak eigen keuzes in maken, die niet noodzakelijkerwijs overeen komen met het beeld dat de domeinexpert in gedachten had bij de specificatie. Ook hier kan het werken met marktpartijen deze problemen vergroten.

5.3.3 Multidisciplinaire systemen

De functionele compleetheid en consistentie van de besturingseisen en de interpretatie van natuurlijke taal zijn niet de enige aspecten die een kloof tussen specificatie en implementatie veroorzaken. Ook op het gebied van disciplines is er vaak een groot verschil. Zo zullen bijvoorbeeld domein experts die alles weten van bijvoorbeeld sluizen de functionele besturingseisen opschrijven. Echter, degene die dat in bijvoorbeeld PLC-code implementeert is vaak een software engineer die deze expertise niet heeft, of in minder mate. Ze komen van verschillende domeinen, gebruiken vaak andere technische termen, en spreken dus in feite verschillende talen. Dit maakt het lastiger om elkaar te begrijpen, en hindert daarmee goede communicatie.

5.3.4 Abstractieniveaus

Daarnaast is er ook een verschil in abstractie niveau tussen ontwerp en realisatie. De besturingseisen worden vaak als functionele specificaties opgeschreven. In de implementatie zijn er tal van details op een lager abstractieniveau, zoals data structuren, berichtcoderingen en byte volgordes. De functionele specificatie doet daarover doorgaans geen uitspraken. Ook dit maakt het lastiger voor personen met verschillende achtergronden om elkaar te begrijpen.

5.3.5 Mixen van ontwerp en implementatie aspecten

De situatie wordt echter nog complexer als er (onbedoeld) tijdens het ontwerp implementatie aspecten worden verwerkt in de functionele specificatie. De scherpe scheiding tussen ontwerp (specificatie) en realisatie (implementatie) is dan verloren. Dit leidt vaak tot misverstanden, en om die op te lossen is dan vaak meer communicatie en samenwerking vereist.

5.3.6 Verouderde documentatie

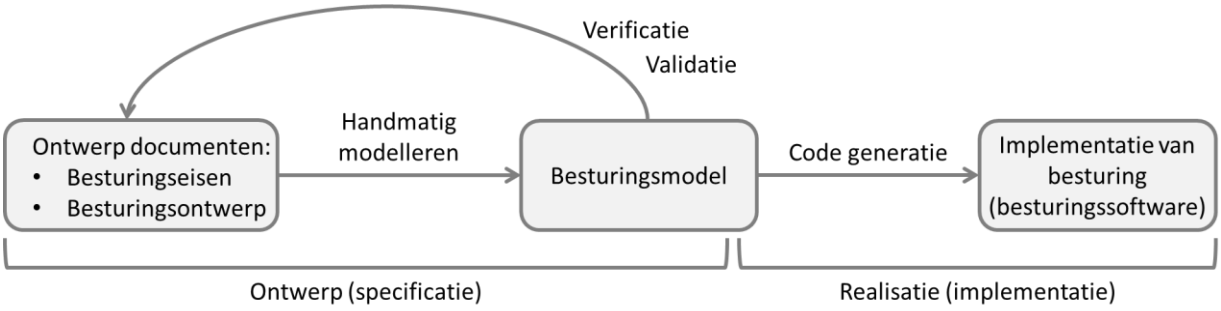
Een ander veel voorkomend probleem met specificatie in documenten is dat zodra de implementatie beschikbaar is, eventuele veranderingen, zoals correcties en nieuwe functionaliteit, alleen nog worden doorgevoerd in de implementatie. Na verloop van tijd is de documentatie meer en meer verouderd en dus onbruikbaar. Dit vergroot de kloof tussen de specificatie en de implementatie.

5.4 Model-gebaseerd ontwikkelen

Model-gebaseerd ontwikkelen, model-gebaseerd ontwerpen, modelmodel-gebaseerde software/systeem ontwikkelen en model-gedreven ontwikkelen (*model-based engineering* of *MBE*, *model-based design, model-based software/system engineering* en *model-driven engineering* in het Engels) zijn gerelateerde termen. Ze plaatsen modellen centraal gedurende het gehele ontwikkelproces en levensduur van het systeem, inclusief ontwerp, implementatie en onderhoud. De modellen vullen op een bepaalde wijze het gat tussen de specificatie en de implementatie.

5.4.1 MBE proces

Figuur 3 toont een zeer versimpelde weergave van het ontwikkelproces voor model-gebaseerd ontwikkelen (MBE) van besturingssoftware.



Figuur 3 – Zeer versimpelde weergave van het proces van model-gebaseerd ontwikkelen (MBE).

In het midden staat het besturingsmodel, een model van de besturing dat eenduidig vastlegt hoe de besturing werkt. Het geeft precies aan hoe de toestand van de besturing wijzigt als een sensorsignaal verandert, en onder welke condities en in welke toestanden een actuator aan of uitgezet mag worden. Idealiter heeft het model een wiskundige basis. Het kan bijvoorbeeld worden gemodelleerd als toestandsmachine.

Het besturingsmodel wordt handmatig gemodelleerd vanuit de ontwerp documenten. Die beschrijven bijvoorbeeld de functionele en veiligheidseisen waar de besturing aan moet voldoen, beschrijven de besturingstoestanden en geven aan wanneer een bepaalde actuator moet worden bedien afhankelijk van sensorsignalen. Die beschrijvingen worden dus handmatig omgezet in een eenduidig model.

Het besturingsmodel moet worden geverifieerd en gevalideerd, net als bij traditioneel ontwikkelen. Echter, bij MBE kan hierbij gebruik worden gemaakt van ondersteuning door formele methoden, methoden met een wiskundige basis, ondersteund door computerprogramma's. Een besturingsmodel kan bijvoorbeeld worden gesimuleerd om meer inzicht te krijgen in hoe het zich gedraagt in diverse situaties. Hierbij kunnen problemen naar boven komen, die dan kunnen worden opgelost om het besturingsmodel te verbeteren.

De besturingssoftware wordt typisch geïmplementeerd in een programmeertaal, zoals PLC-code voor een PLC-platform, of Java of C++ code voor een industriële PC. Dit kan door het bedrijf zelf worden gedaan, door een ander team of andere afdeling, maar kan ook worden uitbesteed aan een marktpartij. Hoewel handmatige implementatie mogelijk is, wordt de code vaak automatisch gegenereerd vanuit het besturingsmodel.

Het gebruik van model-gebaseerd ontwikkelen in combinatie met computer-ondersteund ontwerpen door het gebruik van formele methoden heeft veel voordelen ten opzichte van een traditionele manier van ontwikkelen, en adresseert direct een aantal van de nadelen van de traditionele manier van ontwikkelen. Het maakt het in potentie mogelijk om eenduidige, complete, consistente en up-to-date specificaties te maken, en daaruit besturingen te vervaardigen van hoge(re) kwaliteit tegen gelijke of zelfs lagere moeite en kosten. Een aantal van deze voordelen worden hieronder besproken. Verder kunnen specifiekere vormen van model-gebaseerd ontwikkelen, zoals verificatie-gebaseerd ontwikkelen (zie Sectie 5.5) en synthese-gebaseerd ontwikkelen (zie Sectie 5.6), extra voordelen bieden. En hoewel model-gebaseerd ontwikkelen veel potentiële voordelen heeft, is het significant anders dan een traditionele manier van werken, en moeten de aandachtspunten daarbij dus niet uit het oog worden verloren (zie Hoofdstuk 7).

5.4.2 Eenduidige en intuïtieve specificaties

Het is belangrijk hier gebruik te maken van formele modellen met een wiskundige betekenis (semantiek). Als voorbeeld kan gedacht worden aan bijvoorbeeld toestandsmachines voor het modelleren van besturingsmodellen en logische formules voor het modelleren van besturingseisen. Het gebruik van dergelijke formele modellen leidt tot een eenduidige interpretatie van de besturingseisen en van het gedrag van de besturing.

Het gebruik van de juiste formele modelleertaal, waarin op een intuïtieve en gestandaardiseerde manier de besturingseisen kunnen worden vastgelegd, is essentieel. Dit is het gebied van domein specifieke talen (*Domain Specific Languages, DSLs*, in het Engels). Een dergelijke taal past goed bij de belevingswereld van de domeinexpert, die daarmee rechtstreeks het systeem kan beschrijven in een notatie die naadloos aansluit bij hoe die persoon er over nadenkt. Voor Rijkswaterstaat kan bijvoorbeeld gedacht worden aan een DSL met concepten zoals onder andere sluizen, sluisdeuren, verkeerslichten, hefbomen en sensoren. Dit leidt tot goed leesbare en eenduidige specificaties. Voorbeelden uit andere domeinen zijn notenbalken voor muziekstukken, en bouwtekeningen voor de bouw van huizen.

Naast specifiek voor een bepaald domein, zijn domein specifieke talen ook beperkter in wat je er kan opschrijven dan een algemene modelleertaal. En juist daar ligt de kracht. Door de beperktere hoeveelheid concepten zijn er minder mogelijkheden om hetzelfde te modelleren. Dit leidt tot standaardisatie, meer consistentie en eenvoud.

5.4.3 Overbruggen van het multidisciplinaire specificatie/implementatie gat

Als een goede domein specifieke taal gebruikt wordt, kunnen zowel de domeinexpert als de software engineer deze begrijpen en op dezelfde manier interpreteren, ongeacht de verschillende achtergronden. Uiteraard moet er een taal gebruikt worden die voldoende rijk is om alle relevante aspecten van het domein te kunnen beschrijven. Het moet het juiste abstractieniveau gebruiken.

5.4.4 Complete en consistente specificaties

Het gebruik van eenduidige formele modellen heeft nog meer voordelen. Het maakt het mogelijk voor een computerprogramma om deze modellen te analyseren. De beperktere hoeveelheid concepten van een domein specifieke taal daarentegen helpen mee om dit efficiënt (schaalbaar) te kunnen doen. Computerprogramma's kunnen met formele methoden, dus met wiskundige technieken, razendsnel een grote hoeveelheid scenario's analyseren. Dit in tegenstelling tot de nu vaak gebruikelijke document reviews.

Een voorbeeld voor formele verificatie is model-gebaseerd testen. Hierbij kan een computer automatisch miljoenen of zelfs meer tests genereren uit het besturingsmodel, zowel voor wat de besturing wel als wat die niet mag toelaten. Dit zijn er aanzienlijk meer dan de tientallen, honderden of duizenden testen die vaak handmatig (kunnen) worden gemaakt. Dit maakt het mogelijk om meer gedrag te testen en verhoogt daarmee het vertrouwen in de correctheid van het besturingsmodel en daarmee van de implementatie, zeker als de testen automatisch worden uitgevoerd op het daadwerkelijke systeem.

Een ander voorbeeld is validatie van de specificatie door middel van simulatie. Met behulp van simulatie kunnen diverse besturingsscenario's bekeken worden, en kan het gedrag van de besturing inzichtelijk gemaakt worden. Dit geeft weer nieuwe inzichten, die gebruikt kunnen worden om de specificatie te verbeteren. Vooral voor complexe situaties, die moeilijk zijn om te doorgronden, is dit van grote meerwaarde.

Het gebruik van validatie en verificatie, geautomatiseerd via een computer, legt dikwijls problemen in de specificatie bloot. Model-gebaseerd testen kan bijvoorbeeld een scenario vinden dat nog niet was overwogen en meegenomen tijdens het modelleren van het besturingsmodel, waardoor het besturingsmodel bij het testen op het daadwerkelijke object voor dat scenario niet aan de eisen voldoet. Soortgelijk kunnen door simulatie nieuwe inzichten ontstaan en problemen in het besturingsmodel boven water komen. De specificatie en het ontwerp moeten hierop worden aangepast en wederom geverifieerd en gevalideerd worden. De specificatie en het ontwerp kunnen op deze manier iteratief worden verbeterd, wat leidt tot completere en consistentere specificaties en dus een besturing van hogere kwaliteit.

5.4.5 Vroegtijdig problemen oplossen en voorkomen

Het grote voordeel van model-gebaseerd ontwikkelen is dat verificatie en validatie al in een vroeg stadium plaatsvindt en niet pas in latere stadia, bij implementatie of testen. Het is welbekend in de industrie dat hoe

eerder een fout wordt gevonden en opgelost, hoe minder kosten hiermee gemoeid zijn [10]. In de praktijk blijkt dat implementaties gemaakt door middel van een model-gebaseerd ontwikkelingsproces veelal sneller gemaakt kunnen worden en minder fouten bevatten [11, 12]. Door automatisering kan een wijziging sneller worden doorgevoerd in de modellen, en automatisch opnieuw geanalyseerd worden.

Daarnaast moet ook het nut van de discussies die al vroeg in het ontwerpproces kunnen ontstaan, over hoe de specificatie moet worden aangepast indien deze niet voldoet, niet worden onderschat. Dat er in dit stadium eenduidig kan worden gesproken over besturingseisen en het gedrag van het systeem in onvoorziene omstandigheden, zoals in situaties met bijvoorbeeld defecte sensoren, is dan ook van grote waarde.

5.4.6 Efficiënt correct-door-constructie implementaties verkrijgen

Na diverse iteraties is het vertrouwen in een specificatie van de besturing voldoende groot, en dus de kans op inconsistenties en incompleetheid voldoende klein, gegeven de tijd en hoeveelheid geld die aan het ontwerpproces besteed kan worden. Het ontwerpproces eindigt met een implementatie onafhankelijke specificatie van de besturingslogica, die tijdens de realisatie kan worden geïmplementeerd in softwarecode. Dit kan worden gedaan door een ander team of een andere afdeling binnen het bedrijf, maar ook door een marktpartij. De formele specificaties kunnen dan dienen als een contract met die partij, wat zorgt voor meer controle. Ze kunnen ook worden gebruikt om acceptatietesten uit te voeren op de implementatie.

Hoewel de software handmatig kan worden geïmplementeerd, is automatische generatie van de besturingssoftware vaak een betere keuze. Zoals vaak bij automatisering voorkomt dit fouten die mensen maken als ze handmatig een implementatie maken, wat de consistentie tussen specificatie en implementatie ten goede komt. Daarnaast zorgt automatisering over het algemeen voor een verhoogde efficiëntie. Als het besturingsmodel is aangepast, kan met ‘een druk op de knop’ een nieuwe correct-door-constructie implementatie worden gegenereerd.

5.4.7 Implementatie-onafhankelijke modellen

Een ander voordeel van besturingsmodellen die de besturingslogica vastleggen, is dat dergelijke modellen platform- en leveranciersonafhankelijk kunnen zijn. Omdat een besturingsmodel implementatie-onafhankelijk is, is er een duidelijke scheiding tussen ontwerp (specificatie) en realisatie (implementatie). Dit maakt het mogelijk om code te genereren voor verschillende platformen, zoals industriële PC’s en PLC’s, met verschillende programmeertalen, zoals Java, C en PLC-code, voor zowel 32 bit als 64 bit architecturen, etc. Daarnaast zijn besturingsmodellen leveranciersonafhankelijk, wat het mogelijk maakt om PLC-code te genereren voor PLC-platformen van verschillende leveranciers. Het is zelfs mogelijk om later nog naar een andere taal, ander platform of een andere leverancier over te stappen, of code te genereren voor meerdere talen, platformen en leveranciers.

5.4.8 Up-to-date modellen

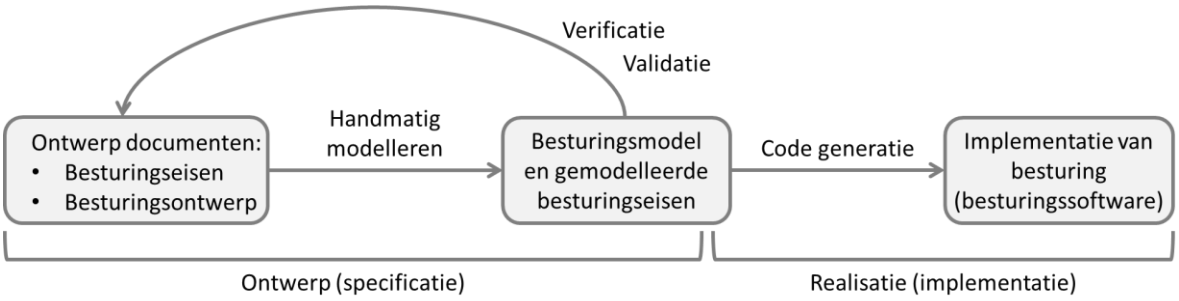
Model-gebaseerd ontwikkelen stelt modellen centraal. Het zijn de modellen die worden aangepast als deze functioneel onjuist zijn, als er inconsistenties inzitten, of als er nieuwe functionaliteit gewenst is. Analyses, synthese en codegeneratie werken op de modellen. In tegenstelling tot documentatie, worden de modellen dan vaker onderhouden en up-to-date gehouden, omdat ze aan de basis staan van de ontwikkeling gedurende de gehele levenscyclus van het systeem, zoals ontwerp, realisatie en onderhoud. Dit kan mogelijk wel extra inspanning vereisen, is mogelijk geen volledige vervanging van documentatie, en de ontwerpkeuzes moeten nog altijd worden vastgelegd.

5.5 Verificatie-gebaseerd ontwikkelen

Verificatie-gebaseerd ontwikkelen (*verification-based engineering* of *VBE* in het Engels) is een vorm van model-gebaseerd ontwikkelen. Het maakt gebruik van formele verificatie om geautomatiseerd aan te tonen dat het besturingsmodel aan de gestelde besturingseisen voldoet.

5.5.1 VBE proces

Het zeer versimpelde proces van verificatie-gebaseerd ontwikkelen, zoals weergegeven in Figuur 4, is nagenoeg hetzelfde als voor model-gebaseerd ontwikkelen, zoals weergegeven in Figuur 3.



Figuur 4 – Zeer versimpelde weergave van het proces van verificatie-gebaseerd ontwikkelen (VBE).

Het voornaamste verschil is de manier waarop de verificatie van het besturingsmodel ten opzichte van de gespecificeerde besturingseisen wordt uitgevoerd. Verificatie-gebaseerd ontwikkelen gebruikt daarvoor formele verificatie, zoals *model checking*, om wiskundig te bewijzen dat een bepaalde eigenschap geldt. Zulke eigenschappen kunnen veiligheidseisen zijn, bijvoorbeeld dat een brug alleen open mag gaan als het corresponderende verkeerslicht voor landverkeer op rood is gezet. Maar het is bijvoorbeeld ook mogelijk om te controleren op de afwezigheid van *deadlock*, een toestand waar je niet meer uit kunt, of *livelock*, waarbij er wel transities mogelijk zijn, maar daarbij geen echt zinnige voortgang wordt geboekt, bijvoorbeeld omdat telkens dezelfde effectloze lokale loop wordt doorlopen. Formele verificatie kan bewijzen dat dergelijke eigenschappen gelden voor elk scenario dat volgens het besturingsmodel mogelijk is.

Als een eigenschap niet geldt, dan produceert formele verificatie een tegenvoorbeeld, typisch in de form van een reeks gebeurtenissen en hoe de besturing daar op reageert, die leiden tot een toestand in het besturingsmodel waar de eigenschap niet geldt. Dit maakt het mogelijk om precies vast te stellen waar het probleem zit in het model, en dat te adresseren. De specificatie en het ontwerp moeten dan worden aangepast en wederom geverifieerd worden. De specificatie kan op deze manier iteratief worden verbeterd, door telkens problemen te adresseren, opnieuw te verifiëren, de daar uit volgende problemen weer te adresseren, weer te verifiëren, etc. Zodra verificatie geen tegenvoorbeeld meer produceert zijn alle eigenschappen geverifieerd en is het gegarandeerd dat deze eigenschappen in alle situaties gelden in het besturingsmodel.

Om formele verificatie toe te passen moet niet alleen het besturingsmodel formeel worden gespecificeerd, maar ook de te verifiëren eigenschappen. Dit betekent dat de besturingseisen niet langer (alleen) in natuurlijke taal in documenten worden beschreven, maar (ook) in wiskundig eenduidige specificaties worden gemodelleerd. Een voorbeeld is een toestandsmachine die de volgorde van bepaalde acties vastlegt, zoals dat een bepaalde sensor aan moet gaan voordat een bepaalde actuator mag worden geactiveerd. Een ander voorbeeld is een logische formule die aangeeft dat een bepaalde combinatie van toestanden in het besturingsmodel nooit mag voorkomen, omdat die bijvoorbeeld een botsing aangeven die voorkomen moet worden.

5.5.2 Voordelen van VBE

Verificatie-gebaseerd ontwikkelen heeft al de voordelen van model-gebaseerd ontwikkelen (zie Sectie 5.4). Daarnaast heeft het ook nog het extra voordeel dat succesvolle formele verificatie garandeert dat in het gemodelleerde besturingsmodel aan de besturingseisen wordt voldaan. Formele verificatie beschouwt namelijk elk mogelijk scenario van dat model. Het kan daarom wiskundig bewijzen dat een gespecificeerde besturingseis geldt in het besturingsmodel. Het is daarmee in deze context krachtiger dan testen, waarbij praktisch altijd slechts een eindig aantal scenario's kan worden gecontroleerd, en dat daarom niet uitputtend alle scenario's controleert.

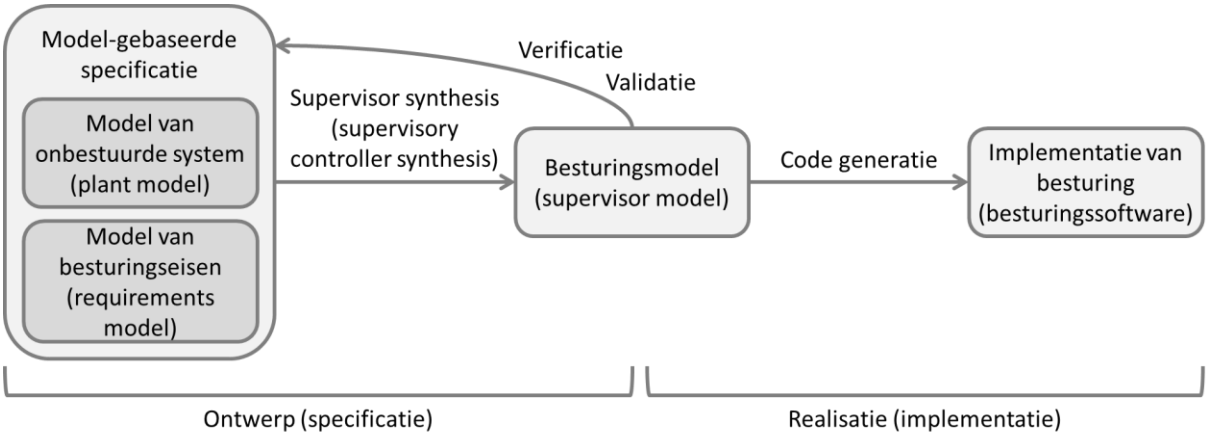
Ten opzichte van model-gebaseerd ontwikkelen geeft dit dus meer garanties, en kan het daarmee zorgen voor nog completere en consistentere specificaties en dus besturingen van hoge(re) kwaliteit. Echter, net als model-gebaseerd ontwikkelen moeten hierbij ook voor verificatie-gebaseerd ontwikkelen de aandachtspunten niet uit het oog worden verloren (zie Hoofdstuk 7). Verder kan synthese-gebaseerd ontwikkelen nog extra voordelen bieden (zie Sectie 5.6).

5.6 Synthese-gebaseerd ontwikkelen

Synthese-gebaseerd ontwikkelen (*Synthesis-Based Engineering* of *SBE* in het Engels) is een vorm van model-gebaseerd ontwikkelen gecombineerd met computer-ondersteund ontwerpen. Het heeft als uitgangspunt om correct-door-constructie besturingen te vervaardigen, door zo veel mogelijk stappen in het ontwikkelproces te automatiseren.

5.6.1 SBE proces

Figuur 5 geeft een zeer versimpelde weergave van het SBE proces weer.



Figuur 5 – Zeer versimpelde weergave van het proces van synthese-gebaseerd ontwikkelen (SBE).

Net als bij model-gebaseerd ontwikkelen staat het besturingsmodel met een wiskundige basis centraal. Vanuit het besturingsmodel wordt nog altijd de besturingssoftware automatisch gegenereerd, alhoewel ook hier een handmatige implementatie mogelijk is.

Echter, bij SBE wordt het besturingsmodel niet handmatig gemodelleerd vanuit de ontwerp documenten. In plaats daarvan wordt het automatisch gegenereerd uit modellen van het onbestuurde systeem (*plant* model) en de besturingseisen (*requirements* model).

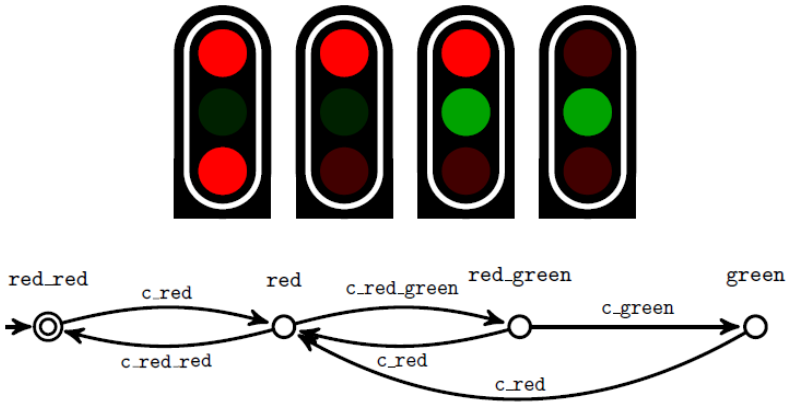
Verificatie om te garanderen dat het besturingsmodel voldoet aan de gestelde besturingseisen is daarmee overbodig, omdat het gesynthetiseerde besturingsmodel correct-door-constructie is. Synthese garandeert

dus bepaalde eigenschappen voor het system als het wordt bestuurd door de gegenereerde besturing, waaronder dat aan de gemodelleerde besturingseisen wordt voldaan en dat het systeem niet blokkeert. Als nog andere eigenschappen gewenst zijn, zoals sterkere voortgangsgaranties, kunnen deze alsnog met formele verificatie worden bewezen. Verder moet het besturingsmodel ook hier nog worden gevalideerd om te zorgen dat het zich gedraagt zoals is bedoeld. De gespecificeerde besturingseisen kunnen namelijk ook bij SBE verkeerd zijn gemodelleerd, of bijvoorbeeld te strikt zijn, waardoor het systeem ongewenst verdrag vertoont of niet alle gewenste functionaliteit kan vertonen.

5.6.2 Plant en requirement modellen

Een essentieel onderdeel van SBE is besturingssynthese (*supervisory controller synthesis* of *supervisor synthesis* genaamd in het Engels). Dit is een techniek die zijn oorsprong heeft in de discrete regeltechniek van de jaren ’80 [13, 14]. Het idee van synthese is dat een model van de besturing automatisch kan worden gegenereerd door een computerprogramma. Hiervoor zijn twee modellen nodig. Het eerste type model wordt het plant[1] model genoemd in *control theory*. Plant modellen beschrijven het mogelijke gedrag van het te besturen systeem, zonder dat het bestuurd wordt door een besturing. Het representeert daarmee het mogelijke gedrag van het onbestuurde systeem, zoals een brug, sluis of tunnel. Het tweede type model wordt een *requirements* model genoemd. Het legt de besturingseisen vast waar de besturing aan moet voldoen. Het legt restricties op aan het gedrag van de plant, zodat alleen het gewenste gedrag overblijft.

Een plant model kan bijvoorbeeld vastleggen welke sensoren en actuatoren er zijn in het systeem, en welke eventuele relaties ertussen bestaan. Bijvoorbeeld dat de sensor die aangeeft dat een brug dicht is en een sensor die aangeeft dat een brug open is, onder normale omstandigheden niet tegelijk actief kunnen zijn. Een plant model wordt vaak gemodelleerd als toestandsmachine. Figuur 6 toont als voorbeeld een sluis verkeerslicht (actuator) [15] met het bijbehorende plant model als toestandsmachine. Initieel is het verkeerslicht rood/rood (*red_red*). Het kan dan enkel rood worden (*red*) via actie *c_red*. Door middel van de overige acties kunnen ook de andere toestanden worden bereikt. Een toestandsmachine kan maar in een enkele toestand tegelijk zijn, en het verkeerslicht kan dus niet tegelijk en enkel groen en enkel rood zijn. In dit geval is er dus gekozen om de drie actuatoren voor de twee rode lampen en de enkele groene lamp samen te nemen tot een enkele actuator, en deze als een enkele plant te modelleren, om het abstractieniveau iets te verhogen. Het is uiteraard ook mogelijk de drie lampen apart te modelleren als ieder een toestandsmachine van twee toestanden, waarbij de lampen onafhankelijk aan- en uitgezet kunnen worden.



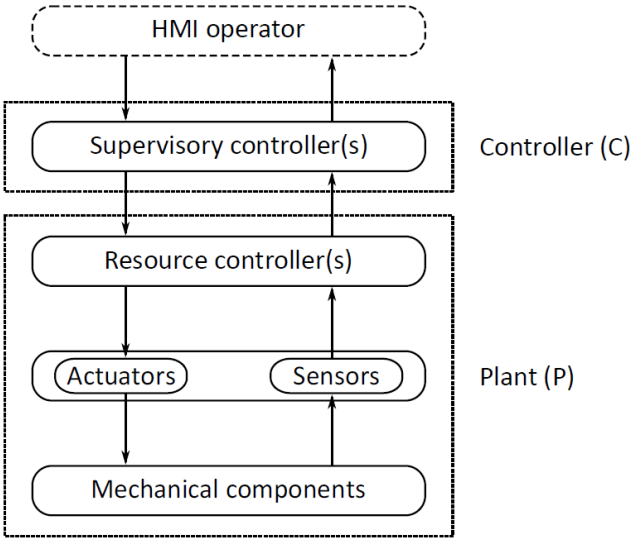
Figuur 6 – Voorbeeld verkeerslicht actuator, als grafische weergave (boven) en plant model (onder).

[1] Het woord ‘plant’ wordt hier niet in de Nederlandse betekenis van ‘flora’ gebruikt, maar naar de betekenis van het Engelse woord ‘plant’ in de zin van “de grond, gebouwen, machines, apparaten en werktuigen die worden gebruikt bij het uitoefenen van een handels- of industrieel bedrijf” [102].

Acties, zoals het aan of uit gaan van een sensor, en het activeren of deactiveren van een actuator, worden *events* genoemd. Besturingssynthese onderscheidt twee soorten events, *controllable* events en *uncontrollable* events. Controllable events kunnen worden geïnitieerd door de besturing. Het activeren en deactiveren van actuatoren worden typisch gemodelleerd als controllable events, zodat de besturing kan beslissen wanneer de activatie of deactivatie plaats vindt. Uncontrollable events treden vanuit het perspectief van de besturing autonoom op. Een besturing kan niet voorkomen dat dergelijke events optreden in het systeem. Een gebruiker kan bijvoorbeeld op een knop drukken, en de bijbehorende sensor zal dan aangeven of de knop is ingedrukt of niet. De events die aangeven dat de status van de sensor is veranderd zullen optreden. De besturing kan dat niet voorkomen. Een ander voorbeeld van uncontrollable events zijn de limietsensoren van een beweging. Zodra de beweging is voltooid en dus de eindpositie is bereikt, zal de limietsensor dit aangeven, wat leidt tot het optreden van een uncontrollable event dat de sensor van waarde is veranderd.

Limietsensoren zijn doorgaans uitgevoerd als ‘verbreker’, zodat het uitvallen van voeding leidt tot het stoppen van de beweging, net als bij het bereiken van de eindpositie. Dit betekent dat een dergelijke sensor doorgaans een ‘hoog’ of ‘aan’ signaal geeft, en dat bij het detecteren van een object of het wegvallen van stroom de sensor dan een ‘laag’ of ‘uit’ signaal geeft. Er zijn zowel sensoren die normaal gesproken een ‘hoog’ signaal geven en sensoren die normaal gesproken een ‘laag’ signaal geven. Dit is een punt van aandacht bij het definiëren van de bijbehorende events.

Hoewel plant modellen op het relatief lage abstractieniveau van sensoren en actuatoren zeer gebruikelijk zijn, kan er ook op een hoger abstractieniveau worden gemodelleerd en bestuurd. Besturingen op alle lagen in de architectuur van een systeem zijn mogelijk, waaronder bijvoorbeeld een hoger niveau besturing die meerdere lagere niveau besturingen aanstuurt en coördineert. Figuur 7 toont een veel gebruikte architectuur voor supervisory controllers [16, 17] op een relatief laag abstractieniveau.



Figuur 7 – Supervisory controller architectuur.

Het is uiteindelijk belangrijk alle voor de besturing relevante aspecten van het systeem en de omgeving van het systeem mee te nemen in het plant model. Vaak wordt gebruik gemaakt van abstractie, waarbij de irrelevante aspecten worden weggelaten of eenvoudiger worden gerepresenteerd, om zaken simpel te houden en toch genoeg detail te hebben om de besturing te kunnen ontwikkelen.

Een requirements model legt de functionele en veiligheidseisen voor de besturing vast. Bijvoorbeeld dat de motor om de brug open te maken pas aangezet mag worden als de slagboom om het wegverkeer tegen te houden helemaal dicht is. Besturingseisen kunnen worden gespecificeerd door middel van

toestandsmachines, maar vaak is het gebruik van een logische formule intuïtiever. Combinaties zijn ook mogelijk. Goed gespecifieerde logische formules zijn eenvoudig te begrijpen, ook voor personen zonder een wiskundige achtergrond. Figuur 8 toont een voorbeeld [18] van een requirement in drie vormen: natuurlijke taal, als wiskundige formule (logische expressie) en gemodelleerd in een domein-specifieke modelleertaal. Als modelleertaal is gebruik gemaakt van CIF, onderdeel van de Eclipse ESCET toolkit (zie ook Secties 6.2.7 en 7.11.1). Er is in dit specifieke geval gekozen om gebruik te maken van een limietsensor die normaal gesproken 'uit' is, wat zoals hiervoor besproken ongebruikelijk is vanuit veiligheidsperspectief.

Tekstuele beschrijving:

"De opwaartse actuator mag alleen aan gaan wanneer de slagboom niet open staat."

Logische expressie:

$$\texttt{actuator\_op.c\_aan} \implies \neg\texttt{sensor\_open.aan}$$

¬ = negatie teken
actuator_op = actuator_opwaarts
sensor_open = sensor_opendetectie

CIF code:

```
requirement actuator_op.c_aan needs not sensor_open.aan;
```

Figuur 8 – Voorbeeld van een requirement in drie vormen: tekst (boven), formule (midden) en model (onder) [18].

5.6.3 Besturingssynthese

Besturingssynthese (of simpelweg synthese) genereert uit een plant en een requirements model een besturingsmodel, een model van de besturingslogica, genaamd een *supervisor* of *supervisory controller*. Het gesynthetiseerde besturingsmodel is correct-door-constructie. Het voldoet dus in alle situaties aan alle gemodelleerde besturingseisen.

Synthese wordt typisch toegepast voor *discrete event systemen*, waarvoor het gedrag kan worden gemodelleerd als toestandsmachines. Het gaat dan voornamelijk over events, en de volgorde waarin die events kunnen en mogen plaatsvinden. Het besturingsmodel kan eveneens worden gerepresenteerd als een toestandsmachine, maar ook bijvoorbeeld een lijst condities waaronder actuator events geactiveerd mogen worden. Het gesynthetiseerde besturingsmodel als een toestandsmachine, of het plant model samen met de supervisor in de vorm van extra gesynthetiseerde besturingscondities, vormt het *bestuurde systeem (controlled system* in het Engels*)*.

Supervisor synthese garandeert vanuit de *supervisory control* theorie formeel vier eigenschappen:

- **Safety**: De gegenereerde supervisor voldoet aan alle gedefinieerde eisen (*requirements*). Deze supervisory control theorie notie van *safety* bekijkt veiligheid puur vanuit de besturingseisen, die zowel functioneel als veiligheidsgericht kunnen zijn. Hierin verschilt het van de notie van veiligheid zoals die in bijvoorbeeld de machinerichtlijn wordt bedoeld, waar alleen de veiligheidsgerichte eisen worden beschouwd, en er meer aspecten bij komen kijken om een object veilig te beschouwen (zie ook Sectie 7.6.4).
- **Controllability**: Hiervoor worden alleen bestuurbare acties (*controllable events* zoals gerelateerd aan actuatoren) beperkt toegelaten.
- **Non-blockingness**: Het bestuurde systeem blokkeert niet. Denk hierbij aan het ontbreken van livelock en deadlock.

- **Maximal permissiveness**: Synthese legt niet meer restricties op dan noodzakelijk is om de voorgaande drie eigenschappen te garanderen, en laat dus zoveel mogelijk gedrag toe. Dit wordt ook wel *minimally restrictive* genoemd.

Voor een uitgebreidere wetenschappelijke onderbouwing van de theorie achter synthese, en de precieze betekenis en definities van deze begrippen, zie [19, 20].

In de praktijk worden de termen *supervisor* en *besturing* (*controller* in het Engels) vaak door elkaar gebruikt. Formeel is er echter een verschil. Een *supervisor* laat maximaal gedrag toe en staat daarmee mogelijk meerdere (veilige) keuzes toe, bijvoorbeeld tussen het activeren van verschillende actuatoren, of tussen het activeren van de ene actuator en het deactiveren van een andere actuator. Een *controller* daarentegen kiest expliciet een bepaald controllable event in plaats van er meerdere toe te staan. Meer informatie hierover is te vinden in de literatuur [17] en in Sectie 7.2.3.

Synthese-gebaseerd ontwikkelen heeft al de voordelen van model-gebaseerd ontwikkelen (zie Sectie 5.4). Daarnaast geeft het modelleren via plant en requirements modellen en het gebruik van besturingssynthese een aantal extra potentiële voordelen. Die worden hieronder besproken. Echter, net als model-gebaseerd ontwikkelen moeten ook voor synthese-gebaseerd ontwikkelen de aandachtspunten niet uit het oog worden verloren (zie Hoofdstuk 7).

5.6.4 Hoge(re) kwaliteit tegen lagere kosten

Computer-ondersteund ontwerpen en automatisering verkorten de ontwerpcyclus en reduceren menselijke fouten. Hiermee kan in potentie de kwaliteit en betrouwbaarheid van besturingen worden verbeterd, en kunnen de inspanning en kosten om de besturing te realiseren worden gereduceerd.

Meer concreet biedt besturingssynthese computer ondersteuning voor het specificeren en ontwerpen van besturingssoftware. Het kan bijvoorbeeld automatisch conflicterende besturingseisen detecteren. Het detecteert ook dat een bepaalde actuator in een bepaalde toestand niet mag worden geactiveerd, omdat dit onder bepaalde condities later onoverkomelijk tot een onveilige situatie zal leiden, en levert een besturingsmodel op dat deze situaties voorkomt. Voor complexe systemen kan dit soort situaties vaak lastig te overzien zijn voor mensen. Het is daarmee lastig om hiervoor handmatig een correct besturingsmodel te maken.

5.6.5 Focus op ‘wat’ in plaats van ‘hoe’

Door gebruik te maken van supervisor synthese kan met ‘een druk op de knop’ uit plant en requirements modellen het besturingsmodel worden gesynthetiseerd. Vandaaruit kan ook de besturingssoftware automatisch worden gegenereerd door middel van codegeneratie. Verificatie is voor een deel overbodig omdat het besturingsmodel al correct-door-constructie is ten opzichte van de besturingseisen, en vanwege bepaalde andere garanties en eigenschappen van synthese. De focus voor het ontwikkelen van besturingssoftware voor een object ligt daarom dus op de resterende ontwerpstappen, het specificeren van de plant en requirements modellen, en de validatie daarvan. Hiermee kan er worden gefocust op ‘*wat* moet de besturing doen’ (de besturingseisen) in plaats van op ‘*hoe* moet de besturing (model dan wel software) dat bewerkstellingen’.

Synthese doorziet bijvoorbeeld zelf dat het een bepaalde actuator in een bepaalde toestand niet mag activeren, omdat dit onder specifieke condities later tot een onveilige toestand zal leiden. Dit soort situaties is voor complexe besturingen vaak lastig te doorzien voor mensen, en dus lastig correct te krijgen bij het handmatig modelleren van de besturing. Het specificeren van een *First In - First Out* (FIFO) eis kan bijvoorbeeld eenvoudig zijn, terwijl de realisatie van die eis complex kan worden door de vele mogelijkheden die kunnen optreden in het (deel)systeem [21]. Met supervisor synthese kunnen alle mogelijke combinaties van condities worden doorgerekend en kan een wiskundig correcte besturingslogica worden gegenereerd voor al deze situaties. Dit soort automatisering van het ontwerp is nog waardevoller als er meerdere, complexe en gerelateerde besturingseisen moeten worden beschouwd. De

gesynthetiseerde supervisor is correct-door-constructie voor al die besturingseisen in alle situaties, en dit voorkomt menselijke fouten.

5.6.6 Oplossingen in plaats van problemen

SBE gaat ver voorbij VBE (zie Sectie 5.5). Formele verificatie, zeker bij *model checking*, geeft problemen aan. Het vertelt je dat het besturingsmodel niet correct is en in welke situaties, maar je moet zelf steeds iteratief het model aanpassen. Elke keer dat verificatie een tegenvoorbeeld produceert om aan te geven dat een besturingseis niet gegarandeerd wordt, moet het besturingsmodel handmatig worden aangepast.

SBE daarentegen geeft oplossingen. Het synthetiseert automatisch een besturingsmodel waarin alle besturingseisen worden gegarandeerd. Synthese produceert dus in één keer een besturingsmodel met alle extra condities die moeten worden afgedwongen om te zorgen dat in alle situaties aan alle gestelde besturingseisen wordt voldaan. Dit maakt formele verificatie van het besturingsmodel tegen de besturingseisen overbodig, omdat het besturingsmodel al correct-door-constructie is.

Wel is de benodigde computerrekenkracht voor synthese doorgaans groter dan bij verificatie. Verificatie hoeft namelijk alleen aan te tonen dat een bepaalde besturingseis in bepaalde situaties niet geldt, terwijl synthese daarnaast ook nog de juiste extra condities moet uitrekenen om die ongewenste situaties te voorkomen.

5.6.7 Behoudt maximale ontwerpvrijheid

Synthese garandeert dat het gesynthetiseerde besturingsmodel *maximally permissive* is. Dit zorgt ervoor dat de maximale ontwerpvrijheid behouden wordt. Als handmatig een besturingsmodel wordt gemaakt, kan een ontwerper simpele besturingscondities gebruiken die de gedragsruimte sterk verkleinen. Synthese legt echter minimale restricties op, en behoudt dus maximale vrijheid, terwijl wel aan alle gemodelleerde besturingseisen wordt voldaan. De maximale vrijheid maakt het bijvoorbeeld mogelijk om uit de maximally permissive supervisor een tijd-optimale *controller* af te leiden, door te kiezen uit de veilige alternatieven.

5.6.8 Ondersteunt modulariteit en incrementeel werken

Elk deel van de plant en iedere besturingseis kan apart worden gespecificeerd. Zo wordt het eenvoudig specifieke plants en besturingseisen aan te passen, of toe te voegen. Modulaire specificaties maken dus een efficiënte incrementele manier van werken mogelijk, omdat met ‘een druk op de knop’ een nieuw correct-door-constructie besturingsmodel en de bijbehorende besturingssoftware kan worden gegenereerd.

5.6.9 Ondersteunt hergebruik en standaardisatie

De apart gespecificeerde plants en requirements kunnen in een bibliotheek met herbruikbare gestandaardiseerde ‘bouwblokken’ worden gestopt (zie ook Sectie 7.4). Dit stelt modelleurs in staat om eenvoudig nieuwe specificaties op te bouwen uit bestaande en bewezen kwalitatief goede bouwblokken, door ze op verschillende manier te combineren. Uiteindelijk leidt dit tot meer uniformiteit en verbetert dit de efficiëntie, omdat niet steeds het wiel opnieuw uitgevonden hoeft te worden, waarmee voorkomen kan worden dat ook steeds (door verschillende personen) dezelfde fouten worden gemaakt.

5.6.10 Intuïtieve specificaties en gedetailleerde traceerbaarheid

Elke apart gespecificeerde plant en requirement heeft een duidelijk doel. Dit geeft een goed overzicht van de besturingseisen, en geeft gedetailleerde traceerbaarheid van besturingseisen in de modellen. Dit in tegenstelling tot veel handmatig gemaakte besturingsmodellen, waar vaak een enkele eis op diverse manieren tot uitdrukking komt, omdat deze invloed kan hebben op diverse delen (toestanden) van de besturing en dus verspreid kan zitten in het besturingsmodel (en besturingssoftware), of gemixt kan zijn

met andere besturingseisen. Duidelijke modulaire besturingseisen voorkomen dat ongewenst gedrag verstopt zit in een groot en complex besturingsmodel.

5.7 Begrippen

Hier volgt een samenvatting van een aantal belangrijke begrippen, die veelvuldig terug zullen keren in de rest van dit rapport:

- **Modelgebaseerd ontwerpen/ontwikkelen (MBE)**: Het centraal stellen van modellen gedurende het gehele ontwikkelproces en de gehele levenscyclus van een systeem, waaronder tijdens ontwerp, realisatie en onderhoud.

- **Model**: Een representatie van relevante concepten op een eenduidige manier, idealiter met een wiskundige onderbouwing. Bijvoorbeeld een model van de besturingseisen in de vorm van logische formules of een model van het gedrag van een besturing in de vorm van een toestandsmachine.

- **Modelleertaal**: Een taal waarin modellen kunnen worden gespecificeerd, op een eenduidige manier en idealiter met een wiskundig onderbouwde betekenis (semantiek).

- **Domein specifieke taal**: Een modelleertaal met specifieke concepten voor een bepaald domein, bijvoorbeeld voor het domein van besturingen of het domein van bruggen, sluizen en tunnels.

- **Onbestuurd systeem / plant**: Het onbestuurde systeem (*uncontrolled system* in het Engels) is het systeem zoals het is, zonder de besturing. Het wordt ook wel de *plant* genoemd in *control theory*. Op een laag abstractieniveau kan het bijvoorbeeld bestaan uit de individuele sensoren en actuatoren van een brug, sluis of tunnel, en op een hoger abstractieniveau uit een aantal besturingen van deelsystemen.

- **Besturingseisen**: De eisen die gesteld worden aan de besturing, ten opzichte van het onbestuurde systeem. Ook wel *requirements* genoemd in *control theory*. Bijvoorbeeld de functionele en veiligheidseisen.

- **Besturingsmodel**: Een model van de besturing, dat eenduidig vastlegt hoe de gehele besturing werkt. Ook wel *controller model* genoemd. Legt bijvoorbeeld precies vast hoe de toestand verandert als een sensorsignaal verandert en onder welke condities in welke toestand een actuator zoals een motor wordt aangezet. In tegenstelling tot een supervisor model maakt het dus expliciete keuzes. Als dit onderscheid niet relevant is worden ‘besturingsmodel’ en ‘supervisor model’ in de praktijk vaak door elkaar gebruikt.

- **Besturingssoftware**: De implementatie van de besturing in software. Bijvoorbeeld PLC-code voor een PLC-platform, of bijvoorbeeld Java of C++ code voor een industriële PC.

- **Formele methode**: Een methode met een wiskundige basis en wiskundige garanties, doorgaans geautomatiseerd in computer tools. Bijvoorbeeld formele verificatie of supervisor synthese.

- **Correct-door-constructie formele methode**: Een formele methode die bepaalde garanties biedt, zoals de garantie dat het resultaat van de methode aan alle gestelde besturingseisen voldoet, of dat deze niet in een blokkerende toestand terecht kan komen. Dit maakt dan verificatie van het resultaat ten opzichte van die garantie, bijvoorbeeld de gestelde eisen, overbodig. Verificatie van andere eigenschappen kan alsnog benodigd zijn. Ook validatie van het resultaat is doorgaans alsnog nodig, bijvoorbeeld om te controleren dat de gestelde eisen ook daadwerkelijk de gewenste eisen zijn.

- **Validatie**: Het controleren dat een besturing of besturingsmodel het gewenste gedrag bewerkstelligt, en dus dat de besturing de gewenste besturing is. Aangezien een besturing aan de gestelde eisen moet voldoen, bevat dit ook het controleren of de besturingseisen de gewenste eisen zijn.

- **Verificatie**: Het controleren of een besturing of besturingsmodel aan bepaalde eigenschappen voldoet, zoals bijvoorbeeld het voldoen aan de gestelde besturingseisen of het ontbreken van situaties waarin de besturing geblokkeerd raakt.

- **Synthese-gebaseerd ontwikkelen (SBE)**: Een manier om besturingssoftware te ontwikkelen waarbij model-gebaseerd ontwikkelen wordt gecombineerd met computer-ondersteund ontwerpen, om zoveel mogelijk stappen van het ontwikkelproces te automatiseren, waaronder door het gebruik van supervisor synthese.

- **Supervisor synthese**: Een correct-door-constructie formele methode voor het automatisch synthetiseren (genereren) van een correct-door-constructie supervisor model uit een model van het onbestuurde systeem en een model van de besturingseisen. Ook wel (*besturings)synthese*, *supervisor synthese* of *supervisory controller synthesis* genoemd. Als hiervoor wordt gekozen, dan maakt dit verificatie ten opzichte van de opgestelde besturingseisen overbodig. Verder biedt het een aantal andere garanties, waaronder het afwezig zijn van blokkerende toestanden.

- **Event**: Een actie die iets representeert dat in het systeem kan gebeuren. Bijvoorbeeld acties op een laag abstractieniveau voor het aangaan en uitgaan van sensoren of het aanzetten en uitzetten van actuatoren, of acties op een hoger abstractieniveau zoals commando's om een object van de ene naar de andere positie te laten bewegen of die aangeven dat er fout is opgetreden in een deelsysteem.

- **Controllable event**: Een event dat wordt geïnitieerd (*controlled* in het Engels) door de besturing. Events om een actuator te besturen (aan of uit te zetten) zijn doorgaans controllable events.

- **Uncontrollable event**: Een event dat autonoom optreedt, vanuit het perspectief van de besturing, en daarmee dus niet wordt geïnitieerd of kan worden tegengehouden door de besturing. Events die aangeven dat een knop is ingedrukt of is losgelaten zijn doorgaans uncontrollable events, net als events van limietsensoren die aangeven dat een beweging bij de eindpositie is aangekomen of dat een bewegend object zich niet langer op een bepaalde positie bevindt.

- **Supervisor (model)**: Een maximaal permissief besturingsmodel, zoals dat door supervisor synthese wordt gegenereerd. In tegenstelling tot een besturingsmodel bevat het nog de maximale vrijheid om uit het veilige gedrag keuzes te maken, en door voor alle keuzes een keuze te maken kan er een besturingsmodel worden verkregen. In de praktijk worden 'supervisor model' en 'besturingsmodel' vaak door elkaar heen gebruikt, zeker als het verschil ertussen niet van belang is.

- **Bestuurd systeem**: Het bestuurde systeem (*controlled system* in het Engels) is het onbestuurde systeem dat wordt bestuurd door een supervisor of besturing (*controller* in het Engels).

- **Codegeneratie**: Het automatisch genereren van correct-door-constructie besturingssoftware uit een besturingsmodel.

6 Introductie SBE bij Rijkswaterstaat

In dit hoofdstuk wordt dieper ingegaan op de introductie van SBE bij Rijkswaterstaat. Eerst wordt kort de visie van Rijkswaterstaat met betrekking tot SBE beschreven (Sectie 6.1). Vervolgens worden de concrete plannen en al gemaakte keuzes voor de introductie van SBE, en daarmee de realisatie van de visie, benoemd (Sectie 6.2). Daarna wordt ingegaan op de zorgen en twijfels die nog bestaan binnen de organisatie, zoals die tot uitdrukking komen in de aan TNO gestelde vragen (Sectie 6.3). Dit hoofdstuk sluit af met de aanpak waarmee TNO in de rest van dit rapport die vragen beantwoordt (Sectie 6.4).

Dit hoofdstuk is voornamelijk gebaseerd op informatie die vanuit Rijkswaterstaat aan TNO is aangeleverd, en dan met name de oplegmemo met betrekking tot marktinschakeling bij het toepassen van SBE in projecten [22] en de opdrachtomschrijving aan TNO [3].

6.1 Visie Rijkswaterstaat

Zoals eerder in Sectie 3.1 genoemd, wordt binnen Rijkswaterstaat in het MultiWaterWerk (MWW) project samengewerkt om te komen tot gestandaardiseerde werkwijzen voor VenR-projecten op het gebied van sluizen. De standaardisatie moet leiden tot uniformer functionerende sluizen, en een voor zowel Rijkswaterstaat als voor marktpartijen efficiëntere werkwijze, zonder in te boeten op de effectiviteit, en de betrouwbaarheid van de besturingssoftware. Belangrijke factoren hierbij zijn onder meer de totale kosten, beschikbaarheid en veiligheid.

Een belangrijk onderdeel hierbij is het introduceren van productfamilies [23], groepen van soortgelijke objecten. Hierbij worden zo veel mogelijk de binnen Rijkswaterstaat bestaande productfamilies behouden. Producten binnen een productfamilie hebben doorgaans kenmerken en eisen die grotendeels overeenkomen, en waarvoor veel kan worden hergebruikt. Zo zullen wat betreft deuren van sluizen bijvoorbeeld twee roldeuren veel gemeen hebben met elkaar, maar grotere verschillen vertonen in relatie tot draaideuren. Daarbij wordt voorzien dat voor verschillende onderdelen van het systeem (hardware, software, processen, etc) andere keuzes gemaakt kunnen worden wat betreft de mate van standaardisatie die wordt toegepast. Echter, het principe is telkens hetzelfde. Minder verschillen per object, en minder maatwerk, moet leiden tot meer hergebruik, wat zorgt voor minder werk en lagere kosten.

Concreet moet voor de introductie van productfamilies (ook wel productplatformen), bekeken worden wat er gemeenschappelijk is binnen een productfamilie. Daarnaast kan het zijn dat bepaalde aspecten optioneel kunnen zijn, zoals bijvoorbeeld het wel of niet aanwezig zijn van bepaalde verkeerslichten afhankelijk van de verkeerssituatie voor landverkeer voor het betreffende object. Hoewel het alsnog kan voorkomen dat bepaalde zaken niet goed te standaardiseren zijn, en het dan mogelijk is deze aspecten vrij te laten, is het beter dit zo veel mogelijk te voorkomen. Vanuit de gemeenschappelijke en optionele aspecten kunnen modules worden samengesteld. Door een basismodule met de gemeenschappelijke elementen te gebruiken, en hier de relevante optionele modules aan toe te voegen, kan een nieuw product binnen de productfamilie worden samengesteld. In plaats van dat op basis van maatwerk een nieuwe product vanuit het niets geheel opnieuw moet worden ontworpen (*make-to-order*, gemaakt op bestelling), kan het product dus worden samengesteld (geconfigureerd) uit al bestaande modules (*configure-to-order*, configureren op bestelling). Rijkswaterstaat werkt hiervoor aan een configurator, met een bibliotheek van modules, waarmee een sluis kan worden samengesteld. Gegeven dat bepaalde algemene eisen worden ingevoerd, zoals de afmetingen van het object, komen uit de configurator dan de meer gedetailleerde eisen voor de vraagspecificatie.

Zoals al aangegeven, wordt voorzien dat voor verschillende onderdelen van het systeem (hardware, software, etc) andere keuzes gemaakt kunnen worden wat betreft de mate van standaardisatie die wordt toegepast. Specifiek voor de besturingssoftware stelt Rijkswaterstaat dat de standaardisatie moet leiden tot uniformer functionerende sluizen, en een voor zowel Rijkswaterstaat als voor marktpartijen efficiëntere werkwijze, zonder in te boeten op betrouwbaarheid van de besturingssoftware.

Om die standaardisatie te bewerkstelligen moeten besturingseisen dan modulair kunnen worden gespecificeerd. De besturingseisen die moeten gelden voor alle sluizen, worden gescheiden gehouden van de besturingseisen die gemeenschappelijk zijn voor alle roldeuren, en van de eisen die alleen moeten gelden voor draaideuren. Gegeven een geconfigureerd nieuw product komen de bijbehorende eisen dan uit de configurator, en kunnen deze waar nodig worden aangevuld met product-specifieke eisen. En ook de besturingssoftware wordt dan niet meer telkens vanuit het niets geheel opnieuw ontworpen en gerealiseerd, maar voor een groot gedeelte gebaseerd op hergebruik vanuit werk dat is gedaan voor eerdere producten, en is geconsolideerd in modules.

6.2 Plannen Rijkswaterstaat

In deze sectie worden de plannen van Rijkswaterstaat rondom SBE besproken, inclusief de al gemaakte keuzes.

6.2.1 Rolverdeling Rijkswaterstaat en marktpartijen

MWW heeft de afgelopen jaren samen met de Technische Universiteit Eindhoven (TU/e) onderzocht in hoeverre SBE voor het ontwikkelen van besturingssoftware invulling kan geven aan deze geschetste visie. Dus, in hoeverre SBE gebruikt kan worden om op betrouwbare en efficiënte wijze de besturing te specificeren, ontwerpen, valideren, realiseren en standaardiseren. In hoofdstuk 7 wordt in meer detail ingegaan op de uitkomsten hiervan. De voorlopige conclusie vanuit MWW is dat SBE zorgt voor completere en consistentere besturingseisen, en dat zowel de besturingseisen als de uiteindelijke besturingssoftware efficiënter kunnen worden ontwikkeld. Met behulp van modulaire plant en requirements specificaties lijkt dit ook uitstekend te passen binnen de visie rondom productfamilies en standaardisatie.

In de samenwerking met de TU/e is voornamelijk door master studenten en promovendi SBE toegepast, inclusief onder andere het modelleren van plants en requirements, het synthetiseren van besturingsmodellen, het simuleren van besturingen, en het genereren van besturingssoftware. In de dagelijkse praktijk werkt Rijkswaterstaat echter samen met marktpartijen. Dit is ook voorzien voor de toekomst. Dit mede omdat door het grote aantal VenR projecten dat moet worden uitgevoerd de komende jaren, Rijkwaterstaat niet in staat is die allemaal volledig zelf uit te voeren. Wat betreft rolverdeling is Rijkwaterstaat dan verantwoordelijk voor het aanleveren van de juiste eisen voor de besturing, en een marktpartij maakt op basis daarvan de daadwerkelijke besturing. Bij de samenwerking tussen Rijkswaterstaat en de TU/e zijn marktpartijen tot nu toe niet of slechts beperkt betrokken geweest.

6.2.2 Toepassingsmogelijkheden van SBE binnen Rijkswaterstaat

Rijkswaterstaat heeft daarom een analyse gedaan naar mogelijke scenario's voor de marktinschakeling. Hierbij is geconcludeerd dat SBE voor verschillende doeleinden kan worden gebruikt, en zijn voor Rijkswaterstaat de volgende voornaamste toepassingen geformuleerd [22]:

1. Het uitvoeren van simulaties voor de sluisbediening en -besturing,
   a. in de contractvoorbereidingsfase om klanteisen te valideren.
   b. in de realisatiefase om het ontwerp te valideren voordat de besturingssoftware wordt gerealiseerd.

2. Het genereren van besturingssoftware van een sluis in de realisatiefase,
   a. voor de besturing van het testsysteem dat de aannemer gebruikt (bijvoorbeeld voor een 'digital twin').
   b. voor het besturingssysteem van de sluis.

6.2.3 Verantwoordelijkheid binnen Rijkswaterstaat voor toepassing SBE

Wat betreft de verantwoordelijkheid binnen Rijkswaterstaat met betrekking tot marktinschakeling zijn door Rijkswaterstaat de volgende twee opties gedefinieerd [22]:

A. “De verantwoordelijkheid voor de realisatie van de SBE-producten ligt bij een centraal RWS-onderdeel die deze werkzaamheden uitbesteedt aan een deskundige partij. De SBE-producten worden vervolgens ter beschikking gesteld aan de hoofdopdrachtnemer van het project.”
B. “De verantwoordelijkheid voor de realisatie van de SBE-producten ligt bij de partij die een deskundige marktpartij inschakelt. In dat geval kunnen hoofdopdrachtnemers zelf een deskundige partij inschakelen om SBE-producten te realiseren.”

Met ‘SBE producten’ wordt hier bedoeld de met een SBE manier van werken ontwikkelde modellen en besturingssoftware.

Optie B is de werkwijze die veel aanleg- en onderhoudsprojecten toepassen waar geen bouwblokken van toepassing zijn, om een opdrachtnemer integraal verantwoordelijk te kunnen stellen voor het gerealiseerde werk. Ook bij het 3BT-bouwblok (bouwblok ter standaardisatie van de bediening, besturing en bewaking voor tunnels) wordt deze werkwijze gevolgd.

6.2.4 Combinaties toepassingen en verantwoordelijkheid

Verder is er gekeken naar de combinatie van toepassingen en verantwoordelijkheid. Hieruit zijn de volgende combinaties gekomen [22]:

I. Combinatie AA (‘A’ voor toepassingen 1 en 2).
II. Combinatie AB (‘A’ voor toepassing 1, ‘B’ voor toepassing 2).
III. Combinatie BB (‘B’ voor toepassingen 1 en 2).

Rijkswaterstaat ziet voor SBE combinatie BA niet als een reële optie, en combinatie BB wordt door directeuren van Rijkswaterstaat gezien als ideale eindsituatie. Daarbij wordt dus de volledige verantwoordelijkheid bij de projecten zelf belegd, en worden project teams dus als opdrachtgever verantwoordelijk voor de uitbesteding, en zijn marktpartijen de opdrachtnemers die de realisatie uitvoeren.

6.2.5 Groeipad en kwalificatiesysteem voor SBE

Gezien dat SBE voor zowel Rijkswaterstaat als marktpartijen nieuw is, wordt een groeipad voorzien [22], om tot deze gewenste eindsituatie te komen. In proefprojecten zal via combinatie AA eerst SBE centraal worden aangestuurd. Heeft een marktpartij via een proefproject aangetoond voldoende deskundig te zijn in SBE om dit succesvol te kunnen toepassen, dan zal worden overgegaan tot combinatie AB. Tevens moet Rijkswaterstaat als deskundig opdrachtgever voldoende kennis en ervaring opbouwen en behouden. Het kan er dus voor kiezen specifieke projecten niet aan de markt uit te besteden maar zelf uit te voeren.

Verder wordt een kwalificatiesysteem voorzien [22], om de correcte toepassing van SBE door marktpartijen te waarborgen. Zeker in het geval die niet centraal worden aangestuurd (verantwoordelijkheidsoptie B). Om de kwalificatie te behalen, doorloopt de marktpartij getrapt het kwalificatietraject: eerst voor toepassing 1 (simulatie) en vervolgens voor toepassing 2 (realisatie). Marktpartijen kunnen zich aanmelden om de mogelijkheid te krijgen deze ervaring op te doen. Rijkwaterstaat zal ruim van te voren aankondigen dat voor toekomstige renovaties SBE gebruikt gaat worden, zodat gegadigde marktpartijen zich hiervoor kunnen voorbereiden en kwalificeren. Verder zorgt Rijkwaterstaat ervoor dat het programma open is, met als doel keuzevrijheid voor opdrachtgevers uit diverse marktpartijen, en schaalbaarheid gezien de grote VenR uitdagingen. Voor 3BT-bouwblokken wordt een soortgelijk kwalificatiesysteem opgezet.

6.2.6 SBE proefprojecten

Voordat een dergelijk kwalificatiesysteem wordt opgezet wil Rijkswaterstaat echter eerst meer ervaring opdoen met SBE, ook in de samenwerking met marktpartijen. In oktober 2021 is de Roadmap SBE besproken met de directeuren en afdelingshoofden van de in MWW samenwerkende partijen. Hier is besloten om SBE toe te gaan passen voor het domein sluizen in drie proefprojecten, uitgevoerd onder combinatie AA. De volgende drie proefprojecten zijn gedefinieerd [22]:

1. Beproeven simulatie in planfase (toepassing 1a): Den Oever.
2. Beproeven simulatie in ontwerpfase realisatie (toepassing 1b): Weurt – Heumen.
3. Beproeven genereren in realisatie softwarecode besturingssysteem sluis (toepassing 2): Maaswerken 2.

Hierbij kan Rijkswaterstaat er voor kiezen om het modelleren gedeeltelijk zelf te doen, om kennis op te bouwen, en gedeeltelijk uit te besteden aan marktpartijen, om ervaring op te doen met de toepassing van SBE door marktpartijen.

Na elk proefproject volgt een evaluatie, conform Innoveren, Uniformeren en Produceren (IUP). De criteria hiervoor volgen uit de doelstelling, dus in hoeverre de betreffende toepassing van toegevoegde waarde is voor projecten en marktpartijen om op betrouwbare en efficiënte wijze de besturing te specificeren, ontwerpen, valideren, realiseren en standaardiseren. Na elke evaluatie volgt een besluit over ingebruikname bij toekomstige projecten.

De realisatie van de besturingssoftware op een sluis (in proefproject 3) is onderdeel van de industriële automatisering (IA). Vandaar dat is afgesproken dat voordat besloten wordt om gegenereerde besturingssoftware in gebruik te nemen in sluizen, er via Industriële Automatisering Sourcing (IAS) beleidsafstemming zal plaatsvinden.

6.2.7 SBE toolkeuze

Een belangrijk onderdeel van SBE is automatisering en computer ondersteuning. Hiervoor zijn softwaretools nodig. Die tools stellen Rijkswaterstaat en marktpartijen onder andere in staat om plants en requirements te modelleren, besturingsmodellen te synthetiseren, besturingen te simuleren, en uiteindelijk de code van de besturingssoftware te genereren.

Rijkswaterstaat heeft in de samenwerking met de TU/e gebruik gemaakt van de CIF modelleertaal en bijbehorende tools. CIF is ontwikkeld door de TU/e. Sinds 2020 is het onderdeel van de Eclipse Supervisory Control Engineering Toolkit (ESCET™) [24]. Dit is een open source project van de Eclipse Foundation. Rijkwaterstaat heeft voor de ESCET toolkit gekozen vanwege de kernwaardes van de Eclipse Foundation: transparantie, openheid, meritocratie en leveranciersonafhankelijkheid. De Eclipse Foundation streeft naar open ecosystemen, waarin diverse partijen kunnen samenwerken, en niemand wordt buitengesloten.

In de proefprojecten wordt expliciet het gebruik van de ESCET toolkit meegenomen, wat betreft opbouw van kennis, en wat betreft geschiktheid voor de doelstellingen van Rijkswaterstaat.

6.2.8 Centraal SBE team

Rijkswaterstaat zal een centraal SBE team oprichten om alle activiteiten rond SBE te regisseren en uit te voeren. Het voorziet de volgende taken voor dit team [22]:

- Het realiseren en/of begeleiden van projecten bij de toepassing van SBE en het toetsen en beoordelen van SBE-producten die binnen projecten worden gerealiseerd.
- Onderzoek, ontwikkeling en het borgen dat de kennis en ervaring die binnen RWS op gebied van SBE worden opgebouwd op peil blijven.

- Opstellen van generieke modellen, waaronder een Generiek Model Sluis (GMS) en Generiek Model Brug (GMB), op basis waarvan de besturingen van alle RWS sluizen en bruggen gemodelleerd kunnen worden, en het inrichten van het wijzigingsbeheer op deze modellen.
- Inrichten en aansturen van het kwalificatiesysteem voor SBE-opdrachtnemers (en daarbinnen het begeleiden van potentiële SBE-gekwalificeerde opdrachtnemers bij eerste toepassing SBE).
- Beheren van de specifieke modellen per sluis (optioneel).

Het team zal ook nauw betrokken zijn bij de proefprojecten, onder andere om Rijkswaterstaat te laten ‘leren, ervaren en evalueren’, en om geschikte marktpartijen te verkrijgen. Hierbij zal veel met de marktpartijen moeten worden samengewerkt. Zodra meerdere marktpartijen zijn gecertificeerd zal meer worden uitbesteed. Uiteindelijk voorziet Rijkswaterstaat dat deze SBE taken binnen de lijnorganisatie worden belegd.

6.3 Vraagstelling aan TNO

6.3.1 Onderzoeksopdracht vanuit IAS

Zoals reeds besproken in Sectie 3.1 onderzoekt IAS een lichtere vorm van standaardisatie voor sluizen, naar aanleiding van het BIT-advies. SBE is hiervoor een optie die wordt overwogen. Ter voorbereiding op proefproject 3 wordt in opdracht van IAS daarom onderzoek gedaan specifiek naar het genereren van besturingssoftware. Het doel hierbij is om te achterhalen wat het effect is van het gebruik van SBE in het algemeen, en codegeneratie in het bijzonder, met inzet van de ESCET toolkit, op de resulterende besturingssoftware. Belangrijk hierbij is het effect op de betrouwbaarheid van de besturingssoftware, en in hoeverre deze nieuwe werkwijze in overeenstemming is met de machinerichtlijn en andere regels en richtlijnen waar Rijkswaterstaat aan moet voldoen. Het onderzoek is opgenomen in de Plan van Aanpak / jaarplan 2022 van IAS en wordt namens IAS uitgevoerd door MWW.

Dit onderzoek richt zich op toepassing 2 (realisatie) en niet op toepassing 1 (simulatie). Specifiek wordt voor dit onderzoek gekeken naar het genereren van besturingssoftware voor een draaibrug, de Oisterwijksebaanbrug bij Tilburg. Daarvoor zijn reeds de afgelopen jaren in de samenwerking met de TU/e via SBE diverse modellen in diverse gradaties gemaakt [25, 26, 27].

Rijkswaterstaat past SBE tijdens dit onderzoek ook zelf toe, om er meer ervaring mee op te doen, onder andere door het opstellen van een risicobeoordeling, het ontwikkelen van modellen voor synthese en simulatie rekening houdend met foutgedrag, het opdelen van de eisen in een regulier en safety deel (zie ook Secties 7.2.3 en 7.9), het genereren van PLC-code, het opstellen van testprotocollen en hardware-in-the-loop testen, en het documenteren van het proces en de modellen.

De resultaten van het onderzoek worden door het kernteam IAS, en in samenspraak met de themagroep Architectuur, gebruikt bij de beoordeling of SBE geschikt wordt geacht voor standaardisatie van de besturing van objecten. MWW zal hierbij als adviseur optreden.

6.3.2 Vragen van Rijkswaterstaat

Concreet heeft Rijkswaterstaat de volgende 10 vragen opgesteld [3]:

1. Is de ESCET-toolkit voor zowel RWS als marktpartijen voldoende inzetbaar om gestandaardiseerde en onderhoudbare besturingssoftware te vervaardigen voor besturingssystemen van infrastructurele objecten, zoals beweegbare bruggen en sluizen?

2. Wat zijn sterke punten en kansen ten aanzien van de inzetbaarheid van de ESCET-toolkit voor softwaregeneratie voor besturingssystemen van infrastructurele objecten, zoals beweegbare bruggen en sluizen?

3. Wat zijn zwaktes en bedreigingen ten aanzien van deze inzetbaarheid en welke beheersmaatregelen zijn hiervoor te nemen?

4. Is het mogelijk middels de ESCET-toolkit besturingssoftware te vervaardigen die in overeenstemming is met de machinerichtlijn en de relevante geharmoniseerde normen en is het toepassen van ESCET een belemmering om te voldoen aan SIL of PL levels?

5. Zouden de modellen en de softwarecode die in de Proof of Concept worden vervaardigd voor de Oisterwijksebaanbrug in de praktijk van een op te zetten renovatieproject veilig te maken zijn, in overeenstemming met de machinerichtlijn en met eventueel relevante geharmoniseerde normen?

6. Indien het antwoord op vraag 5 negatief is, welke aanpassingen aan de modellen en tools in ESCET moeten dan plaatsvinden om softwarecode te verkrijgen die in overeenstemming is met de machinerichtlijn en de relevante geharmoniseerde normen?

7. Kan de gegenereerde code door compilers in de praktijk van een op te zetten renovatieproject betrouwbaar worden omgezet naar machinecode, zowel van veel gebruikte PLC-fabrikanten zoals Siemens, ABB en Schneider, als van veel gebruikte industriële pc's?

8. Kan de verkregen code in de praktijk van een op te zetten renovatieproject betrouwbaar op een PLC dan wel industriële pc werkend gemaakt worden, als gebruik gemaakt wordt van de modellen op basis waarvan de software is gegenereerd?

9. Voldoet de softwarecode die wordt gegenereerd middels ESCET-toolkit altijd aan de ingevoerde plant- en eisenmodellen? De beantwoording van deze vraag kwalitatief/theoretisch onderbouwen door beschouwing van het algoritme.

10. Wordt door het toepassen van SBE middels de ESCET-toolkit betrouwbare software gegenereerd? Hiermee wordt bedoeld dat deze altijd op dezelfde wijze functioneert als de aangeleverde modellen en de besturing niet in ongedefinieerde toestanden terecht kan komen. De beantwoording van deze vraag kwalitatief/theoretisch onderbouwen.

Hierbij zijn vragen 1-3 en 9+10 voorgelegd aan TNO (en TU/e, zie Sectie 6.4), en vragen 4-8 aan CGI. Hierbij is expliciet voorzien dat partijen tijdens de uitvoering van het onderzoek onderling informatie uitwisselen. Gezien de focus van CGI op richtlijnen en normen, en de aspecten rondom codegeneratie aan het einde van het ontwerpproces, gaat TNO hier iets minder op in met dit rapport. De antwoorden op vragen 4-8 zoals beantwoord door CGI, en de aanbevelingen en verdere conclusies vanuit het CGI rapport [4], zijn deels meegenomen in dit TNO rapport. Het is echter aan te raden bij het lezen van dit TNO rapport ook het CGI rapport te lezen, aangezien de twee rapporten samen de volledige tien vragen beantwoorden.

6.4 Aanpak van TNO

TNO ziet vraag 1 als de hoofdvraag, vragen 2 en 3 als directe subvragen die samen hoofdvraag 1 beantwoorden, en vragen 4-10 als deelaspecten daarvan. TNO is verder van mening dat om de aan TNO gestelde vragen te beantwoorden, niet alleen naar toepassing 2 (realisatie) kan worden gekeken, maar ook toepassing 1 (simulatie) moet worden meegenomen. De kwaliteit van de uiteindelijke besturingssoftware hangt af van tal van aspecten, die invloed hebben tijdens het gehele ontwerpproces. Daarnaast moet niet alleen specifiek naar de Eclipse ESCET toolkit gekeken worden, maar naar de SBE methode in het algemeen, met de Eclipse ESCET toolkit als een mogelijke toolkit om te gebruiken bij het toepassen van deze methode.

TNO beantwoordt daarom voor dit onderzoek de volgende hoofdvraag (n.a.v. vraag 1):

In hoeverre is SBE een methode die voor Rijkswaterstaat geschikt is voor het vervaardigen van besturingssoftware voor besturingssystemen van infrastructurele objecten, zoals beweegbare bruggen en sluizen?

Hierbij gaat TNO in op de kansen (sterke punten) ten aanzien van deze methode, maar ook de aandachtspunten (zwaktes en bedreigingen) en hoe die kunnen worden ondervangen met beheersmaatregelen (n.a.v. vragen 2 en 3). Specifiek wordt gekeken naar diverse relevante aspecten, waaronder hoe de methode relateert aan standaardisatie, betrouwbaarheid / veiligheid, onderhoud, efficiëntie, samenwerking met en uitbesteding aan marktpartijen, en tool ondersteuning (via de ESCET toolkit).

Belangrijk hierbij is om te begrijpen hoe de SBE methode voor dergelijke aspecten verschilt van een traditionele op documenten gebaseerde methode, en welke onderdelen van de SBE methode, zoals specificatie, synthese, simulatie en codegeneratie, hier aan bijdragen (en met welke garanties) of juist een extra uitdaging opleveren (en hoe daar mee om te gaan). Daarom worden, waar het mogelijk en relevant is, de verschillende aspecten gerelateerd aan de stappen van de SBE methode.

Verder is het goed om onderscheid te maken tussen de verschillende toepassingsmogelijkheden voor SBE. Als bijvoorbeeld gekeken wordt naar toepassing 1a (simulaties om klanteisen te valideren), zijn niet alle stappen van het ontwikkelproces relevant. Voor toepassingen 1b (simulaties om ontwerp te valideren voor realisatie) en 2a (genereren besturingssoftware voor test systeem) zijn meer stappen relevant, en voor toepassing 2b (genereren besturingssoftware voor de sluis) zijn alle stappen relevant. Daarom worden, waar het mogelijk en relevant is, de diverse aspecten en stappen ook gerelateerd aan de mogelijke toepassingen, die gefaseerd kunnen worden uitgerold.

TNO is van mening dat met deze aanpak er een goed beeld geschetst kan worden van de verschillen tussen een traditionele methode en de SBE methode, de diverse aspecten die daarbij komen kijken, en wat de meerwaarde en risico’s zijn. Daarmee kan Rijkswaterstaat vervolgens beter bepalen in hoeverre het SBE geschikt vindt voor het vervaardigen van besturingssoftware voor besturingssystemen van infrastructurele objecten. Het kan dan beter inschatten wat de voordelen zijn ten opzichte van de huidige manier van werken, wat de risico’s zijn, of het de risico’s voldoende denkt te kunnen beperken, en hoe het de verhouding tussen de meerwaarde en (overgebleven) risico’s ziet, om daarmee te besluiten of het SBE al dan niet breder in wil zetten.

TNO maakt voor dit rapport gebruik van diverse bronnen, waaronder eigen kennis en ervaring, en informatie vanuit Rijkswaterstaat, TU/e en CGI. In het bijzonder worden de conclusies vanuit het CGI rapport [4] met betrekking tot vragen 4-8 in dit rapport deels meegenomen. Ook wordt gebruik gemaakt van specifieke expertise van de TU/e. Zie voor de verdere verantwoording ook Hoofdstuk 8.

7 Kansen en aandachtspunten SBE

7.1 Introductie

In dit hoofdstuk worden de kansen en aandachtspunten met betrekking tot SBE besproken voor diverse aspecten, en worden deze aspecten waar het mogelijk en relevant is ook gerelateerd aan de stappen van het SBE ontwikkelproces, en aan de gefaseerde introductie van SBE. Bij de aandachtspunten wordt tevens aangegeven aan hoe Rijkswaterstaat de risico's ervan zou kunnen beperken. Voor een genummerd overzicht van de aanbevelingen, zie Sectie 4.3.

Hoewel de kansen en aandachtspunten over het algemeen in bepaalde mate ook voor model-gebaseerd (MBE) en verificatie-gebaseerd ontwikkelen (VBE) gelden, ligt de focus in dit hoofdstuk (en dit rapport) voornamelijk op SBE ten opzichte van een traditionele manier van ontwikkelen.

7.1.1 Aspecten

Bij het ontwikkelen van besturingssoftware, en het veranderen van de manier waarop die besturingssoftware wordt ontwikkeld, spelen tal van aspecten een rol. In dit rapport worden diverse van deze aspecten besproken. Hierbij is het praktisch ondoenbaar om alle aspecten uitvoerig te bespreken. Er is daarom een selectie gemaakt van die aspecten waarvan TNO denkt dat deze het meest relevant zijn voor de uitdagingen van Rijkswaterstaat, en de kansen en aandachtspunten die daarbij een rol spelen als SBE als methode wordt ingezet om die uitdagingen te adresseren. Hier zijn ook de door Rijkswaterstaat gestelde vragen expliciet in meegenomen. De volgende aspecten zullen worden besproken:

Ontwikkelproces

De invloed van SBE op het ontwikkelproces ten opzichte van traditioneel ontwikkelen wordt besproken: van het basisontwikkelproces voor het vervaardigen van een eerste besturing van een object, tot de onderhoud en evolutie ervan, en de rol van hergebruik en standaardisatie als een object onderdeel is van een productfamilie.

- **Ontwikkelproces SBE** (Sectie 7.2): Als eerste wordt per stap van het basisontwikkelproces in meer detail ingegaan op het verschil tussen traditioneel ontwikkelen en SBE.
- **Onderhoud en evolutie** (Sectie 7.3): Na het ontwikkelen van een besturing voor een object volgen in de levenscyclus van dat object doorgaans diverse keren onderhoud (typisch kleinere wijzigingen, zoals het repareren van fouten) en evolutie (bijvoorbeeld ook grotere wijzigingen, zoals het toevoegen van nieuwe functies). De invloed die SBE heeft op deze processen wordt apart besproken.
- **Hergebruik en standaardisatie** (Sectie 7.4): Rijkswaterstaat heeft tal van objecten die het onder wil brengen in productfamilies, om via standaardisatie en hergebruik tot betere kwaliteit te komen tegen lagere kosten. Hier wordt bekeken hoe SBE hier aan kan bijdragen en hoe dit het ontwikkelproces beïnvloedt.

Impact

Hier wordt besproken wat de voornaamste impact is die SBE heeft op het ontwikkelproces, in de vorm van de (extra) garanties die het geeft, en de invloed daarvan op een aantal voor Rijkswaterstaat essentiële aspecten: de veiligheid van de objecten, efficiëntie van het ontwikkelproces en de afhankelijkheid van leveranciers.

- **Garanties van SBE** (Sectie 7.5): De formele modellen en formele methoden die bij SBE gebruikt worden geven tal van garanties, die hier in meer detail worden beschreven.

- **Veiligheid en betrouwbaarheid** (Sectie 7.6): De impact van het SBE ontwikkelproces met bijbehorende garanties op de veiligheid en betrouwbaarheid van de objecten wordt hier nader bekeken.
- **Efficiëntie** (Sectie 7.7): De impact van het SBE ontwikkelproces met bijbehorende garanties op de efficiëntie van het ontwikkelproces wordt hier nader bekeken, waarbij ook de impact op communicatie essentieel is.
- **Afhankelijkheid van leveranciers** (Sectie 7.8): De impact van het SBE ontwikkelproces en de bijbehorende garanties op de afhankelijkheid van zowel fabrikanten van PLC's als marktpartijen die voor Rijkswaterstaat besturingen ontwikkelen wordt hier nader bekeken.

Toepasbaarheid

Hier wordt besproken of SBE kan worden toegepast op infrastructurele systemen zoals die van Rijkswaterstaat, en of daarmee de voordelen van SBE ook voor die toepassingen kunnen worden behaald.

- **Geschiktheid voor infrastructurele domein** (Sectie 7.9): Hier wordt besproken in hoeverre de toepasbaarheid voor de systemen van Rijkswaterstaat in de praktijk is aangetoond, en in hoeverre infrastructurele systemen verschillen van andere systemen waarvoor SBE is toegepast.
- **Schaalbaarheid van formele methoden** (Sectie 7.10): SBE maakt gebruik van formele methoden zoals synthese. In hoeverre zijn deze methoden en de daarbij gebruikte algoritmes schaalbaar genoeg om toegepast te worden voor infrastructurele systemen, om daarbij de voordelen van SBE wat betreft onder andere efficiëntiewinst te kunnen behalen.

Volwassenheid

De volwassenheid van SBE wordt hier besproken, door te kijken naar de tooling die er voor beschikbaar is, hoe het staat met de kennis en kunde van Rijkswaterstaat en marktpartijen omtrent SBE, de impact die het introduceren van SBE heeft op bedrijfsprocessen en bedrijfscultuur, hoe het staat met het ecosysteem rondom SBE, en de positie die SBE heeft ten opzichte van andere ontwikkelprocessen.

- **Tooling** (Sectie 7.11): Hier wordt besproken wat Eclipse ESCET biedt aan tooling ter ondersteuning van SBE, of er alternatieven zijn, en hoe het staat met de kwaliteit, certificatie en commerciële ondersteuning van SBE-tools.
- **Kennis en kunde** (Sectie 7.12): Hier wordt besproken in hoeverre Rijkswaterstaat en marktpartijen personeel met de juiste kennis en kunde hebben, kunnen aantrekken of kunnen opleiden, om SBE en de bijbehorende tooling succesvol in te kunnen zetten.
- **Veranderen van bedrijfsprocessen en bedrijfscultuur** (Sectie 7.13): Het introduceren van SBE heeft impact op bedrijfsprocessen en bedrijfscultuur, en de manier waarop dit veranderproces wordt ingezet wordt hier nader bekeken.
- **Ecosysteem** (Sectie 7.14): In hoeverre het ecosysteem rondom SBE en de ESCET toolkit rijk en volwassen genoeg is, dan wel een risico vormt voor de continuïteit naar de toekomst toe, wordt hier toegelicht.
- **Ontwikkelproceskeuze MBE/VBE/SBE** (Sectie 7.15): Hoe SBE zich verhoudt tot alternatieve ontwikkelprocessen wordt hier verder besproken, kijkend naar de alternatieven.

Deze aspecten zijn zo gekozen dat ze zoveel mogelijk onafhankelijk kunnen worden besproken. Echter, ze relateren wel op tal van manieren aan elkaar. Er wordt dan ook vaak bij het bespreken van het ene aspect verwezen naar diverse andere aspecten.

7.1.2 Stappen

Deze aspecten zullen, waar het mogelijk en relevant is, ook worden gerelateerd aan de stappen van het ontwikkelproces voor besturingssoftware, zoals reeds besproken in Sectie 5.2, en dan met name hoe dit verschilt voor een traditionele en SBE manier van ontwikkelen:

- **Specificatie:** Het specificeren van complete, consistente en eenduidige besturingseisen, traditioneel in de vorm van documenten, en bij SBE in de vorm van plant en requirements modellen.
- **Ontwerp:** Het ontwerpen van de besturing, traditioneel handmatig in de vorm van documenten, en bij SBE automatisch door middel van synthese van het besturingsmodel.
- **Realisatie:** Het realiseren van de besturingssoftware, traditioneel door middel van handmatig programmeren, en bij SBE met behulp van automatische codegeneratie.
- **Verificatie:** Het controleren dat de besturing voldoet aan de specificatie, traditioneel door middel van testen, en bij SBE door dit met computer ondersteuning te bewijzen, voor zover dit nog niet door constructie is gegarandeerd.
- **Validatie:** Het controleren dat de besturing naar wens opereert, traditioneel door middel van testen, en bij SBE additioneel door middel van door de computer mogelijk gemaakte simulaties.

In het daadwerkelijke ontwerpproces worden deze stappen vaak niet precies in deze volgorde, en niet slechts een enkele keer doorlopen. Specificatie en ontwerp gaan vaak (gedeeltelijk) hand in hand en dat geldt ook voor verificatie en validatie. Zeker met SBE kunnen verificatie en validatie ook al eerder op de modellen worden uitgevoerd in plaats van op het eind op de realisatie, en vaker plaatsvinden (op de modellen en de realisatie). Verificatie en validatie kunnen worden gecombineerd of omgedraaid. Verificatie en validatie kunnen problemen aan het licht brengen, wat dan leidt tot het gedeeltelijk herhalen van eerdere stappen, bijvoorbeeld door de specificatie of het ontwerp aan te passen. Daarnaast wordt vaak iteratief ontwikkeld, waarbij eerst een deelsysteem wordt ontworpen en vervolgens de stappen nog meerdere malen worden doorlopen voor uitgebreidere delen van het systeem. Kortom, er kan veel variatie zitten in het precieze ontwerpproces. Echter, deze vijf stappen zijn typisch onderdeel van het proces, en het is derhalve nuttig voor de verdere uiteenzetting in dit rapport om ze apart te kunnen benoemen.

7.1.3 Fasen

Verder zullen de aspecten en stappen, waar het mogelijk en relevant is, ook worden gerelateerd aan de gefaseerde introductie van SBE, waarbij TNO de volgende fasen onderscheidt:

- **Eisenformulering:** Het toepassen van SBE volgens doelstelling 1a, om in de planfase en/of contractvoorbereidingsfase te komen tot kwalitatief goede eisen voor aanbesteding. Dit betreft specificatie, synthese, simulatie en eventueel formele verificatie.
- **Validatie proefontwerp:** Het toepassen van SBE volgens doelstelling 1b, om na aanbesteding het proefontwerp van de aannemer te valideren. Dit betreft specificatie, synthese, simulatie en formele verificatie.
- **Realisatie testsysteem:** Het toepassen van SBE volgens doelstelling 2a, om de besturingssoftware voor het testsysteem van de aannemer te genereren. Dit betreft specificatie, synthese, simulatie, formele verificatie en codegeneratie.
- **Realisatie voorlopige besturing:** Het toepassen van SBE volgens doelstelling 2b, om de voorlopige besturingssoftware voor een object te genereren, terwijl programmeurs verantwoordelijk blijven voor het vervaardigen van de definitieve besturingssoftware voor het object. Dit betreft specificatie, synthese, simulatie, formele verificatie en codegeneratie.
- **Realisatie besturing:** Het toepassen van SBE volgens doelstelling 2b, om de besturingssoftware voor een object te genereren, en ongewijzigd in te zetten voor het besturen van het object. Dit betreft specificatie, synthese, simulatie, formele verificatie en codegeneratie.

Elke fase bevat ook alle vorige fasen, en gaat daarmee dus wat verder, tot de volledige inzet van SBE tijdens het gehele ontwikkelproces in de laatste fase. Waar de relatie met deze fasen niet expliciet wordt gemaakt, kan de relatie doorgaans worden afgeleid uit de stappen die in een bepaalde fase een rol spelen.

7.2 Ontwikkelproces SBE

Ten opzichte van een traditionele manier om besturingssoftware te ontwikkelen is SBE is een alternatief ontwikkelproces, waarbij voor elke stap van het ontwikkelproces wel iets verandert (zie ook Sectie 5.2). Hier volgt in wat meer detail een beschouwing van het gehele proces, stap voor stap, en hoe dit verschilt tussen de traditionele manier van werken en SBE. In deze sectie ligt de focus vooral op het proces en hoe het verschilt. Diverse aspecten die hieraan relateren worden in andere secties besproken.

7.2.1 Stap S1: Specificatie

Het vastleggen van besturingseisen gebeurt traditioneel in documenten. Dit leidt vaak tot incompleetheid, inconsistentie en ambiguïteit. Bij SBE worden de besturingseisen gemodelleerd in de vorm van plant en requirements modellen (zie Sectie 5.6.2). Hoewel bij SBE nog altijd een document als basis voor de modellen kan dienen, staan bij SBE als model-gebaseerde methode de modellen centraal. Dit heeft diverse voordelen, zoals reeds besproken in Sectie 5.6.

Het eenduidig opschrijven van besturingseisen in een document is namelijk bijzonder lastig. Vaak zijn tekstuele beschrijvingen in natuurlijke taal voor meerdere interpretaties vatbaar. De veiligheid van objecten is in het geding als de eisen ambigu zijn, omdat dan onduidelijk is wat er in die situaties moet gebeuren. Diegene die de eisen opstelt heeft daar een bepaald beeld bij. Een ander persoon (of afdeling of marktpartij), die de eisen vervolgens in software moet implementeren, kan deze anders interpreteren. Hierbij is sprake van een kloof tussen de specificatie en de implementatie. Eisen worden mogelijk niet juist afgehandeld in de besturingssoftware, of de verkeerde eisen worden getest, wat het risico op onveilige situaties verhoogt. Het gebruik van (uitgebreide) reviews kan deze problemen verkleinen, maar vaak niet volledig voorkomen.

Dit is eerder onderzocht door de TU/e, door een andere groep dan waar Rijkswaterstaat mee samenwerkt rondom SBE. Voor de Algerabrug in Krimpen aan de IJssel zijn de besturingseisen geformaliseerd [28]. Dit bleek een moeizaam proces, omdat de eisen over verschillende documenten waren verspreid, lang niet altijd precies en consistent waren, en er eisen ontbraken. Daarnaast werden overwegingen die vooral op implementatieniveau spelen gemengd met hoog niveau correctheidseisen. In het werk wordt geconcludeerd dat besturingseisen een prominentere rol dienen te krijgen in het ontwerpproces en formeel dienen te worden gespecifieerd. Daarnaast is ten behoeve van verificatie gekeken naar systeemvalidatie van beweegbare bruggen [29]. Hier is wederom gebruik gemaakt van functionele en veiligheidseisen uit bestaande documenten van Rijkswaterstaat en zijn deze formeel gespecifieerd. Ook hier wordt geconcludeerd dat het buitengewoon moeilijk is om gedrag van systemen effectief in natuurlijke taal te beschrijven. De documenten bevatten dan ook zowel ambiguïteiten als concreet aanwijsbare fouten.

Bij SBE worden formele modellen gebruikt met een wiskundige betekenis (semantiek), zoals toestandsmachines en logische formules. Het gebruik van dergelijke formele modellen leidt tot een eenduidige interpretatie van de besturingseisen. Deze hebben een eenduidige betekenis. Verder wordt hiermee de kloof tussen specificatie en implementatie verkleind (zie onder andere Secties 5.3 en 5.4.3), zodat personen met verschillende achtergronden en rollen elkaar beter begrijpen, onder andere door verbeterde communicatie (zie Sectie 7.7). Het in een vroeg stadium formeel specificeren helpt verder bij het compleet en consistent krijgen van de besturingseisen (zie onder andere Sectie 7.2.5), het veiliger maken van besturingen (zie Sectie 7.6), en het efficiënter maken van het ontwikkelproces (zie Sectie 7.7).

Echter, modelleren is een activiteit die wezenlijk ander is dan het schrijven van documenten. Dit geldt voor MBE in het algemeen, en voor SBE in het bijzonder, aangezien daarbij in restricties moet worden gedacht. Hiervoor moet wel de juiste kennis en kunde aanwezig zijn (zie Sectie 7.12) en er moet de juiste

softwareondersteuning voor worden gebruikt (zie Sectie 7.11). Ook is het veranderen van de manier van werken iets wat aandacht verdient (zie Sectie 7.13). Verder dienen de aanbevelingen van CGI omtrent specificeren en modelleren geadresseerd te worden (zie Sectie 7.6.4).

7.2.2 Stap S2: Ontwerp

Als de besturingseisen eenduidig zijn, moeten deze vervolgens in het ontwerp van de besturing worden verwerkt. De besturing moet in alle mogelijke situaties aan alle gestelde besturingseisen voldoen. Bij een groot aantal besturingseisen, en relaties tussen die eisen, is dat niet altijd triviaal. In het ontwerp van de besturing (in een document of model), of later bij de handmatige implementatie ervan, moet voor elke toestand handmatig bekeken worden of en hoe die eis tot uitdrukking komt. Een enkele eis kan effect hebben op diverse delen (toestanden) van de besturing en kan dus verspreid zitten in het besturingsontwerp, besturingsmodel en de besturingssoftware. Daarnaast kan een ontwerpkeuze in een bepaalde toestand ertoe leiden dat later een onveilige situatie ontstaat. Voor mensen is het lastig dit alles te overzien.

SBE biedt computerondersteund ontwerpen, waarbij de computer helpt bij het maken van correct-door-constructie besturingen. Onderdeel hiervan is het gebruik van synthese, waarbij een computerprogramma alle mogelijke eisen en situaties bekijkt en zorgt dat het gegenereerde besturingsmodel wiskundig correct alle eisen afdekt in alle toestanden. Het houdt hierbij ook rekening met acties die als ze nu worden ingezet later onder bepaalde condities zouden kunnen leiden tot onveilige situaties. Het gegenereerde besturingsmodel voldoet dus per constructie in alle situaties aan alle gestelde eisen. De toepasbaarheid en voordelen van synthese zijn door Rijkswaterstaat samen met de TU/e ook in de praktijk aangetoond in tal van case studies (zie onder andere Sectie 7.9).

Synthese maakt een handmatig ontwerp van een besturingsmodel in feite overbodig door het automatisch genereren van een correct-door-constructie besturingsmodel uit, doorgaans simpelere, plant en requirement modellen. Ook geeft het diverse extra garanties, waarmee ook faalkans omlaag gaat (zie Sectie 7.5). Verder voorkomt dit fouten die mensen vaak maken als ze handmatig complexe zaken moeten uitvoeren, en verhoogt daarmee de veiligheid van de besturing (zie ook Sectie 7.6). Verder zorgt de automatisering voor efficiëntiewinst in het ontwikkelproces (zie Sectie 7.7).

Hierbij is het echter wel essentieel dat de juiste kennis en kunde aanwezig is (zie Sectie 7.12) met betrekking tot wat SBE wel en niet biedt (zie Sectie 7.5), zodat daar op kan worden vertrouwd. Daarnaast moet de juiste softwareondersteuning worden gebruikt (zie Sectie 7.11), moet die op de juist manier worden ingezet (zie Sectie 7.12), en moeten daarbij de juiste schaalbare technieken worden gebruikt (zie Sectie 7.10). Ook is het veranderen van de manier van werken iets wat aandacht verdient (zie Sectie 7.13). Verder dienen de aanbevelingen van CGI omtrent ontwerpen geadresseerd te worden (zie Sectie 7.6.4).

7.2.3 Stap S3: Realisatie

Traditioneel wordt de voor realisatie handmatig geprogrammeerd. Hierbij kunnen zaken fout worden geïnterpreteerd of simpelweg fouten worden gemaakt. Een goede software/PLC-engineer zal dit tot een minimum weten te beperken, maar dit is geen volledige garantie [9] (zie ook Sectie 5.3.2).

Bij SBE wordt vanuit de modellen door een computerprogramma automatisch softwarecode gegenereerd. Dit zorgt er voor dat de code gegarandeerd fout-vrij is, in de zin dat het zich precies zo gedraagt als is vastgelegd in het besturingsmodel (zie ook Sectie 7.5). De toepasbaarheid en voordelen van codegeneratie zijn door Rijkswaterstaat samen met de TU/e ook in de praktijk aangetoond in tal van case studies (zie onder andere Sectie 7.9).

Daarnaast maakt SBE het mogelijk voor verschillende PLC-platformen en leveranciers code te genereren, of voor programmeertalen als C en Java voor industriële PC’s. Daarmee behoort ook het genereren van code voor meerdere platformen en leveranciers tot de mogelijkheden, en kan wellicht eenvoudiger van platform of leverancier worden gewisseld (zie ook Sectie 7.8).

Diverse aspecten die spelen bij handmatig programmeren worden net zo goed meegenomen bij codegeneratie. Zo is het voor software geïmplementeerd in PLC's en waar veiligheid belangrijk is, gebruikelijk om safety PLC's te gebruiken, en veiligheidseisen van de (overige) besturingseisen te scheiden. Dit is ook voor SBE mogelijk [27] (zie ook Sectie 7.9).

Specifiek voor MBE, en daarmee SBE, is dat de code gegenereerd wordt uit modellen. Er zijn intrinsieke verschillen tussen besturingsmodellen en PLC-code. In een toestandsmachine worden bijvoorbeeld acties om meerdere actuatoren te activeren achter elkaar uitgevoerd, terwijl in een PLC diverse actuatoren aan het einde van een cyclus tegelijk actief gemaakt kunnen en moeten worden. Er zijn een aantal extra eigenschappen die moeten gelden voor het besturingsmodel, om er PLC-code uit te kunnen genereren die het systeem correct laat functioneren. Specifiek betreft het dan de *confluence* eigenschap (het maakt niet uit in welke volgorde je de actuatoren aanzet, omdat het effect gelijk is), de *finite response* eigenschap (het aantal acties om actuatoren aan te zetten is eindig), en de *nonblocking under control* eigenschap (het scheiden van controllable en uncontrollable events in de PLC-code levert geen problemen op voor de garanties van synthese). Deze eigenschappen zijn met de computer te controleren met formele verificatie (zie Sectie 7.2.4). Als deze eigenschappen gelden, kan door een codegenerator juiste code worden gegenereerd, die zich gedraagt zoals het besturingsmodel. Het maakt dan niet uit welke keuze in het *supervisor* model wordt gemaakt, omdat het een zogenaamde *implementeerbare supervisor* betreft, en elke keuze tussen controllable events zal leiden tot een juist *controller* model (zie ook Sectie 5.6.3). De precieze definities van de drie eigenschappen, en verdere informatie over het verschil tussen een supervisor en een controller, is te vinden in de literatuur [17].

Een aandachtspunt bij het gebruik van codegeneratie is wat voor code er wordt gegenereerd. Zo moet bij het gebruik van safety PLC's, de PLC-code aan bepaalde regels voldoen, zoals het vermijden van *loops* in de code en het gebruiken van leverancier specifieke veiligheidsfuncties. Ook traceerbaarheid is hierbij belangrijk, zodat eventuele problemen met de besturingssoftware in het daadwerkelijke object terug kunnen worden herleid naar de code en vervolgens naar de specificatie en het ontwerp, waaronder de plant en requirement modellen (zie ook Sectie 7.3). CGI heeft hier specifiek ook naar deze aspecten gekeken, en komt daarbij tot diverse aanbevelingen (zie Sectie 7.6.4).

Codegeneratie is een vorm van automatisering en verhoogt daarmee de efficiëntie (zie Sectie 7.7). Verder verhoogt het de veiligheid van de besturing, doordat het fouten voorkomt die mensen kunnen maken bij handmatige implementatie (zie Sectie 7.6). Dit geldt uiteraard wel alleen als de codegenerator juist is geïmplementeerd, en alle stappen daadwerkelijk zijn geautomatiseerd (zie Sectie 7.11). Daarnaast moet er voldoende ervaring zijn met codegeneratie, om dit juist toe te passen, en om op de gegenereerde code te kunnen vertrouwen (zie Sectie 7.12).

7.2.4 Stap S4: Verificatie

Traditioneel wordt de besturing geverifieerd door middel van testen, om zo te bepalen of de besturingseisen en andere gewenste eigenschappen juist zijn opgenomen in het ontwerp en de implementatie. Het is tijdrovend om al die testen zo te formuleren dat een computer ze kan uitvoeren. Het is nog tijdrovender als dit handmatig wordt gedaan. Daarnaast kan er altijd maar een eindig aantal tests worden gedaan, waardoor nooit volledige garanties kunnen worden gegeven, en er dus fouten in de specificatie, het ontwerp en de uiteindelijke besturing blijven zitten (zie onder andere Sectie 5.3.2).

Synthese biedt diverse theoretische garanties (zie onder andere Secties 5.6.3 en 7.5). Voor deze garanties is het theoretisch gezien niet nodig deze met verificatie aan te tonen, omdat deze door constructie al worden gegarandeerd. Het is dus niet nodig om bijvoorbeeld met testen te verifiëren dat de eisen zoals ze zijn opgesteld altijd gelden in het gesynthetiseerde besturingsmodel, omdat synthese een correct-door-constructie besturingsmodel synthetiseert waarin altijd aan deze eisen wordt voldaan.

Niet alle eigenschappen worden gegarandeerd door synthese. Zo garandeert synthese bijvoorbeeld (nog) niet de extra eisen waar een gesynthetiseerde supervisor aan moet voldoen voor codegeneratie (zie Sectie

7.2.3). Een ander voorbeeld is *liveness*. De *nonblockingness* eigenschap die synthese garandeert (zie Sectie 5.6.3) is een vrij zwakke vorm van liveness, waarmee wordt gegarandeerd dat uiteindelijk altijd een gemarkeerde toestand bereikt *kan* worden. Voor bijvoorbeeld een enkelbaans brug kan hierbij worden gedacht aan de eigenschap dat elke auto een keer die enkelbaans brug over kan. Er zijn sterkere liveness eigenschappen te specificeren, als dat relevant is. Zo is het mogelijk voor een bepaalde brug te specificeren dat bijvoorbeeld elke auto ook daadwerkelijk een kans krijgt die brug over te gaan. Daarmee wordt er dus een eerlijke keuze gemaakt door de besturing. Dergelijke eigenschappen kunnen alsnog met formele verificatie worden gecontroleerd en bewezen. Een computerprogramma kan deze eigenschappen namelijk controleren, en aangeven of ze gegarandeerd zijn, of juist niet. Geldt een eigenschap niet, dan moet de specificatie (plant of requirements) worden aangepast, net als na het vinden van een fout bij testen. Bij formele verificatie, zoals model checking, wordt hiervoor dan veelal een tegenvoorbeeld gegeven, in de vorm van een reeks van acties die leidt tot een situatie waarin de eigenschap niet geldt. Hiermee geeft de computer dus niet alleen aan dat een eigenschap niet geldt, maar ook waar die niet geldt. Vaak zijn er ook modelleerregels te definiëren, die als ze worden gevolgd ervoor zorgen dat de eigenschappen altijd gelden, of de kans dat ze gelden aanzienlijk groter is [17, 16]. De toepasbaarheid en voordelen van formele verificatie zijn door Rijkswaterstaat samen met de TU/e ook in de praktijk aangetoond in diverse case studies [28, 29].

Hoewel niet altijd strikt noodzakelijk, kan net als bij een traditionele manier van werken ook bij SBE gekozen worden voor traditioneel testen. Dit kan een goede manier zijn om extra zekerheid te verkrijgen en risico's verder te beperken, zeker als er nog niet veel ervaring is met het toepassen van SBE (zie Sectie 7.12) of de gebruikte tools niet zijn gecertificeerd (zie Sectie 7.11.4). Daarnaast kan gebruik worden gemaakt van (een combinatie met) andere formele technieken, zoals model-gebaseerd testen (zie Sectie 5.4.4).

Hiermee heeft SBE een effect op de Verificatie en Validatie (V&V) strategie, en de gebruikte strategie moet (gedeeltelijk) worden herzien om de garanties en mogelijkheden van SBE daarin mee te nemen.

Wat al gegarandeerd is (zie Sectie 7.5) hoeft niet te worden getest. Sommige andere eigenschappen worden niet gegarandeerd door synthese, maar kunnen automatisch geverifieerd worden. Wordt toch testen gebruikt, dan kan bijvoorbeeld model-gebaseerd testen veel werk uit handen nemen. Dit alles bespaart veel tijd (zie Sectie 7.7). Daarnaast kan formele verificatie garanties geven voor alle situaties, terwijl testen slechts voor een eindig aantal situaties uitsluitsel geeft. Daarmee kunnen formele verificatie methoden en de garanties die ze geven ook de kwaliteit van de besturing ten goede komen (zie Sectie 7.6) en de faalkans verlagen (zie Sectie 7.5).

Hoewel formele verificatie dus diverse voordelen biedt, gelden deze alleen als de algoritmes hiervoor ook juist zijn geïmplementeerd in de geautomatiseerde software tools (zie Sectie 7.11). Verder moeten de juiste schaalbare tools ingezet worden, om verificatie van complexe modellen en eigenschappen mogelijk te maken (zie Sectie 7.10). Daarnaast moet er voldoende kennis en ervaring zijn met formele verificatie, om dit juist toe te passen, en om op de resultaten ervan te kunnen vertrouwen (zie Sectie 7.12). Ook is het veranderen van de manier van werken iets wat aandacht verdient (zie Sectie 7.13). Daarnaast is verificatie alleen niet voldoende, en moet validatie niet worden vergeten (zie Sectie 7.2.5). Verder dienen de aanbevelingen van CGI omtrent Verificatie en Validatie (V&V) geadresseerd te worden (zie Sectie 7.6.4).

7.2.5 Stap S5: Validatie

SBE geeft tal van garanties, waaronder dat de besturingseisen juist in het besturingsmodel zijn verwerkt (zie Sectie 7.2.2). Met verificatie kan worden bewezen dat bepaalde andere eigenschappen gelden, zoals dat een besturingsmodel geen livelocks bevat (zie Sectie 7.2.4). Dat wil echter nog niet zeggen dat de gewenste besturing is gerealiseerd. Zo betekent het niet automatisch dat de besturingseisen correct, consistent en compleet zijn. Het kan namelijk zijn dat de verkeerde besturingseisen zijn gemodelleerd, die vervolgens zijn gerealiseerd in het besturingsmodel en de besturingssoftware. Validatie is dus nog altijd van groot belang, om te zorgen dat de gewenste eisen juist zijn geformuleerd, en dat het bestuurde systeem zich gedraagt zoals dat is voorzien. De veiligheid van objecten is namelijk in het geding als de

besturingseisen incompleet of inconsistent zijn, omdat dan onduidelijk is wat er in die situaties die ontbreken of inconsistent zijn moet gebeuren, hoe dit dan wordt afgehandeld in de besturingssoftware, en of er wel (juist) op wordt getest. Dit verhoogt het risico op onveilige situaties, zeker als het veiligheidseisen betreft.

De compleetheid van besturingseisen is lastig te bepalen uit documenten. Vaak worden de in normale situaties voorkomende gevallen wel afgedekt, maar blijven de complexere randgevallen en uitzonderingen onderbelicht (zie ook Secties 5.1 en 5.3.2), terwijl deze zeker zo belangrijk zijn voor de veiligheid. Ook inconsistenties (tegenspraken) zijn dan vaak niet uit te sluiten. En er kunnen simpelweg typefouten gemaakt worden, of teksten worden gekopieerd zonder ze vervolgens aan te passen aan de nieuwe context. Goede experts zullen het aantal fouten reduceren, maar toch kan een volledig foutloze software niet worden gegarandeerd (zie ook Sectie 5.3.2).

Traditioneel wordt verder gebruik gemaakt van testen. Het handmatig testen is vaak zeer tijdrovend. Daarom wordt er in de praktijk vaak met een beperkte set tests gewerkt, om de hoeveelheid werk praktisch te houden. Daarnaast kan testen doorgaans pas plaatsvinden als de implementatie is voltooid. Fouten in specificaties en het ontwerp zijn dikwijls lastiger te vinden later in de ontwikkeling, en het kost dan vaak meer tijd en geld om ze te corrigeren [30, 31].

SBE biedt de mogelijkheid tot simulatie, voordat aan de implementatie wordt begonnen. Daarmee kunnen bijvoorbeeld typefouten vaak eenvoudig gevonden worden, terwijl dat in documenten doorgaans een stuk lastiger is. Maar ook het gedrag kan inzichtelijk gemaakt worden en worden gevalideerd. In tegenstelling tot traditioneel testen kan hiermee de validatie vaak al in een eerder stadium van de ontwikkeling plaatsvinden, waar dat sneller en goedkoper kan. Aangezien de tijd die besteed kan worden aan validatie praktisch altijd eindig is, kan eerdere validatie daarmee resulteren in nog betere kwaliteit, omdat eerdere detectie van fouten er dan voor zorgt dat hetzelfde aantal problemen in een kortere tijd gevonden kunnen worden, of dat in dezelfde tijd meer problemen gevonden kunnen worden.

Een vorm van simulatie is modelsimulatie (zie ook Sectie 5.4.4), waarmee diverse besturingsscenario's kunnen worden bekeken, en het gedrag van de besturing inzichtelijk kan worden gemaakt. Vooraf gedefinieerde scenario's kunnen worden afgespeeld op de modellen, waarmee het in feite traditioneel testen is, maar dan op model niveau, en dus eerder in het ontwikkelproces.

Maar ook interactieve simulatie is mogelijk, om snel diverse scenario's te bekijken en inzicht te verkrijgen, zeker als het wordt gekoppeld met intuïtieve visualisaties. Samen met een iteratief en incrementeel ontwikkelproces maakt dit *rapid prototyping* mogelijk. Hierbij worden korte en snelle iteraties gebruikt om snel inzicht te krijgen in het gedrag van toegevoegd plants en requirements. Elke wijziging in die modellen kan met behulp van automatische synthese met 'een druk op de knop' worden omgezet in een nieuw besturingsmodel, dat dan vervolgens weer gesimuleerd kan worden. Zo kan het model steeds correct, consistent en compleet gemaakt worden, voordat een groter gedeelte van het systeem wordt beschouwd en het aantal situaties complexer wordt.

Naast simulatie puur op modellen kan *hardware-in-the-loop* simulatie gebruikt worden. Hierbij worden de realisatie van de software en een model van het systeem gebruikt. Dit ligt daarmee wat besturing betreft dichter bij de volledige realisatie, terwijl het toch al mogelijk is wanneer het systeem nog niet bestaat. Ook hier kunnen dus al in een eerder stadium problemen worden gevonden. Hardware-in-the-loop simulatie kan plaats vinden met hybride CIF modellen (zie ook Sectie 7.11.1), maar er kan ook gebruik worden gemaakt van een digitale tweeling (*digital twin* in het Engels). Hoewel het ontwikkelen van een digitale tweeling wat meer moeite kost, heeft het als voordeel dat het nog realistischer is, en daarmee intuïtiever is. Dit maakt het mogelijk diegenen die het object daadwerkelijk besturen mee te laten kijken bij de validatie [32, 33]. Daarnaast kan het ook voor trainingsdoeleinden worden gebruikt.

De toepasbaarheid en voordelen van simulatie zijn door Rijkswaterstaat samen met de TU/e ook in de praktijk aangetoond in tal van case studies (zie onder andere Sectie 7.9).

Net als bij een traditionele manier van werken kan ook bij SBE gekozen worden voor traditioneel testen, bijvoorbeeld op de volledige realisatie van het object en besturingssoftware. Testen kan dan complementair zijn, omdat bijvoorbeeld verkeerd op de PLC aangesloten I/O kabels op die manier kunnen worden gedetecteerd. Daarnaast kan testen een goede manier zijn om extra zekerheid te verkrijgen en risico's verder te beperken, zeker als er nog niet veel ervaring is met het toepassen van SBE in het algemeen en simulatie in het bijzonder (zie Sectie 7.12) of de gebruikte tools niet zijn gecertificeerd (zie Sectie 7.11.4). Daarnaast kan gebruik worden gemaakt van (een combinatie met) andere formele technieken, zoals model-gebaseerd testen (zie Sectie 5.4.4).

Hiermee heeft SBE een effect op de Verificatie en Validatie (V&V) strategie, en de gebruikte strategie moet (gedeeltelijk) worden herzien om de garanties en mogelijkheden van SBE daarin mee te nemen.

Met zowel testen als simuleren kan slechts een eindig aantal situaties worden bekeken. In dat opzicht zal validatie nooit 100% garanties bieden (zie ook Sectie 7.5). In de praktijk is het bij zowel een traditionele manier van werken als bij SBE een kwestie van de risico's juist inschatten en daar simuleren/testen waar de risico's het grootst zijn, om zoveel mogelijk risico uit te sluiten tegen minimale inspanning/kosten. Dat SBE het mogelijk maakt in een vroeg stadium te simuleren en daarmee snel en efficiënt problemen te vinden (zie ook Sectie 7.7) is een goede toevoeging op testen, en kan daarmee helpen de kwaliteit nog verder te verbeteren (zie ook Sectie 7.6).

Hoewel simulatie dus diverse voordelen biedt, gelden deze alleen als de simulator hiervoor ook juist is geïmplementeerd (zie Sectie 7.11). Daarnaast moet er voldoende kennis en ervaring zijn met simulatie, om dit juist toe te passen, en om op de resultaten ervan te kunnen vertrouwen (zie Sectie 7.12). Ook is het veranderen van de manier van werken iets wat aandacht verdient (zie Sectie 7.13). Verder dienen de aanbevelingen van CGI omtrent Verificatie en Validatie (V&V) geadresseerd te worden (zie Sectie 7.6.4).

7.2.6 Conclusie rondom ontwikkelproces SBE

Concluderend kan worden gesteld dat SBE in elke stap van het ontwikkelproces tal van voordelen kan bieden. Het maakt het in potentie onder andere mogelijk om eenduidige, complete, consistente en up-to-date specificaties te vervaardigen, en daarmee mogelijk sneller en goedkoper besturingssoftware te realiseren met een zelfde of nog betere kwaliteit.

Hiervoor moet wel de juiste kennis en kunde aanwezig zijn om SBE op de juiste manier toe te passen, en om op de technieken en resultaten ervan te kunnen vertrouwen (zie Sectie 7.12). Hierbij moeten onder andere de juiste, schaalbare technieken worden ingezet om de gewenste efficiëntiewinst te behalen (zie Sectie 7.10). Ook moet de juiste softwareondersteuning worden gebruikt, en moeten die tools juist zijn geïmplementeerd (zie Sectie 7.11). Verder is het veranderen van de manier van werken iets wat de nodige aandacht verdient (zie Sectie 7.13). Daarnaast dienen de aanbevelingen van CGI omtrent het ontwikkelproces met SBE geadresseerd te worden (zie Sectie 7.6.4).

Naast het eenmalige ontwikkelproces voor de eerste versie van de besturingssoftware van een object, is het ook belangrijk het onderhoud en de evolutie van de besturing te beschouwen (zie Sectie 7.3). Daarnaast kunnen modellen gestandaardiseerd en hergebruikt worden voor productfamilies (zie Sectie 7.4).

7.3 Onderhoud en evolutie

De objecten van Rijkswaterstaat staan gedurende lange tijd in de openbare ruimte. In de loop van de jaren moet er wel eens wat onderhoud worden gepleegd, bijvoorbeeld als een object zich niet zoals gewenst gedraagt en die fout moet worden gerepareerd. Ook kan het zijn dat er grotere wijzigingen nodig zijn, zoals wanneer aan andere eisen moet worden voldaan, of er geheel nieuwe functionaliteit gewenst is, bijvoorbeeld bij een renovatie. De besturingssoftware is dus niet altijd constant, maar moet ook worden onderhouden en evolueert over de tijd.

7.3.1 Probleemdetectie

Bij het gebruik van objecten kunnen zaken stuk gaan. Het kan zijn dat een object niet meer juist functioneert, bijvoorbeeld omdat een sensor kapot is, of een motor stuk is. Dit kan bijvoorbeeld door de persoon die het object bedient worden vastgesteld. Er zal gekeken moeten worden wat er aan de hand is. Het kan zijn dat er inderdaad een stuk hardware kapot is, of is versleten, en dat het dan moet worden vervangen. Maar het kan ook zijn dat de besturingssoftware, ondanks alle verificatie en validatie tijdens het ontwerp ervan (zie Secties 7.2.4 en 7.2.5), zich onjuist gedraagt.

De meeste simpele problemen kunnen eenvoudig en snel worden opgelost. Soms is het echter lastiger te achterhalen wat er aan de hand is. Een onderhoudsmonteur kan dan de PLC debuggen, en de toestand van de PLC loggen met functionaliteit die typisch door een SCADA-systeem wordt aangeboden. Daarbij is te zien wat de waardes zijn van de diverse sensoren, zoals deze op een bepaald moment worden doorgegeven aan de PLC. Tevens is te zien welke actuatoren door de PLC op dat moment zijn geactiveerd, en welke niet. Door gedurende een zekere tijd deze ingaande en uitgaande signalen in de gaten te houden, kan het gedrag van de besturingssoftware worden uitgelezen, zeker als de foutieve situatie reproduceerbaar is, dus herhaald kan worden om de gegevens uit te lezen.

Uiteindelijk kan worden achterhaald waar bijvoorbeeld de PLC-code de sensorsignalen verkeerd interpreteert, of waar het verkeerde beslissingen neemt wat betreft het activeren of deactiveren van actuatoren. Hiervoor is het dan wel nodig dat de PLC-code leesbaar is en dus kan worden begrepen door de onderhoudsmonteur, en dat de PLC-code terug traceerbaar is naar het ontwerp en de specificatie van de besturingseisen (zie ook Secties 7.2.3 en 7.6.4). Het op deze manier diagnosticeren van problemen kan echter zowel tijdrovend zijn, als lastig om uit te voeren zonder gedetailleerde kennis van het systeem [34] en de juiste achtergrond (zie ook Sectie 7.12).

Bij het gebruik van modellen, is het echter ook mogelijk uit het gelogde informatie een ‘observatiemodel’ af te leiden. Dat observatiemodel kan dan samen met de al aanwezige modellen gesimuleerd worden, om meer inzicht te krijgen in het geobserveerde gedrag [34]. Verder kan het observatiemodel samen met het al aanwezig plant model gebruikt worden om te achterhalen waar het geobserveerde gedrag afwijkt van wat in het plant model mogelijk is, wat dan of een fout in het systeem of een fout in het plant model blootlegt. Soortgelijk kan het observatiemodel ook worden vergeleken met de supervisor, om illegale dan wel ontbrekende acties in het systeem te detecteren.

De bestaande werkwijze voor het loggen en debuggen van de PLC kan dus ook bij SBE toegepast worden, maar SBE en de model-gebaseerde manier van ontwikkelen biedt ook extra mogelijkheden voor het analyseren van problemen.

7.3.2 Up-to-date consistente artefacten

Een wijziging kan dus het corrigeren van een probleem betreffen, of bijvoorbeeld het toevoegen van nieuwe functionaliteit. Echter, bij het doorvoeren van elke wijziging, wat die ook is, is het belangrijk om te weten wat er nu is, dus wat voor functionaliteit het object heeft, wat de veiligheidseisen zijn, hoe de besturing is ontworpen en geïmplementeerd, etc. Het juist maken van wijzigingen is tenslotte alleen mogelijk als duidelijk is wat er nu is, en welke verandering gewenst is.

Bij traditioneel ontwikkelen, gebruik makend van documenten, worden bij het ontwerp zaken die uit de specificatie ambigu zijn op een bepaalde manier geïnterpreteerd. En bij de implementatie worden zaken die ambigu zijn in het ontwerp op een bepaalde manier geïnterpreteerd. Er worden dus keuzes gemaakt ten opzichte van de resultaten van eerdere stappen van het ontwikkelproces. Dat is prima, maar dan is het wel zaak om die keuzes ook weer expliciet te maken, en dus te verwerken in nieuwe versies van de documenten die in de vorige stap zijn geschreven. Op die manier worden de keuzes vastgelegd en niet later alsnog weer anders geïnterpreteerd. Dit wordt echter dikwijls overgeslagen, waardoor specificatie, ontwerp en implementatie niet meer consistent zijn. Als er later nog meer zaken veranderen, dan worden

de specificatie en implementatie langzaam aan nog meer inconsistent. Dit maakt het lastig om te weten wat er is geïmplementeerd, wat de relatie is met de specificaties en het ontwerp, en wat er in bijvoorbeeld de implementatie moet worden aangepast als een besturingseis moet worden aangepast of een nieuwe functie moet worden toegevoegd.

Daarentegen stelt SBE, als vorm van model-gebaseerd ontwikkelen, modellen centraal. Het zijn de modellen die worden aangepast als deze functioneel onjuist zijn, als er inconsistenties in zitten, of als er nieuwe functionaliteit gewenst is. Analyses, synthese en codegeneratie werken allemaal op de modellen. In tegenstelling tot documentatie, worden de modellen dus continu aangepast en daarmee onderhouden. Ze blijven up-to-date, omdat ze aan de basis staan van de ontwikkeling gedurende de gehele levenscyclus van het systeem, zoals ontwerp, realisatie en onderhoud. Hiermee is er een duidelijk en consistent beeld van de diverse artefacten, zoals de eisen, het ontwerp en de realisatie. Dit kan mogelijk wel extra inspanning vereisen, is mogelijk geen volledige vervanging van documentatie, en de ontwerpkeuzes moeten nog altijd worden vastgelegd.

7.3.3 Manier van werken

De ideale manier van werken is bij SBE echter wel fundamenteel anders dan een traditionele manier van werken. Hoewel het nog altijd mogelijk is rechtstreeks de gegenereerde code aan te passen, is dat in principe bij het maken van een wijziging niet meer de bedoeling. Het zijn altijd de plant en requirement modellen die aangepast dienen te worden. De modellen worden dan opnieuw gesynthetiseerd, geverifieerd en gevalideerd, en er wordt nieuwe code gegenereerd.

Hierbij vindt het wijzigen van de besturingssoftware bij SBE typisch gestructureerd plaats volgens het normale ontwikkelproces (zie Sectie 7.2), zodat er geen stappen worden overgeslagen. En is er daarbij dus ook computerondersteuning voor al deze stappen, net zoals bij de eerste keer dat de besturingssoftware was ontwikkeld. Met voldoende kennis en kunde (zie Sectie 7.12), en bij het gebruik van voldoende schaalbare technieken (zie Sectie 7.10), moet dit dan efficiënt kunnen plaatsvinden (zie Sectie 7.7), omdat de ontwerpcyclus kort gehouden kan worden (zie Sectie 7.2.5). Daarmee gelden de voordelen van SBE tot op zekere hoogte ook voor onderhoud en evolutie.

Het is, zoals gezegd, wel een fundamenteel andere manier van werken. Voor elke wijziging moet je door alle stappen heen, en dit vereist betrokkenheid van de personen die de juiste kennis en kunde hebben (zie Sectie 7.12). Daarbij is het voor Rijkswaterstaat belangrijk dit proces verder uit te werken, waaronder wie er verantwoordelijk is voor dit soort aanpassingen, wie ze uitvoert, en wie het coördineert. Hierbij kan het centraal SBE team wellicht een rol spelen (zie ook Secties 6.2.8 en 7.12). Verder dient er meer ervaring opgedaan te worden. Hierbij dienen dan alle betrokken partijen van ontwerpers tot onderhoudsmonteurs te worden betrokken. Ook is het relevant dan niet alleen te kijken naar de evolutie van een enkel object, maar ook naar de productfamilie waar dat object een deel van uitmaakt, hoe het proces van wijzigingen in die situatie verandert, en hoe de rollen en verantwoordelijkheden dan liggen (zie ook Sectie 7.4). Daarnaast dienen de aanbevelingen van CGI omtrent wijzigingen geadresseerd te worden (zie Sectie 7.6.4).

7.4 Hergebruik en standaardisatie

Het gebruik van SBE biedt diverse voordelen bij het ontwikkelen van de besturingssoftware voor een enkel object (zie onder andere Sectie 7.2). Het is echter de visie van Rijkswaterstaat om te komen tot productfamilies, en herbruikbare modules (zie Sectie 6.1). Het zou bij SBE dan gaan om modules met herbruikbare standaard plant en requirement modellen. Deze kunnen modulair gespecificeerd worden, door elk fysiek element en elke besturingseis apart te modelleren. Dat kan bijvoorbeeld op het niveau van individuele sensoren en actuatoren, maar ook op het niveau van deelsystemen bestaande uit meerdere sensoren en actuatoren. Vervolgens kunnen ze gecombineerd worden om op die manier voor een specifiek object een set modellen samen te stellen, volgens het *configure-to-order* principe (zie ook Sectie 6.1). Bij voldoende overeenkomst tussen objecten in dezelfde productfamilie zou een significant gedeelte van de plant en requirement modellen dan hetzelfde kunnen blijven, en bijvoorbeeld een aantal vaste plant en

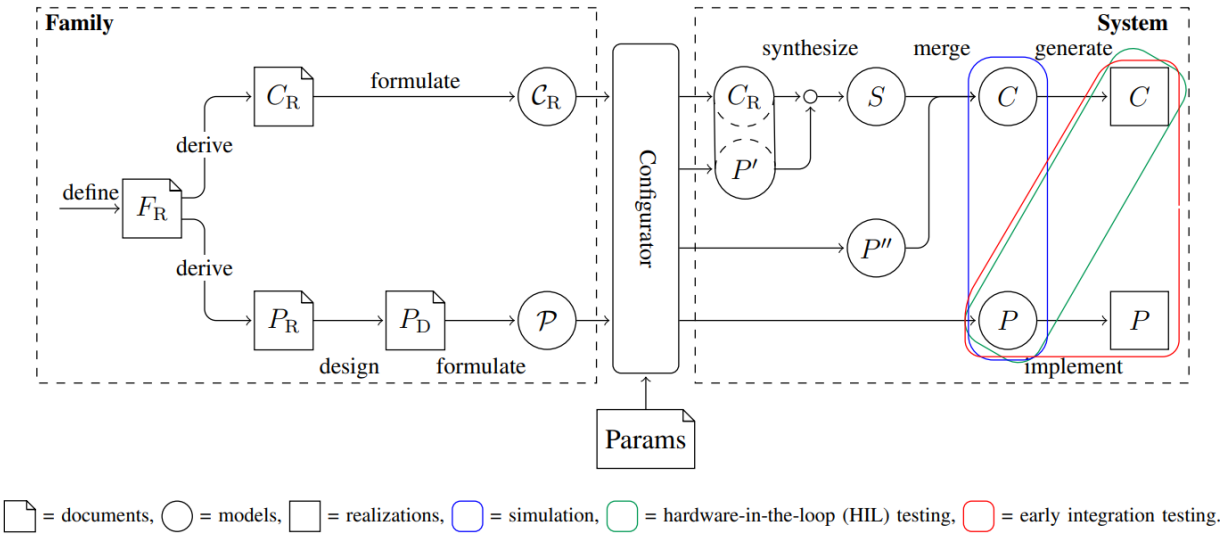
requirement modellen kunnen worden toegevoegd afhankelijk van de gewenste functionaliteit en situatie waarin het object zich bevindt. Hierbij kan worden gedacht aan de door het Centraal SBE team te ontwerpen Generiek Model Sluis (GMS) en Generiek Model Brug (GMB), zie ook Sectie 6.2.8.

De apart gespecificeerde plants en requirements kunnen dus in een bibliotheek met herbruikbare gestandaardiseerde ‘bouwblokken’ worden gestopt (zie ook Sectie 5.6.9). De modulariteit die SBE ondersteunt (zie Sectie 5.6.8) stelt de modelleur in staat om eenvoudig nieuwe specificaties op te bouwen uit bestaande en bewezen kwalitatief goede bouwblokken. Te verwachten valt dat deze modellen na verloop van tijd zich bewijzen in de praktijk, door het gebruik voor meer en meer objecten. Hiermee neemt het vertrouwen in de juistheid verder toe. In plaats van telkens voor elk object alles opnieuw uit te denken, kan gebruik worden gemaakt van modellen die zich hebben bewezen. Dit kan de veiligheid ten goede komen, onder andere omdat hiermee wordt voorkomen dat steeds (door verschillende personen) dezelfde fouten worden gemaakt (zie ook Sectie 7.6). Daarnaast leidt het tot meer uniformiteit, door het hergebruik van (gestandaardiseerde) deelmodellen en bouwblokken. Verder verbetert het de efficiëntie (zie ook Sectie 7.7), omdat niet steeds het wiel opnieuw uitgevonden hoeft te worden met maatwerk, want veel werk is dus al eerder gedaan en kan worden hergebruikt.

SBE gaat hier een stap verder dan traditioneel ontwikkelen, MBE en VBE, omdat bij SBE doorgaans gewerkt wordt met kleine plant en requirement specificaties, bijvoorbeeld elke sensor en actuator apart, en elke besturingseis apart. Dit geldt ook voor binnen een bouwblok. Daardoor is het bijvoorbeeld eenvoudiger mogelijk om een deel van een bouwblok (bijvoorbeeld een enkele sensor) te vervangen, of een extra besturingseis toe te voegen. Bij een bouwblok met een handmatig gemodelleerd besturingsmodel zit dit doorgaans meer verweven in het gehele model, zelfs als er bijvoorbeeld een hiërarchische structuur wordt gebruikt.

Dat hergebruik, standaardisatie en configure-to-order met SBE mogelijk zijn, is reeds aangetoond in de samenwerking tussen Rijkswaterstaat en de TU/e. Zo is gekeken naar verkeersmanagement systemen, te weten de ongeveer 6.000 wegkantsystemen die juiste beelden op de matrixborden boven snelwegen zetten als reactie op metingen van de detectoren in het wegdek en opdrachten uit de verkeerscentrale. Met behulp van een prototype configurator is aangetoond dat het in potentie mogelijk is een model te genereren voor elk van die systemen. Concreet is het als proof-of-concept toegepast op een enkel representatief wegkantsysteem, en daarvoor is vervolgens ook vanuit het gegenereerde model een supervisor gesynthetiseerd [35]. Hiervoor zijn in de configurator enkel de parameters binnen de productfamilie ingesteld, zoals het aantal rijbanen, het aantal detectoren, en de snelheidslimieten. De configurator genereert dan automatisch het model met de juiste plants en de juiste en relevante requirements, uit een vooraf eenmalig gemodelleerd geparametriseerde model. Voor het modelleren van het geparametriseerde model is gebruik gemaakt van CIF (zie ook Sectie 7.11.1).

Figuur 9 toont het configurator-gebaseerd ontwikkelproces dat hierbij is gebruikt. Vanuit het requirements document (document F_R) voor de productfamilie worden plant requirements (document P_R) en besturingseisen (document C_R) afgeleid. De besturingseisen worden formeel gemodelleerd (model C_R). Vanuit de plant requirements wordt de plant ontworpen (document P_D) en gemodelleerd (model $\mathcal{P}$) voor de gehele productfamilie. Vanuit de gekozen waardes voor de parameters genereert de configurator dan de modellen van de plant (model P') en requirements (model C_R) voor synthese van de supervisor (model S). Het genereert ook de delen van de modellen (model P'') die niet worden meegenomen in de synthese, bijvoorbeeld de tijdsgerelateerde aspecten. De combinatie van deze twee modellen (model C) is het *controller* model van waaruit de besturingssoftware kan worden gegenereerd (realisatie C). Verder genereert de configurator ook een model (model P) voor simulatie (blauw vak), hardware-in-the-loop simulatie/testen (groen vak) en integratie testen (rood vak).



Figuur 9 – Het gebruikte proces voor configurator-gebaseerd ontwikkelen voor een productfamilie, in het Engels uit [35].

Dit is echter niet alleen voor wegkantsystemen gedaan. Met een ander prototype is aangetoond dat het mogelijk is automatisch modellen te genereren voor 22 tunnels, wederom door enkel de parameters binnen de productfamilie in te stellen [36]. Het betreft dan onder andere het aantal verkeersbuizen, nooduitgangen en waterdrainagesystemen. Maar ook de diverse eigenschappen daarvan, zoals de rijrichting en afmetingen van een tunnelbuis en of de tunnelbuis al dan niet hoogtedetectie heeft. De gegenereerde modellen kunnen dan vervolgens wederom worden gebruikt voor synthese en simulatie. Bij dit werk is aangetoond dat als de verschillende objecten in een productfamilie voldoende overeenkomsten hebben, het de efficiëntie verhoogt omdat er amper tot niets hoeft te worden gemodelleerd (zie ook Sectie 7.7). Daarnaast verlaagt het de SBE kennis die nodig is, omdat er slechts hoeft te worden geconfigureerd in plaats van dat er uitgebreid moet worden gemodelleerd. Het modelleren verplaatst zich dan naar het ontwikkelen van het geparametriseerde model voor de gehele productfamilie, wat kan worden ontwikkeld door een kleiner team en daarmee voor de gehele productfamilie minder personen met expertkennis in SBE vereist (zie ook Sectie 7.12). Wel zijn de geparametriseerde modellen waarschijnlijk complexer dan niet-geparametriseerde modellen, wat de kans op fouten zou kunnen vergroten. De schaalbaarheid was bij dit experiment voldoende om “binnen minuten” per tunnel een besturing te synthetiseren (zie ook Sectie 7.10).

Verder is dit ook toegepast voor een familie van 17 beweegbare bruggen over kanalen in het zuiden van Nederland, bestaande uit basculebruggen, hefbruggen en een draaibrug [37]. Hierbij is een iets andere aanpak gebruikt. Met behulp van een interactieve configurator kan uit een bibliotheek van modules op een visuele manier een brug worden samengesteld. Deze kan dan worden geconfigureerd. De modellen voor synthese en simulatie worden wederom automatisch gegenereerd. Synthese duurde hierbij slechts “enkele seconden” per object.

Voor sluizen is gekeken naar de keuzes bij het ontwerp voor het productplatform, met als doel om via een gestructureerd proces tot een portfolio van sluisconfiguraties te komen [38, 39, 40]. Daarbij is onder andere gekeken naar het selecteren en standaardiseren van de componenten, het analyseren van de variëteit in het portfolio van mogelijke component combinaties om te komen tot alleen de geldige combinaties, en het afwegen van alternatieve ontwerpen. Synthese is nog niet toegepast.

Deze experimenten tonen aan dat de standaardisatie en hergebruik die Rijkswaterstaat voor ogen heeft (zie Sectie 6.1) in potentie mogelijk zijn. Het kan de efficiëntie verhogen, leiden tot nog betere kwaliteit, en de drempel voor toepassing verlagen door het verminderen van de benodigde expertkennis.

Rijkswaterstaat heeft dit ook reeds zelf ondervonden bij het modelleren van foutgedrag van een sluis. Daarbij is voor bepaald foutgedrag een specifieke besturingseis geformuleerd. Bij een volgende sluis, zeker een uit dezelfde productfamilie, kan dan worden bekeken of die daar ook zou moeten worden meegenomen. Zo kan de kennis die is opgedaan bij een object, en die expliciet is vastgelegd in de modellen, bij het volgende object worden hergebruikt.

Synthese voor productfamilies is dus reeds toegepast voor wegkantsystemen, tunnels en bruggen, maar nog niet voor sluizen. Hoewel hier voor een groot deel soortgelijke uitkomsten te verwachten zijn, is het toch aan te raden het ook voor dit type object uit te proberen, om er meer zekerheid over te krijgen. Verder zijn de eerste experimenten bemoedigend, maar moet het een en ander nog wel verder worden uitgewerkt voordat dit grootschalig kan worden toegepast. Zo betreffende de gebruikte configuratoren slechts prototypes, en zijn nog geen *fault-tolerant* supervisors gesynthetiseerd, waarvoor synthese mogelijk minder efficiënt is (zie ook Sectie 7.10). Verder moet onder andere, zoals Rijkswaterstaat zelf aangeeft (zie Sectie 6.2.8) en ook het BIT adviseert [1], het geparametriseerde model voor de productfamilie wel goed beheerd worden. Hierbij dient verder goed gekeken te worden naar het onderhoudsproces en de evolutie van de modellen (zie ook Sectie 7.3). Verder dienen de aanbevelingen van CGI omtrent productfamilies geadresseerd te worden (zie Sectie 7.6.4).

7.5 Garanties van SBE

Het gebruik van formele methoden, zoals synthese, geeft bepaalde garanties. Hier wordt per stap van het ontwikkelproces (zie Sectie 7.2) besproken welke garanties SBE wel en niet geeft. Dit beïnvloedt de veiligheid van de besturing (zie ook Sectie 7.6). In elke stap van het ontwikkelproces kunnen namelijk fouten gemaakt worden, waardoor de uiteindelijke besturingssoftware niet juist functioneert, en dus de veiligheid van het infrastructurele object in het geding is, zeker als het fouten met betrekking tot veiligheidseisen betreft. Bepaalde garanties kunnen dergelijke fouten verminderen of voorkomen, en daarmee de veiligheid van de besturing en het object verhogen.

Voor het specificeren van de besturingseisen, wordt bij SBE gebruik gemaakt van modelleren om plant en requirement modellen op te stellen. Voor dergelijke modellen is er, in tegenstelling tot documenten, de garantie dat deze een eenduidige wiskundige betekenis (semantiek) hebben (zie ook Sectie 7.2.1). Op deze manier kunnen dus bijvoorbeeld de besturingseisen eenduidig worden geïnterpreteerd, wat ook de communicatie verbetert (zie ook Sectie 7.7). Door ook de exceptionele gevallen mee te nemen, dat bijvoorbeeld sensoren stuk kunnen zijn, ligt ook voor deze situaties het gedrag eenduidig vast.

Het ontwerp hoeft bij SBE niet handmatig te worden gemaakt. Via besturingssynthese kan automatisch een correct-door-constructie supervisor model worden gegenereerd (zie Sectie 7.2.2). Deze voldoet in alle situaties aan de gestelde eisen, ook voor lastig te doorgronden situaties, in uitzonderlijke situaties (indien ze mee zijn gemodelleerd), en voor situaties die pas later tot problemen leiden. Synthese garandeert dus dat het gegenereerde supervisor model door constructie in alle situaties aan alle gestelde eisen voldoet. Dit wordt *safety* genoemd in supervisor control theorie. Dit is echter wel een een andere notie van veiligheid dan in bijvoorbeeld de machinerichtlijn (zie ook Secties 5.6.3 en 7.6.4). Het is één van de vier garanties van synthese die wiskundig zijn bewezen, naast de wat technischere *controllability*, *non-blockingness* en *maximal-permissiveness*/*minimal-restrictiveness* garanties [19, 20] (zie ook Sectie 5.6.3). Synthese maakt een handmatig ontwerp van de besturing in feite overbodig door het automatisch genereren van een correct-door-constructie besturingsmodel, en geeft daarbij dus vanuit de theorie vier specifieke garanties.

Hier ligt een relatie met de door Rijkswaterstaat gehanteerde TOPAAS methode [41] (zie ook Sectie 7.6.2). Dit is een structurele aanpak voor faalkansanalyse voor software intensieve systemen. Het gebruik van formele testtechnieken, zoals model-gebaseerd testen, wordt in de TOPAAS methode gezien als procesmatig beter dan ‘common practice’ en leidt daarmee tot lagere faalkansen, zeker als een hoge dekkingsgraad van het gedrag van het systeem kan worden bereikt. Hiermee kan dus de faalkans van (componenten van) software systemen worden verkleind. Hoewel synthese in de TOPAAS methode niet wordt genoemd, worden formele methoden in het algemeen genoemd als een manier om aan te tonen dat

logische softwarefouten afwezig zijn, of waar die fouten zitten. Daarbij wordt aangegeven dat als formele methoden strikt worden toegepast en er rigoureuze reviews plaatsvinden, dit het gevolg heeft dat de faalkans ten gevolge van logische softwarefouten “de absolute nul benadert”. Synthese, met zijn correct-door-constructie garanties, is een voorbeeld van een formele methode. De automatisering via een computer voorkomt fouten die mensen (kunnen) maken als ze handmatig een complexe taak moeten uitvoeren, en de computer kan alle mogelijke toestanden doorrekenen en daarmee dus een volledige dekkingsgraad verkrijgen ten opzichte van de gespecificeerde modellen en eisen. De invloed van het gebruik van SBE op de faalkansen zoals die bekeken worden in de TOPAAS methode is echter niet onderzocht. Het is wellicht interessant om een update van het document dat de TOPAAS methode beschrijft te overwegen, aangezien dit document in 2011 is geschreven, en diverse formele methoden de laatste jaren en stuk volwassener zijn geworden, waaronder besturingssynthese [42, 5].

Voor het implementeren van de besturing in PLC-code is het belangrijk de intrinsieke verschillen tussen besturingsmodellen en PLC-code te kennen (zie ook Sectie 7.2.3). In een state machine worden bijvoorbeeld acties om meerdere actuatoren te activeren achter elkaar uitgevoerd, terwijl in PLC-code diverse actuatoren aan het einde van een cyclus tegelijk actief gemaakt worden. Er zijn diverse extra eigenschappen die moeten gelden voor het besturingsmodel om er PLC-code uit te kunnen genereren die het systeem correct laat functioneren, te weten *confluence, finite response* en *nonblocking under control* (zie Sectie 7.2.3). Deze eigenschappen kunnen met de computer formeel worden geverifieerd (zie Sectie 7.2.4), om zo te garanderen dat ze gelden.

Ook de realisatie wordt bij SBE geautomatiseerd, door gebruik te maken van codegeneratie (zie ook Sectie 7.2.3). Dit voorkomt fouten in de interpretatie en implementatie van de specificatie en het ontwerp die bij handmatig programmeren kunnen worden gemaakt. Een correcte codegenerator garandeert dat de gegenereerde code correct-door-constructie is, dus dat deze zich precies zo gedraagt als is vastgelegd in het controller model waarvoor de code is gegenereerd. De correct-door-constructie garanties van codegeneratie kunnen gezien worden als een verder voorbeeld van formele technieken. In de literatuur wordt echter niet altijd formeel bewezen dat de code zich hetzelfde gedraagt als het besturingsmodel waar het uit is gegenereerd [17]. Er zijn echter wel codegeneratoren die zijn gecertificeerd (zie Secties 7.11.2 en 7.11.4).

Synthese geeft vier theoretische garanties, zoals dat het besturingsmodel aan de gestelde eisen voldoet. Overige eigenschappen, die (nog) niet door synthese worden gegarandeerd, zoals striktere *liveness* eigenschappen, kunnen formeel worden geverifieerd. Hiermee kan, in tegenstelling tot testen, worden gegarandeerd dat ze in alle gevallen gelden (zie ook Sectie 7.2.4). De formele verificatie, zoals model checking, is een verder voorbeeld van een formele techniek.

Naast verificatie is ook validatie essentieel. Bij SBE kan hiervoor gebruik worden gemaakt van zowel modelsimulatie als hardware-in-the-loop simulatie (zie Sectie 7.2.5). Simulatie kan worden gezien als een formele techniek waarmee logische fouten gevonden een aangewezen kunnen worden. Hoewel simulatie, net als testen, geen 100% garantie kan geven op afwezigheid van fouten, kan de eerdere detectie van fouten er wel voor zorgen dat in dezelfde tijd meer problemen gevonden worden, waarmee de kwaliteit van de besturing aanzienlijk verbeterd kan worden.

Concluderend kan gesteld worden dat het SBE proces diverse garanties biedt die de kwaliteit en daarmee de veiligheid van de besturing ten goede kunnen komen, zeker voor complexe objecten met vele gerelateerde veiligheidseisen (zie ook Sectie 7.6). Hierbij is sprake van vergaande inzet van formele methoden, en het inzetten van formele methoden kan volgens de auteurs van de TOPAAS methode leiden tot faalkansen die de absolute nul kunnen benaderen. Deze gelden echter alleen als de theoretische algoritmes ook daadwerkelijk juist zijn geïmplementeerd in de geautomatiseerde software tools (zie Secties 7.11.3 en 7.11.4), en de formele methoden en software tools juist worden toegepast (zie Sectie 7.12).

7.6 Veiligheid en betrouwbaarheid

Veiligheid is zeer belangrijk voor kritieke infrastructurele systemen zoals die van Rijkswaterstaat. In elke stap van het ontwikkelproces kunnen er zaken fout gaan waardoor de uiteindelijke besturingssoftware niet juist functioneert, en dus de veiligheid van het infrastructurele object in het geding is. Veiligheid is een complex onderwerp. In deze sectie wordt het aspect veiligheid vanuit een aantal invalshoeken bekeken, zonder daarbij compleet te zijn in alles wat er bij komt kijken om een veilige besturing te ontwerpen en te realiseren:

- **Veiligheid in termen van besturingseisen** (Sectie 7.6.1): Hierbij gaat het er om hoe SBE bijdraagt aan het verkrijgen van specificaties van de juiste besturingseisen, en dan met name de veiligheidseisen, en hoe het kan helpen om te zorgen dat de besturingssoftware voldoet aan die besturingseisen. Dit puur vanuit het perspectief van de software, waarbij ervan uit wordt gegaan dat de hardware juist functioneert.
- **Faalkans van de besturingssoftware** (Sectie 7.6.2): Hierbij gaat het erom hoe waarschijnlijk het is dat de besturingssoftware faalt, en wat de impact van het gebruik van SBE daarbij is.
- **Betrouwbaarheid van de hardware** (Sectie 7.6.3): Hierbij gaat het er om in hoeverre het systeem veilig is als de hardware faalt, en wat SBE daar aan bijdraagt.
- **Wet- en regelgeving rondom veiligheid** (Sectie 7.6.4): Hierbij gaat het om de wet- en regelgeving omtrent veiligheid en gezondheid waar de objecten van Rijkswaterstaat aan moeten voldoen, en de relatie daarvan met SBE.

7.6.1 Veiligheid in termen van besturingseisen

In het begintraject van de ontwikkeling van een besturing voor een object worden de besturingseisen gespecificeerd (zie ook Sectie 7.2.1). Als deze ambigu, incompleet, conflicterend of simpelweg onjuist zijn, dan heeft dit directe invloed op de veiligheid van het systeem, zeker als dit veiligheidseisen betreffen. De besturingseisen staan namelijk aan de basis van de gehele verdere ontwikkeling van de besturingssoftware. Als de besturingseisen bijvoorbeeld onjuist worden gespecificeerd, is de kans aanwezig dat ook het ontwerp en de implementatie onjuist zullen zijn (zie ook Sectie 5.3.2). SBE stelt formele modellen met een eenduidige betekenis centraal gedurende het gehele ontwikkelproces (zie ook Secties 5.4 en Sectie 7.2), die door personen met verschillende achtergronden begrepen kunnen worden, en daarmee de kloof tussen specificatie en implementatie verkleinen. Dit verbetert onder andere de communicatie tijdens alle stappen van het ontwikkelproces (zie ook Sectie 7.7). Het voorkomt daarmee veel onduidelijkheden, misinterpretaties, fouten en discussies, zeker als ook exceptionele gevallen worden meegenomen tijdens het ontwikkelproces. Het expliciet bespreken en modelleren van het gewenste gedrag van alle mogelijke situaties levert een compleet en consistent beeld op van wat de besturing moet doen. Hiermee heeft het logischerwijs ook direct een grote impact op de veiligheid van het systeem. Per slot van rekening, hoe kan men een juist besturing realiseren als al niet goed vaststaat wat de besturing überhaupt moet doen?

Een andere aspect dat de veiligheid ten goede komt, is automatisering. SBE heeft als doel zoveel mogelijk stappen van het ontwikkelproces te automatiseren, door gebruik te maken van formele methoden, waaronder automatische besturingssynthese, automatische formele verificatie, automatische codegeneratie en geautomatiseerd testen door middel van model-gebaseerd testen (zie ook Sectie 7.2). Alles wat is geautomatiseerd kan met ‘een druk op de knop’ worden uitgevoerd door een computer, die daarbij in tegenstelling tot mensen bij de uitvoering geen fouten maakt. Want hoe goed een expert ook is in zijn of haar werk, iedereen kan fouten maken, en bij het uitvoeren van complexe taken is het uitsluiten daarvan praktisch gezien vrijwel onmogelijk (zie ook Sectie 5.3.2). Een (foutloze) computer maakt die fouten dus niet. Zo kan synthese met ‘een druk op de knop’ een correct-door-constructie besturingsmodel genereren, dat in alle situaties aan alle geformuleerde besturingseisen voldoet. En codegeneratie kan met een druk op correct-door-constructie code genereren, die zich precies zo gedraagt als in het besturingsmodel is vastgelegd. Dit heeft potentieel echter wel een keerzijde (zie Sectie 7.7).

De automatisering levert ook efficiëntiewinst op (zie Sectie 7.7). SBE stelt ontwikkelaars daarmee in staat te focussen op *wat* de besturing moet doen in plaats van *hoe* die dat moet doen. Dit betekent dat de tijd kan worden besteed precies aan die zaken die voor de kwaliteit en daarmee veiligheid van de besturingssoftware essentieel zijn: het juist specificeren van de juiste besturingseisen.

SBE maakt verder onder andere rapid prototyping, modelsimulatie, hardware-in-the-loop simulatie, en model-gebaseerd testen mogelijk (zie Sectie 7.2.5). Hoewel validatie geen 100% juiste besturingseisen kan garanderen (zie ook Sectie 7.5), maken deze extra opties het wel mogelijkheid om meer inzicht te krijgen in het gedrag van het systeem en de gemodelleerde besturingseisen, en geeft het de mogelijkheid meer te testen dan met traditionele aanpakken. Hiermee kan de kwaliteit van de besturingseisen, en het vertrouwen erin, wel aanzienlijk worden vergroot. Zoals besproken heeft het daarmee in potentie ook een grote positieve impact op de uiteindelijke veiligheid van het systeem.

De formele methoden die bij SBE worden gebruikt hebben, in tegenstelling tot validatie, wel het voordeel dat ze bepaalde garanties geven (zie ook Sectie 7.5). Zo geeft besturingssynthese de garantie dat het gesynthetiseerde besturingsmodel in alle situaties aan alle gestelde besturingseisen voldoet (zie ook Sectie 7.2.2). En formele verificatie, bijvoorbeeld door middel van model checking, kan diverse eigenschappen 100% bewijzen (zie ook Sectie 7.2.4). Dit soort technieken geven daarmee meer garanties dan bijvoorbeeld traditioneel testen. Dit kan leiden tot besturingseisen, besturingsmodellen, en besturingssoftware van nog betere kwaliteit, en tot nog meer vertrouwen in de besturing, waarmee de veiligheid van het object kan worden verhoogd [43].

Daarnaast maakt SBE het mogelijk om aan vergaand hergebruik te doen voor productfamilies, via gestandaardiseerde modellen die gedurende langere tijd zijn ontwikkeld, en die zich in het verleden reeds hebben bewezen (zie ook Sectie 7.4). Hergebruik en standaardisatie kunnen daarmee eerder gemaakte fouten voorkomen, omdat het wiel niet telkens opnieuw hoeft te worden uitgevonden. Ook dit kan de kwaliteit en dus veiligheid van de uiteindelijke besturing ten goede komen [37].

Verder wordt voor onderhoud en evolutie bij SBE hetzelfde proces gevolgd als voor de eerste ontwikkeling van de besturingssoftware (zie ook Sectie 7.3). Daarmee heeft het ook daar al de bovenstaande voordelen.

Samenvattend kan SBE op tal van manieren de kwaliteit van de besturingseisen en de veiligheid van de besturing verhogen, door met eenduidige modellen communicatie te verbeteren en misinterpretaties, fouten en discussies te voorkomen, door tal van zaken te automatiseren en daarmee menselijk fouten te reduceren, door met efficiëntiewinst meer tijd te geven voor zaken die de veiligheid bevorderen, door met extra validatieopties meer fouten te vinden, door meer formele correctheidsgaranties te geven dan traditionele aanpakken, door met hergebruik en standaardisatie eerder gemaakte fouten te voorkomen, en door dit niet alleen bij de normale ontwikkeling maar ook bij onderhoud en evolutie mogelijk te maken.

Dit zijn niet alleen theoretische voordelen. Veel van deze voordelen zijn bijvoorbeeld aangetoond in de samenwerking met de TU/e (zie onder andere Sectie 7.9). Studenten van de TU/e blijken in staat te zijn om, met slechts beperkte kennis van de systemen van Rijkswaterstaat, in korte tijd een goede besturing te maken (zie ook Sectie 7.13). Maar ook in de literatuur zijn dit soort kwaliteitsvoordelen bekend. Zo zorgde de introductie van model-gebaseerd ontwikkelen bij Motorola voor een reductie van 1,2 keer tot 4 keer in aantallen software fouten [44].

Concluderend kan worden gesteld dat met SBE niet alleen theoretisch, maar ook in de praktijk, veel fouten kan voorkomen, en daarmee de veiligheid van objecten kan verhogen. Hiervoor moet echter een aantal andere aspecten niet uit het oog worden verloren. Zo moeten dan onder andere wel de juiste tools worden ingezet, die juist zijn geïmplementeerd (zie Sectie 7.11), moet de juiste kennis en kunde aanwezig zijn om deze juist in te zetten (zie Sectie 7.12), en moet het veranderen van werkwijze niet worden onderschat (zie Sectie 7.13).

7.6.2 Faalkans van de besturingssoftware

Rijkswaterstaat hanteert de TOPAAS methode [41], een structurele aanpak voor faalkansanalyse van software intensieve systemen (zie ook Sectie 7.5). De kern van deze methode is dat software in modulen kan worden opgedeeld en dat faalkansen per module kunnen worden bepaald met behulp van foutbomen.

De TOPAAS methode kan worden ingezet als de hoeveelheid informatie of de benodigde inspanning het niet toelaat dat betrouwbaardere modellen worden toegepast. Het gebruik van formele methoden wordt in de TOPAAS methode als direct alternatief genoemd voor de TOPAAS methode zelf, waarbij “als deze methoden strikt worden toegepast én er rigoureuze reviews op worden uitgevoerd, de faalkans ten gevolge van logische softwarefouten de absolute nul benadert”.

De TOPAAS methode levert momenteel alleen bij het gebruik van ‘formele testtechnieken’ lagere faalkansen op, waarbij de methode het gebruik van model-gebaseerd testen op een geverifieerd model noemt. Het gebruik van dergelijke formele methoden wordt daarbij procesmatig als beter gezien dan ‘common practice’. Een dergelijk geverifieerd model kan worden verkregen met VBE, waar bijvoorbeeld met behulp van formele verificatie door middel van model checking wiskundig bewezen kan worden dat de logica van een component niet kan falen. Verder kan bij SBE kan met behulp van supervisor synthese zelfs een correct-door-constructie besturingslogica worden gegenereerd. SBE behelst wat dat betreft vergaande inzet van formele methoden (zie ook Secties 7.2 en 7.5).

7.6.3 Betrouwbaarheid van de hardware

Voor de betrouwbaarheid van de besturing is het niet alleen noodzakelijk dat de juiste besturingseisen zijn gespecificeerd en dat de besturingssoftware voldoet aan die besturingseisen. Het is ook belangrijk dat het systeem veilig blijft als hardware faalt. Hierbij kan worden gedacht aan een defecte sensor, actuator, of kabel (zie onder andere Secties 5.3.2). In *safety engineering* wordt hier doorgaans gewerkt met *failsafe* gedrag. Daarbij wordt aangenomen dat het systeem wel kan falen, maar dat het dan in een bekende veilige toestand terecht moet komen. Hierbij kan gedacht worden aan een alarm dat niet alleen afgaat als er expliciet een onveilige situatie wordt gedetecteerd, maar dat ook afgaat als de detectie van onveilige situaties niet mogelijk is vanwege een falende detectiemodule.

SBE maakt het mogelijk ook voor die situaties eenduidig vast te leggen wat de besturing moet doen, dus hoe het op dat soort situaties moet reageren. En die situaties kunnen in alle stappen van specificatie en synthese, tot verificatie, validatie en codegeneratie worden meegenomen (zie ook Sectie 7.9). Hiermee kan de gesynthetiseerde uiteindelijke besturing dus expliciet met dit soort situaties omgaan. Ook kan geleerd worden van foutafhandeling bij andere objecten, en kan de foutafhandeling worden gestandaardiseerd (zie ook Sectie 7.4).

Uiteraard kan ook de PLC waarop de besturingssoftware draait defect raken, wat vooral de beschikbaarheid beïnvloedt. Een mogelijke oplossing hiervoor is redundantie, waarbij meerdere PLC’s worden gebruikt en als een daarvan uitvalt en dus geen reactie meer geeft, de ander het overneemt. Als de defecte PLC wel reacties geeft, maar niet de juiste, kan bijvoorbeeld gebruik worden gemaakt van een oneven aantal van (ten minste 3) PLC’s, en kan de meerderheidsuitkomst geëffectueerd worden [45]. Hierbij is het wel zo dat dit het systeem complexer maakt, en dat zaken als synchronisatie en race condities tussen de verschillende PLC’s goed bekeken moeten worden.

In dit rapport wordt niet verder in detail ingegaan op het falen van hardware, aangezien bij SBE dezelfde hardware kan worden gebruikt als bij traditioneel ontwikkelen, en de betrouwbaarheid wat dat betreft dus overeenkomt met een traditionele manier van besturingssoftware ontwikkelen.

7.6.4 Wet- en regelgeving rondom veiligheid

CGI heeft voor het beantwoorden van vraag 4 (zie Sectie 6.3.2) specifiek gekeken naar de wet- en regelgeving omtrent veiligheid en gezondheid waar de objecten van Rijkswaterstaat aan moeten voldoen, en de relatie daarvan met SBE [4]. Hier wordt de conclusie kort samengevat.

Objecten van Rijkswaterstaat, zoals beweegbare bruggen, vallen onder de Europese Machinerichtlijn [46] die is opgenomen in de Nederlandse warenwet. De machinerichtlijn stelt een risicobeoordeling verplicht. Voor risico dat niet met mechanische oplossingen kan worden beperkt moeten er besturingstechnische veiligheidsmaatregelen worden genomen. Het ontwerpen van de besturingssoftware met SBE behelst een ander ontwikkelproces dan gebruikelijk is. CGI concludeert dat de machinerichtlijn het gebruik van SBE als methode en Eclipse ESCET als toolkit niet uitsluit. Er worden wel randvoorwaarden gesteld aan het ontwerp en de wijze van risicobeheersing, maar deze zijn niet uniek voor SBE.

Geharmoniseerde normen geven invulling aan het proces om risico's te reduceren tot een acceptabel niveau, zoals de machinerichtlijn vereist. Er zijn meerdere normen, waarbij CGI vooral heeft gekeken naar de IEC 62061 norm [47], welke door Rijkswaterstaat wordt gehanteerd. Zowel het SBE proces, de Eclipse ESCET toolkit, als de gemaakte modellen en code, moeten aan de in deze norm gestelde eisen aan softwareontwikkeling voldoen. De norm beschrijft drie softwareniveaus, waarbij CGI voornamelijk naar softwareniveau 1 heeft gekeken. Dat is het minst complexe softwareniveau, waarbij gebruikt wordt gemaakt van een beperkte instructieset voor veiligheidsgerelateerde programma's. Voor softwareniveau's 2 en 3 is een zwaarder proces nodig, volgens de IEC 61508-3 norm [48], maar dat is in ieder geval voor bruggen en sluizen niet noodzakelijk. Voor softwareniveau 1 is het toegestaan eerder ontwikkelde bouwblokken te gebruiken, mits rekening wordt gehouden met compilervariaties. Ook worden modellen als voorbeeld voor source code gegeven. CGI concludeert dat het mogelijk is met een SBE proces besturingssoftware te genereren voor SIL niveaus 1-3 en dat het proces daarbij voor die drie SIL niveaus gelijk is. Wel dient daarbij aan de noodzakelijke codeerregels en richtlijnen omtrent traceerbaarheid, reviews, testen, verificatie en validatie en configuratiemanagement wordt voldaan, zowel voor de modellen als de gegenereerde code.

CGI geeft hierbij expliciet aan dat met SBE aan al deze extra eisen kan worden voldaan, en dat SBE voor bepaalde eisen zelfs voordelen biedt. Echter, het merkt daarbij wel op dat nu nog niet alles in ingeregeld om daadwerkelijk aan alle eisen te voldoen. CGI komt dan ook met diverse aanbevelingen.

Aanbevelingen vanuit de machinerichtlijn

Alle veiligheidsaanbevelingen voor het werken met SBE, voortkomende uit de veiligheidsstandaarden, zoals opgesteld door CGI, dienen te worden opgevolgd. Na het opvolgen van deze aanbeveling is volgens CGI het toepassen van SBE in de context van veiligheid mogelijk. Deze aanbevelingen zijn hier integraal overgenomen:

V-1: Werk een robuuste implementatie voor het genereren van code voor Safety en Reguliere PLC's uit. Dit omvat een officiële implementatie van de synthese een model (of meerdere modellen) die in codegeneratie in meerdere PLC's kunnen worden geladen. Hiertoe zijn in ieder geval aanpassingen in de taal CIF nodig (bijvoorbeeld het keyword 'SAFE' ter vervanging van de 'S' in commentaar [49], maar mogelijk ook definitie van 'inter PLC communicatie variabelen') die speciale behandeling nodig hebben (zie problematiek uit [49]).

V-2: Neem bij de OBB modellen de Machinerichtlijn in acht met de keuze van sensoren. Alle positiesensoren voor het stoppen van een beweging (eindschakelaars) moeten in de modellen worden geïnverteerd evenals het stopcommando. Bij het wegvallen van dit soort signalen moet namelijk de beweging stoppen.

V-3: Laat de codegenerator een afschatting maken van de maximale programmaduur om te borgen dat de cyclustijd niet overschreden wordt, zodat responstijden van safety-functies geborgd zijn. Immers,

overschrijding van de cyclustijd resulteert bij een safety-PLC altijd in een STOP en daardoor volledige stilstand van het object.

V-4: Vermijd de WHILE constructie in de codegenerator om te borgen dat het programma voldoet aan de safety requirements.

V-5: Modelleer de voor veiligheidsfuncties benodigde diagnostiek en afhandeling in specifieke lagen. Denk hierbij aan een technische laag die uitval van bijv. outputkaart (met als actie reset) afhandelt en een functionele laag die de safety op objectniveau borgt (denk hierbij aan er moet minstens één werkend scheepvaartsein zijn). Besluit hoe de technische laag geïmplementeerd dient te worden per PLC platform.

V-6: Borg dat de codegeneratoren afdoende, traceerbaar commentaar in de code zetten. De ESCET codegenerator produceert commentaar. De nieuwere, onderzochte codegenerator doet dat niet (meer).

V-7: Borg dat de codegeneratoren geen set/reset functies gebruiken, tenminste niet voor de veiligheidsfuncties.

V-8: Neem als expliciete eis op aan de codegeneratoren, dat de outputs alleen aan het eind van het programma geschreven worden.

V-9: Ontwerp een testaanpak die voortbouwt op de garanties van SBE. Datgene dat SBE al garandeert in de PLC code, hoeft niet in detail getest te worden, focus op de stappen waar (menselijke) fouten geïntroduceerd worden.

V-10: Borg in proces en ESCET dat de gebruikte versies van het model, de synthese software, de gebruikte codegenerator meegaan in de code, zodat de PLC omgeving deze meeneemt in de binary die op de PLC gaat draaien.

Algemene aanbevelingen

De overige, algemene aanbevelingen van CGI voor werken met SBE komen voort uit de kennis en ervaring van CGI met betrekking tot systeemontwikkeling. Deze aanbevelingen betreffen geen vereisten vanuit de veiligheids-standaarden. Deze aanbevelingen zijn hier integraal overgenomen:

A-1: Beperk het gebruik van PLC's tot PLC's met een automatiseringskoppeling (bijvoorbeeld API) voor de platformtooling, die het aanmaken van een project toestaat, zodat vanuit een buildserver vanuit de bron, direct het project (code en hardware-instellingen) gegenereerd kan worden zodat fouten in deze stap geëlimineerd worden.

A-2: Ontwikkel een werkwijze om functieblokken mee te kunnen nemen en hun gedrag mee te modelleren, tezamen met een specificatie die leidt tot de juiste configuratie van het functieblok.

A-3: Neem gebruikersparameters op in de werkwijze, hoe ze in de modellen terechtkomen, wat er in simulatie mee gebeurt, hoe het in de code terecht komt, wat de vertrekwaarden zijn. Dit om te voorkomen dat alle parameters hard gecodeerd zijn en dat er tijdens onderhoud onnodig modelaanpassingen nodig zijn.

A-4: Voeg tracering toe in de codegenerator, zodanig dat het voor een auditor duidelijk is welke code (guards) zijn toegevoegd wegens welke requirement.

A-5: Neem het voorkomen van onnodig grote codewijzigingen door kleine functionele wijzigingen mee als principe voor de ontwikkeling van toekomstige codegeneratoren.

A-6: Zorg voor leverancierspecifieke bibliotheken waarin met name de veiligheidsdiagnose en de interfacing met deze diagnosemodules zijn vastgelegd.

A-7: Borg dat de inherente onvoorspelbaarheid van de moderne Safety PLC's in de codegeneratie meegenomen wordt. Hierbij valt te denken aan duidelijke identificatie van de communicatievariabelen van Regulier naar Safety en andersom. En dat iedere executieslag begint met het kopiëren van deze registers.

A-8: Het is het overwegen waard om enkel de functionaliteit van de veiligheidsfuncties te modeleren om zo een aanscherping van de eisen te verkrijgen en het genereren van veiligheidssoftware voorlopig, tot de V-aanbevelingen zijn overgenomen, achterwege te laen wegens te veel menselijke handelingen.

A-9: Vul de SBE ontwerprichtlijnen aan met best practices op V&V gebied. Bijvoorbeeld moeten modelleurs eerst hun werk laten reviewen voordat ze mogen simuleren?

A-10: Werk een methodiek uit van synthese die leidt tot multi-PLC code. Stel hierbij aandachtspunten op voor V&V in dergelijke situatie. Hoe vinden integratietesten plaats? Welke in het totale model aangetoonde eigenschappen zijn ook te vinden in het systeem van samenwerkende PLCs? Waar is additionele V&V voor nodig?

A-11: Het produceren van een veiligheidssysteem vraagt om goede procesbeheersing. SBE als methode heeft een alternatief Software Development Plan nodig, zodat de eisen op procesvlak kunnen worden afgedekt. Bij ontwikkeling en beheer van modellen door Rijkswaterstaat zelf, dient Rijkswaterstaat een dergelijk plan op te stellen en de kwaliteitscontrole uit te voeren.

A-12: Borg in de methodiek van SBE dat een Plant-model voorzien wordt van een ontwerp met overwegingen (waarom is het zoals het is). Bouw in ESCET voorzieningen om dat te borgen (commentaarblokken, die kunnen worden omgezet naar een ontwerp document) of kies een aanvullende omgeving die tracering/koppeling tussen ontwerpbeslissingen en CIF code kan vastleggen.

A-13: Realiseer 'toetsbaarheid' van de synthese in de toolset, zodanig dat de tracering van code naar de requirements mogelijk is.

A-14: Realiseer een integrale gebruikersbeleving in ESCET die een modelleur helpt met overzicht krijgen en houden van een (bestaand) model, waardoor je gebruikersfouten voorkomt.

A-15: Borg het gebruik van de toolset over de levensduur van de modellen, door middel van langere termijn instandhouding en/of een exit strategie. Doordat de modellen in een formele taal zijn, valt bij een exit strategie ook te denken aan een 'upgrade' naar een uitgebreider formalisme, dat ondersteund wordt met uitgebreidere tools.

A-16: Werk in het OBB model met een lijst met 'plant-overwegingen', waarin aannames, keuzes en evt. openstaande punten worden gedocumenteerd. En valideer deze overwegingen voor een aantal werktuigen 'bottom-up' dus vanuit het bestaande I/O vlak naar de controller toe.

A-17: Beoordeel de werkwijze met model-families op eisen tegen de machinerichtlijn op gebied van behandeling van modules, zodra een concrete werkwijze met model-families beschikbaar is.

A-18: Maak een ontwerpstrategie waar fysieke eigenschappen rondom sensoren mee gemodelleerd worden (denk hierbij aan 'ontdenderen van inputs'). Als dat lastig is in CIF, maak dan standaard PLC code die wordt meegenomen in de codegeneratie.

A-19: Laat een onafhankelijke partij (niet de ontwikkelaars) een beoordelingsaanpak voor de ESCET tool chain ontwikkelen, die zodanig transparant en dekkend is dat het resultaat van de beoordeling gebruikt kan worden voor V&V van een individueel project (ofwel de garanties van SBE zijn in de code aanwezig).

Hierbij komen nogal wat aandachtspunten naar boven die nieuw zijn ten opzichte van de samenwerking met de TU/e. CGI heeft ervaring met het maken van machinebesturingen voor infrastructurele objecten, waaronder die van Rijkswaterstaat. Deze expertise is waarschijnlijk ook aanwezig bij andere marktpartijen die voor Rijkswaterstaat besturingssoftware maken. Het aan te raden de aanbevelingen van CGI te

adresseren met een partij die zowel verstand heeft van de machinerichtlijn en andere relevante normen, als van PLC's en PLC-code, en praktische ervaring heeft met het daadwerkelijk realiseren van machinebesturingen. De input van een dergelijke partij is waarschijnlijk ook van grote waarde bij het maken van de nieuwe codegenerator in de Eclipse ESCET toolkit (zie Sectie 7.11.1).

7.7 Efficiëntie

Een van de voornaamste doelen voor Rijkswaterstaat om SBE in te zetten is dat dit de efficiëntie van de ontwikkeling kan verbeteren (zie onder andere Hoofdstuk 6 en Sectie 7.15), voor zowel Rijkswaterstaat zelf als voor marktpartijen.

SBE stelt modellen met een eenduidige betekenis centraal gedurende het gehele ontwikkelproces (zie ook Sectie 5.4), die door personen met verschillende achtergronden begrepen kunnen worden. Hiervoor moeten dan wel modellen gebruikt worden met concepten die aansluiten bij de belevingswereld van alle betrokkenen, en dienen intuïtieve notaties, representaties en visualisaties gebruikt te worden (zie ook Secties 5.2 en 5.4.2). Dit verbetert de communicatie tijdens alle stappen van het ontwikkelproces (zie ook Sectie 7.7). Het voorkomt onduidelijkheden, en daarmee misinterpretaties, fouten en discussies. Zeker als ook exceptionele gevallen worden meegenomen. Hoewel dit simpel en logisch klinkt, moet het effect hiervan niet worden onderschat. Het voorkomen van problemen, die dan vervolgens niet opgelost hoeven te worden, scheelt veel tijd en bespaart daarmee ook kosten. Dit geldt wellicht nog wel meer als er ook wordt samengewerkt met marktpartijen. Dit is een belangrijk voordeel van MBE en daarmee van SBE [50, 51, 52].

De modellen leggen de kennis over de besturing vast, wat dus relevant is voor iedereen die bij de ontwikkeling van de besturing betrokken is, vanwege de genoemde voordelen in de communicatie. Het is echter zeker ook relevant voor nieuwe medewerkers, die nog niet zo goed op de hoogte zijn van bijvoorbeeld het domein of een specifiek object. Vanuit de modellen kunnen zij snel de kennis die in de modellen is vastgelegd tot zich nemen, en daarmee zich sneller de besturing van dat object, en uiteindelijk de kennis van het domein eigen maken. Dit kan de tijd verkorten die nodig is om nieuwe medewerkers in te werken, waardoor ze eerder effectief hun werkzaamheden kunnen doen. Hierbij is het wel belangrijk dat medewerkers voldoende kennis en kunde hebben om de modellen juist te kunnen interpreteren (zie ook Sectie 7.12).

Een ander aspect dat voor efficiëntiewinst zorgt is automatisering. SBE heeft als doel zoveel mogelijk stappen van het ontwikkelproces te automatiseren, door gebruik te maken van onder andere automatische besturingssynthese, automatische formele verificatie, automatische codegeneratie en geautomatiseerd testen door middel van model-gebaseerd testen (zie ook Sectie 7.2). Alles wat is geautomatiseerd kan met 'een druk op de knop' worden uitgevoerd door een computer, en daarmee kan veel handwerk worden voorkomen. Aangezien computers enorm snel complexe taken kunnen uitvoeren, kan dit veel tijd en geld besparen. Zo kan synthese met 'een druk op de knop' een correct-door-constructie besturingsmodel genereren, zonder dat het de ontwikkelaar tijd kost. En codegeneratie kan met een druk op correct-door-constructie code genereren, zonder dat een PLC-programmeur daar veel tijd aan kwijt is. Duurt een dergelijke geautomatiseerde stap wel langer, dan hoeft niet op het resultaat gewacht te worden, en kan de betreffende persoon in de tussentijd andere werkzaamheden uitvoeren. SBE stelt ontwikkelaars daarmee gedurende het gehele ontwikkelproces in staat zich te concentreren op *wat* de besturing moet doen in plaats van *hoe* die dat moet doen, wat veel tijd scheelt, en zorgt dat de tijd die wel wordt gespendeerd efficiënt kan worden besteed aan datgene waar een computer niet goed in is, namelijk besluiten wat gewenst is voor een besturing.

De vergaande automatisering heeft echter ook een keerzijde. Fouten in de plant en requirement modellen werken direct door in de gesynthetiseerde supervisor en de gegenereerde PLC-code. Hierbij blijven eigenlijk alleen simulatie en testen over om deze problemen te vinden. Bij traditioneel ontwikkelen worden geen stappen geautomatiseerd, en bij MBE en VBE worden minder stappen geautomatiseerd. Er moet dus meer werk verzet worden om bijvoorbeeld handmatig een ontwerp of PLC-code te maken. Dat kost meer

moeite. Maar, het kan ook door verschillende personen worden gedaan. En daarbij betekent meer mensenwerk potentieel dus ook meer kans dat die verschillende personen problemen vinden. Hiermee kan er dus in bepaalde mate sprake zijn van een trade-off tussen efficiëntie en kwaliteit (zie ook Sectie 7.6).

Het gebruik van formele modellen waar de computer aan kan rekenen, in combinatie met een ver geautomatiseerd proces met synthese, maakt ook *rapid prototyping* mogelijk (zie ook Sectie 7.2.5). Na het specificeren van plants en requirements is het met SBE mogelijk om daar automatisch een besturingsmodel uit te genereren en dat vervolgens te simuleren. In combinatie met intuïtieve visualisaties (in 2D dan wel 3D) kan dit snel inzicht geven in het gespecificeerde gedrag. Door het betrekken van diverse partijen, van domein experts tot de personen die het object uiteindelijk moeten bedienen, kan al in een vroeg stadium gekeken worden of de specificatie en het ontwerp voor de gewenste besturing zorgen. Is dat niet het geval, dan kunnen de plants en requirement modellen worden aangepast, en kan met ‘een druk op de knop’ weer een nieuw besturingsmodel worden gesynthetiseerd en gesimuleerd. Deze korte iteraties zijn zeer efficiënt en besparen tijd. Het voorkomt namelijk in veel gevallen dat pas later onvolledigheden, inconsistenties en andere onjuistheden naar boven komen, waar ze lastiger en tegen hogere kosten moeten worden opgelost (zie ook Sectie 5.4.5). Vroege validatie kan, als het juist wordt ingezet, aanzienlijk veel tijd en geld besparen. Het is hiervoor dan wel essentieel dat de gebruikte tools (zie Sectie 7.11) dit ondersteunen, en dat ze schaalbaar genoeg zijn om ook voor complexe modellen die snelle iteraties mogelijk te maken (zie ook Sectie 7.10).

De formele methoden die bij SBE worden gebruikt hebben daarnaast het voordeel dat ze bepaalde garanties geven (zie ook Sectie 7.5). Zo geeft besturingssynthese de garantie dat het gesynthetiseerde besturingsmodel aan alle gestelde besturingseisen voldoet. Door dit soort garanties hoeft er minder te worden getest, wat tijd bespaart.

Een model-gebaseerde aanpak kan ook helpen bij het vroegtijdig trainen van personen die de te besturen objecten zullen bedienen. Ze kunnen namelijk met een digital twin trainen, in plaats van met het echte object, en hiermee al eerder tijdens de ontwikkeling van het object bekend raken met de bediening ervan (zie ook Sectie 7.2.5). Door dit in een vroeg stadium te doen, kan het object zodra het is opgeleverd direct door deze personen worden bestuurd, en kan het object dus sneller in gebruik worden genomen.

Daarnaast kan SBE tijd besparen bij onderhoud en evolutie, door de automatisering en computerondersteuning die het biedt (zie ook Sectie 7.3). In zekere zin is onderhoud en evolutie wat dat betreft niet veel anders dan een extra iteratie in het *rapid prototyping* proces. Dezelfde efficiëntiewinst die te bepalen valt tijdens de eerste ontwikkeling, zou dan ook te behalen moeten zijn voor onderhoud en bij het evolueren van de besturing, zoals bij het toevoegen van extra functionaliteit.

Verder maakt SBE het mogelijk efficiënt met productfamilies om te gaan (zie ook Sectie 7.4). Door hergebruik en standaardisatie kunnen eerder gemaakte fouten bij volgende objecten worden voorkomen. Daarnaast is veel werk al gedaan en hoeft het dus voor een volgend object het wiel niet opnieuw te worden uitgevonden [37].

Samenvattend kan SBE op tal van manieren de efficiëntie verhogen, door met eenduidige modellen communicatie te verbeteren en misinterpretaties, fouten en discussies te voorkomen, door kennis vast te leggen in modellen en daarmee nieuwe werknemers eerder effectief te laten zijn, door tal van zaken te automatiseren, door met korte iteraties problemen sneller en eerder te vinden, door bepaalde formele garanties te geven die bijvoorbeeld testtijd besparen, door eerder training mogelijk te maken, door met hergebruik en standaardisatie eerder gemaakte fouten te voorkomen en werk uit te sparen, en door dit niet alleen bij de eerste ontwikkeling maar ook bij onderhoud en evolutie mogelijk te maken.

Dit zijn niet alleen theoretische voordelen. Veel van deze voordelen zijn bijvoorbeeld aangetoond in de samenwerking met de TU/e (zie onder andere Sectie 7.9). Studenten van de TU/e blijken in staat te zijn om, met slechts beperkte kennis van de systemen van Rijkswaterstaat, in korte tijd een werkende besturing te maken (zie ook Sectie 7.13).

Maar ook in de literatuur zijn dit soort efficiëntievoordelen bekend. Zo zorgde de introductie van model-gebaseerd ontwikkelen bij Motorola [44] initieel al voor een inspanningsreductie van ongeveer 2,3 keer, onder andere door het gebruik van simulatie, automatische codegeneratie en model-gebaseerd testen. Daarnaast heeft bijvoorbeeld de introductie van test generatie voor een reductie van ongeveer 33% gezorgd. Hoewel niet overal (evenveel) voordeel wordt behaald, zijn de voordelen voor bepaalde stappen enorm. Voor het correct oplossen van fouten tijdens systeemintegratie is het bij Motorola niet ongebruikelijk om tijdsreducties te zien van zelfs 30 tot 70 keer. Dit wordt toegeschreven aan de mogelijkheden om veel op modelniveau in plaats van op implementatieniveau te doen, en automatisch nieuwe code te kunnen genereren. In het algemeen zijn bij Motorola in de latere jaren van MBE adoptie typische productiviteitsverhogingen waarneembaar geweest van zo'n 2x tot 8x, werden fouten 3 keer vaker al gevonden tijdens de ontwikkelstap waar ze zijn gemaakt, werden fouten vaker in eerdere ontwikkelstappen al gevonden, en waren de algehele kosten om kwaliteit te garanderen lager dan voorheen.

Concluderend kan worden gesteld dat met SBE niet alleen theoretisch, maar ook in de praktijk, potentieel veel efficiëntiewinst is te behalen. Hiervoor moet echter een aantal andere aspecten niet uit het oog worden verloren. Zo moeten dan onder andere wel de juiste tools worden ingezet (zie Sectie 7.11), moet de juiste kennis en kunde aanwezig zijn (zie Sectie 7.12), moet een volwassen genoeg ecosysteem aanwezig zijn (zie Sectie 7.14), en moet het veranderen van werkwijze niet worden onderschat (zie Sectie 7.13). Verder dienen de aanbevelingen van CGI omtrent automatisering geadresseerd te worden (zie Sectie 7.6.4).

Verder zullen de potentiële voordelen qua efficiëntie niet allemaal vanaf de eerste dag te behalen zijn. Bij de introductie van SBE zal tal van zaken moeten worden ingeregeld en opgezet. Daardoor kan de inzet van SBE in die beginfase meer tijd en geld kosten. De volle potentie van SBE zal daarom pas na die initiële investering tot uitdrukking komen.

7.8 Afhankelijkheid van leveranciers

Rijkswaterstaat is momenteel voor het ontwikkelen van besturingen afhankelijk van tal van leveranciers. In deze sectie wordt kort ingegaan op de invloed van SBE op leveranciersafhankelijkheid voor het ontwikkelen van besturingssoftware, waarbij de focus ligt op twee soorten leveranciers: 1) leveranciers van PLC's en bijbehorende PLC-platformen (met onder andere een PLC-ontwikkelomgeving), en 2) marktpartijen die voor Rijkswaterstaat een oplossing leveren door bijvoorbeeld besturingssoftware te ontwikkelen. Een uitgebreidere beschouwing van leveranciersafhankelijkheid is te vinden in een eerder TNO rapport dat voor Rijkswaterstaat is geschreven [53].

7.8.1 PLC-leveranciers

Leveranciers van PLC's en PLC-platformen blijven bij een SBE aanpak nodig, als gekozen blijft worden voor het gebruik van PLC's en PLC-software (zie ook Sectie 7.2.3). De modellen van besturingslogica zoals deze bij SBE worden gebruikt zijn echter wel platform- en leveranciersonafhankelijk. De duidelijke scheiding tussen ontwerp en realisatie laat de keuze voor platform en leverancier vrij. Er kan dus nog altijd gekozen worden tussen diverse leveranciers van PLC's met bijbehorende platformen, door een andere PLC-codegenerator te gebruiken. Maar het is ook mogelijk voor industriële PC's als platform code te genereren, bijvoorbeeld voor een programmeertaal als C of Java. Dit dient dan wel verder uitgewerkt te worden. Verder is het eventueel wisselen van platform of leverancier in een later stadium wellicht ook eenvoudiger, omdat ook dan de code niet handmatig hoeft te worden aangepast, maar simpelweg opnieuw kan worden gegenereerd.

SBE geeft hier gunstige vooruitzichten. Hiervoor moet de gewenste codegeneratie dan wel worden ondersteund door de gebruikte SBE-tooling, en juist zijn geïmplementeerd (zie ook Sectie 7.11). Daarbij moet ook gelet worden op diverse andere relevante zaken rondom codegeneratie zoals besproken in Sectie 7.2.3, waaronder ook dat er gediend te worden gekeken naar de PLC-specifieke (veiligheids)functieblokken van de diverse PLC-leveranciers (zie ook Sectie 7.6.4) en hoe daar mee om te gaan in bijvoorbeeld modellen van productfamilies (zie ook Sectie 7.4). Verder kan het verstandig zijn al in een vroeg stadium

ervaring op te doen met PLC's van meerdere fabrikanten, en het migreren naar PLC's van een andere fabrikant. Daarnaast dienen de aanbevelingen van CGI omtrent PLC's meegenomen te worden (zie Sectie 7.6.4).

7.8.2 Samenwerking met marktpartijen

Wat betreft marktpartijen die voor Rijkswaterstaat een oplossing leveren door bijvoorbeeld besturingssoftware te ontwikkelen, biedt SBE ook voordelen. Met meer eenduidige, complete, consistente, uniforme en up-to-date formele specificaties staat Rijkswaterstaat sterker tegenover leveranciers, heeft het meer grip op die samenwerking, en kan het beter de regie voeren. De betere specificaties kunnen miscommunicatie en discussies voorkomen (zie ook Sectie 7.7), waarmee in het algemeen de kans groter is dat daadwerkelijk op tijd de gewenste besturing wordt opgeleverd.

Daarnaast kunnen model-gebaseerde formele specificaties gebruikt worden om de opgeleverde realisatie te toetsen tegen de specificatie, bijvoorbeeld als onderdeel van een acceptatietest. Dit zou onder andere gedaan kunnen worden met model-gebaseerd testen. Het maakt daarbij in principe niet uit of de realisatie is gemaakt door middel van handmatig programmeren of via codegeneratie. Rijkswaterstaat kan het toetsen zelf doen, als eis stellen aan de leverancier, of laten uitvoeren door een onafhankelijke derde partij. Rijkswaterstaat heeft hiermee meer grip op wat het geleverd krijgt door leveranciers.

Ook kan de (gedeeltelijke) automatisering van de realisatie van besturingssoftware door middel van bijvoorbeeld codegeneratie (zie ook Sectie 7.2.3) leveranciers hiervoor (gedeeltelijk) overbodig maken. Dit heeft wel als keerzijde dat Rijkswaterstaat hiermee zelf meer verantwoordelijk krijgt. Wordt er wel voor een leverancier gekozen, dan is te verwachten dat de realisatie mogelijk efficiënter en dus tegen lagere kosten uitgevoerd kan worden (zie ook Sectie 7.7), ten minste als het object significant anders is dan andere objecten en de leverancier daar dus niet al een (gedeeltelijke) implementatie voor heeft klaarliggen. Hoe dan ook is het zaak de verantwoordelijkheden van zowel Rijkswaterstaat als marktpartijen goed vast te leggen.

Verder renoveert Rijkswaterstaat de komende jaren een groot aantal objecten. Een herbruikbare bibliotheek van modellen waaruit een specificatie voor een specifiek object kan worden samengesteld (zie ook Sectie 7.4), zou kunnen worden ingezet voor meer uniformiteit en efficiëntie (zie ook Sectie 7.7). Dit kan Rijkswaterstaat meer controle geven, als het zelf deze modellen beheert, maar de verdere realisatie (per object) overlaat aan een marktpartij. Hierbij moet er wel voor worden gewaakt dat de in de markt aanwezig kennis niet verloren gaat en voldoende wordt benut, zoals is geadviseerd in het BIT-advies [1].

Samenvattend kan geconcludeerd worden dat SBE kan leiden tot meer controle en eigenaarschap ten opzichte van marktpartijen, en eventueel zelfs tot gedeeltelijke leveranciersonafhankelijk (wat betreft de besturingssoftware). Dit heeft wel als schaduwzijde dat Rijkswaterstaat dan ook die controle en dat eigenaarschap moet pakken, dat het goed moet kijken hoe het dat wil inrichten, moet bepalen wie dan waar voor verantwoordelijk is, en wie dat gaat regisseren en begeleiden. De juiste balans tussen wat Rijkswaterstaat zelf doet, wat marktpartijen doen, en waar samengewerkt wordt, is hierbij essentieel. Rijkswaterstaat heeft hier ideeën over (zie Hoofdstuk 6). Dit dient echter nog verder uitgewerkt te worden, en er moet nog meer ervaring op worden opgedaan in de praktijk. Een verandering van werkwijze is namelijk niet altijd even eenvoudig te bewerkstelligen (zie ook Sectie 7.13), zeker niet als er een ecosysteem van partijen bij komt kijken (zie ook Sectie 7.14).

7.9 Geschiktheid voor infrastructureel domein

Om te bekijken in hoeverre SBE specifiek geschikt is voor de toepassing op infrastructurele systemen, is het ten eerste goed om te kijken naar gedocumenteerde toepassingen van SBE voor dit domein.

In de samenwerking tussen Rijkswaterstaat en TU/e zijn diverse stappen van SBE toegepast op tal van objecten van Rijkswaterstaat. Gezien het grote aantal van dergelijke toepassingen, wordt er hier een aantal

genoemd, zonder uitputtend te zijn. Voor een model sluis gebaseerd op de sluis bij Terneuzen [54], en onder meer Sluis III in Tilburg [55], zijn supervisors gesynthetiseerd uit gespecificeerde plant en requirements modellen. Verder is er voor Sluis III in Tilburg een groter gedeelte van het ontwikkeltraject doorlopen [56], met onder andere de specificatie van plants en requirements, synthese, validatie middels simulatie en visualisatie, implementatie middels codegeneratie voor PLC's volgens de IEC 61131-3 standaard, en validatie van de PLC-code tegen een experimentele installatie in plaats van de daadwerkelijke sluis, omdat deze nog in ontwikkeling was. Hierbij heeft men genoegen genomen met de garanties van synthese en is er geen extra verificatie uitgevoerd. De eisen in de Landelijke Bruggen- en Sluizenstandaard (LBS) bleken, het faalgedrag buiten beschouwing latend, op een goed niveau voor gebruik bij supervisor synthese. Rijkswaterstaat is van plan extra eisen toe te voegen voor faalgedrag en diagnose daarvan.

Voor de beweegbare Algerabrug in Krimpen aan de IJssel is niet alleen het nominale (*happy flow*) gedrag meegenomen in het ontwikkelproces, maar ook het faalgedrag, het gedrag nadat een fout optreedt, zoals gebroken kabels en defecte sensoren [57]. De gesynthetiseerde *fault-tolerant* supervisor blijft juist functioneren bij het falen van hardware, ook als deze niet redundant is uitgevoerd, en houdt daarmee de brug beschikbaar voor gebruik. Het beperkt bijvoorbeeld het gedrag als een slagboom vast blijft zitten in de gesloten positie om veilig functioneren te blijven garanderen, en neemt de juiste actie als het brugdek onverwacht op het foute moment opent. Ook voor de Oisterwijksebaanbrug is er een supervisor gesynthetiseerd [25] die om kan gaan met gedetecteerde fouten en de brug in die situaties toch juist kan blijven besturen [26].

Daarnaast is er bij de Oisterwijksebaanbrug ook gekeken [27] naar de aspecten die komen kijken bij het omzetten van een gesynthetiseerde supervisor naar een safety controller, zoals het scheiden van de veiligheidseisen van de (overige) besturingseisen en het gedistribueerd implementeren ervan in safety PLC's. De gegenereerde PLC-code voor deze besturing is ook daadwerkelijk getest, en heeft de standaard factory acceptance tests doorstaan, zowel in een hardware-in-the-loop context in combinatie met een SCADA-applicatie als op de daadwerkelijke brug. Ook CGI heeft naar de toepassing van SBE op Oisterwijksebaan gekeken, met specifiek ook aandacht voor het PLC-codegeneratie gedeelte van het ontwikkelproces. Daar zijn diverse aanbevelingen uitgekomen (zie ook Sectie 7.2.3).

Het gebruik en hergebruik van standaard component- en eismodellen voor specificatie en supervisor synthese is gedemonstreerd via diverse use cases, waaronder Sluis III [56], Sluis Empel, Sluis Hintham, Algerabrug [57], Algeracomplex (sluis-brug combinatie) [15], Prinses Marijkecomplex (twee sluizen en keerschuif) [58], Oisterwijksebaanbrug [25], diverse tunnels waaronder de Koning Willem-Alexandertunnel en de Eerste Heinenoordtunnel [36], en voor wegkantsystemen [35].

Recent werk laat zien dat supervisor synthese ook de potentie heeft om ingezet te worden voor het besturen van waterkeringen. Concreet is gekeken naar het toepassen van synthese voor de Maeslantkering, een stormvloedkering [59, 60].

De TU/e heeft een terugblik gepubliceerd rondom recente ontwikkelingen in samenwerking met Rijkswaterstaat [5], waarin ook vooruit wordt gekeken naar de toekomst. De conclusie van de TU/e daarin is (vertaald vanuit het Engels): "De casussen zoals gepresenteerd in dit artikel, in de context van een project met Rijkswaterstaat, laten zien dat de theorie een niveau van volwassenheid heeft bereikt dat het toestaat om volledig te worden omarmd door de industrie." In een ander artikel [61] trekt de TU/e een soortgelijke conclusie, namelijk (vertaald uit het Engels): "Supervisory control synthesis is op een niveau gekomen waar het succesvol kan worden toegepast voor het ontwikkelen van echte systemen."

Ten tweede is het goed om te kijken naar toepassingen van SBE in andere domeinen, en in hoeverre deze verschillen van het infrastructureel domein. De TU/e heeft het synthetiseren van besturingen ook toegepast bij andere bedrijven, zoals voor het besturen van pretpark voertuigen [62, 63], MRI scanners van Philips [64] en lithografie machines van ASML [65]. De TU/e is niet de enige partij die synthese toepast. Zo is het bijvoorbeeld toegepast voor robots [66]. Voor een uitgebreider overzicht, zie [42]. Deze toepassingen

verschillen niet wezenlijk van de toepassingen op infrastructurele systemen van Rijkswaterstaat, in de zin dat ze allen passen binnen in de supervisory control architectuur van Figuur 7 in Sectie 5.6.2.

De diverse ervaringen met het toepassing van SBE op objecten van Rijkswaterstaat, die samen alle stappen van het SBE ontwikkelproces beslaan, geven geen aanleiding om aan te nemen dat SBE niet toepasbaar zou zijn voor het domein van infrastructurele systemen. Daarnaast is SBE toegepast in tal van andere domeinen, en verschillen infrastructurele systemen niet wezenlijk van dergelijke systemen. Hieruit kan geconcludeerd worden dat, de overige elders besproken aspecten buiten beschouwing houdend, SBE in de basis geschikt is voor toepassing op infrastructurele systemen.

7.10 Schaalbaarheid van formele technieken

Een van de bekende uitdagingen bij het gebruik van formele methoden zoals formele verificatie en supervisor synthese is de schaalbaarheid van de gebruikte formele technieken. Het is daarom belangrijk om inzicht te hebben in hoeverre de schaalbaarheid van de formele technieken die ten grondslag liggen aan SBE, waaronder bijvoorbeeld algoritmes voor besturingssynthese, voldoende is voor toepassing op de infrastructurele objecten van Rijkswaterstaat.

Verificatie kon binnen enkele seconden worden uitgevoerd voor de Algerabrug [28]. Ook de eisen uit de standaard voor beweegbare bruggen zijn succesvol geverifieerd [29], alhoewel hier niet alle aspecten zijn meegenomen. Het gebruiken van de juiste state-of-the-art verificatietechnieken voor de juiste situatie is echter wel essentieel. Verificatie is voor objecten van Rijkswaterstaat echter nog slechts vrij beperkt toegepast, en daarmee is nog onvoldoende duidelijk hoe schaalbaar verificatie voor dit soort objecten is. Gedeeltelijk is het ook overbodig als synthese wordt toegepast, aangezien bepaalde eigenschappen dan al door constructie gelden.

Wat betreft synthese, wordt bij de zogenaamde monolithische synthese een enkele supervisor gesynthetiseerd. Dit duurt vaak erg lang voor complexe systemen en hierbij kan ook meer geheugenruimte nodig zijn dan wat beschikbaar is. Daarom is het ook bij synthese essentieel dat gebruik wordt gemaakt van de juiste state-of-the-art technieken, zoals symbolische synthese [67]. Voor Sluis III in Tilburg, gebaseerd op 234 verschillende besturingseisen, kon echter binnen enkele seconden een supervisor gesynthetiseerd worden [56], gebruik makend van state-of-the-art monolithische synthese. Hetzelfde geldt voor 17 beweegbare bruggen [37]. Tunnels zijn over het algemeen een ordegrootte complexer dan sluizen en bruggen. Met behulp van een tunnelconfigurator zijn voor een familie van 22 tunnels supervisors gesynthetiseerd, en voor elke tunnel was de monolithische synthese “een kwestie van minuten” [36]. Voor het Algeracomplex, een sluis-brug combinatie, kon monolithische synthese worden voltooid voor de sluis en brug in respectievelijk 2 en 114 seconden [17]. Voor de combinatie samen duurde het bijna 35 minuten. Hierbij zijn geen fault-tolerant supervisors gesynthetiseerd, wat er voor zou kunnen zorgen dat synthese dan nog langer duurt. Voor de Oisterwijksebaanbrug is een fault-tolerant supervisor gesynthetiseerd voor 82 plantcomponenten, 40 diagnosers en 167 requirements in “een paar minuten” [17].

Waar monolithische synthese tegen grenzen aanloopt kan verdere state-of-the-art voor besturingssynthese worden gebruikt. Zo is multi-level supervisory control synthesis voor diverse objecten gebruikt [54, 55, 16]. In tegenstelling tot monolithische synthese is multi-level synthese een vorm van modulaire synthese. Het maakt gebruik van het feit dat plants (individuele sensoren en actuatoren) en requirements (besturingseisen) modulair gespecificeerd kunnen worden. Multi-level synthese groepeert automatisch de plants en requirements in een boomstructuur van meerdere niveaus. Er is geen fundamentele begrenzing op het aantal niveaus. Voor het Algeracomplex kan bijvoorbeeld gedacht worden aan het scheiden van de brug en sluis. Maar ook een sluis zelf kan (verder) worden opgedeeld in deelsystemen, bijvoorbeeld door het scheiden van besturingen van de noodstop, verkeerslichten, de sluisdeuren en het waterniveau [16]. En te verwachten valt dat ook diagnosers voor fault-tolerant supervisors gedeeltelijk apart gesynthetiseerd kunnen worden. Hierbij wordt dus gebruik gemaakt van in het systeem aanwezige structuur, om voor ieder deelsysteem (zoals gegroepeerd in de boomstructuur) een deelsupervisor te synthetiseren, die samen het gehele systeem besturen. Voor elk deelsysteem wordt monolithische synthese toegepast. Gezien dat de

deelsystemen dan een stuk kleiner en minder complex zijn dan het gehele systeem, is de monolithische synthese voor een dergelijk deelsysteem doorgaans aanzienlijk sneller dan voor het gehele systeem, aangezien de state spaces waaraan wordt gerekend aanzienlijk kleiner zijn [55, 16]. Echter, er moet nog wel een globale non-blockingness check uitgevoerd worden op de resulterende set deelsupervisors, om te zorgen dat het gehele systeem ook non-blocking is. Dit kan nog altijd rekenintensief zijn. Maar, hierbij kan mogelijk wel met abstracties van de deelsystemen worden gewerkt, om de benodigde rekenkracht te beperken. De potentiële efficiëntiewinst maakt dat voor diverse grotere en complexere systemen een supervisor kan worden gesynthetiseerd, waar dat met monolithische synthese niet mogelijk is. Multi-level synthese op dit moment nog niet toegepast op het Algeracomplex.

Als bepaalde eigenschappen gelden is synthese in het geheel niet nodig, omdat de plant en requirement modellen dan samen al een correct-door-constructie supervisor vormen [16, 68] (zie ook Sectie 7.15). Bij multi-level synthese kan het bijvoorbeeld zijn dat voor bepaalde groep plants en requirements geen deelsupervisor hoeft te worden gesynthetiseerd, en voor andere groepen wel. Door geen synthese te hoeven doen kan tijd worden bespaard. Uiteraard moeten de eigenschappen wel gecontroleerd worden. Dit kan echter per plant en requirement apart gebeuren, waardoor dit met beperkte moeite kan worden bepaald. Bij het gebruik van een bibliotheek met herbruikbare bouwblokken zijn bepaalde eisen al gecontroleerd en hoeven deze niet voor elk object nogmaals apart te worden bekeken (zie ook Sectie 7.4).

Hoewel voor het Algeracomplex de monolithische synthese voor de combinatie van brug en sluis meer dan een half uur duurt, kan synthese voor de brug en sluis apart binnen een paar minuten worden voltooid. Hiermee is het voor deelsystemen dus nog altijd mogelijk om snel en efficiënt te synthetiseren. Gezien dat er slechts enkele requirements zijn die relateren aan de combinatie van de brug en sluis, zeker in verhouding met het aantal requirements voor de twee objecten zelf, is het minder relevant dat het synthetiseren van een supervisor voor de combinatie wat langer duurt. Er zal voor de combinatie waarschijnlijk minder iteraties nodig zijn om tot een correct ontwerp te komen. Daarnaast worden in de praktijk de brug en sluis doorgaans ieder van een aparte besturing voorzien, waarmee dit in de praktijk wellicht niet zozeer een probleem is. En dan is enkel nog gebruik gemaakt van monolithische synthese. Bij gebruik van multi-level synthese, en gezien de beperkte koppeling met requirements tussen de brug en de sluis, ligt het in de verwachting dat deze twee deelsystemen daarbij apart gesynthetiseerd kunnen worden. Dit zou aanzienlijk efficiënter moeten zijn.

Het is nog wel verstandig om multi-level synthese ook toe te passen op het Algeracomplex, en daarbij een fault-tolerant supervisor te synthetiseren. Hiermee kan worden bevestigd dat ook voor dit complex, met een juiste combinatie van technieken, de schaalbaarheidsproblemen zijn opgelost en synthese in korte tijd kan worden uitgevoerd, om snelle en efficiënte ontwerpcyclussen mogelijk te maken (zie ook Sectie 7.7).

Daarmee is het momenteel dus nog niet mogelijk definitief vast te stellen dat synthese schaalbaar genoeg is voor toepassing op de infrastructurele objecten van Rijkswaterstaat. Dit is echter wel aannemelijk, gezien de beschreven ervaringen en de positieve vooruitzichten wat betreft multi-level synthese voor het Algeracomplex. Echter, het gebruik van de juiste state-of-the-art technieken voor de juiste situatie is dan wel vereist.

Daarnaast moeten de schaalbare algoritmen ook efficiënt worden geïmplementeerd in tooling, om te zorgen dat de schaalbaarheid in de toolondersteuning niet verloren gaat (zie ook Sectie 7.11). En verder kan de manier van modelleren grote invloed hebben op de synthesecomplexiteit en daarmee de synthesetijd, zo is gebleken uit de samenwerking tussen Rijkswaterstaat en de TU/e [16, 69, 70]. Daarbij zijn wat eerste regels en richtlijnen uitgekomen voor het maken van (herbruikbare) modellen die efficiënt synthetiseerbaar zijn. Rijkswaterstaat zou er goed aan doen om, zelf of samen met een kennispartner, deze regels en richtlijnen verder uit te breiden.

7.11 Tooling

Bij SBE staan automatisering en computer ondersteuning centraal. Dit vereist tooling, computer programma's die het bijvoorbeeld mogelijk maken om te modelleren, synthetiseren, simuleren en code te genereren. Rijkswaterstaat heeft hierbij gekozen voor de Eclipse ESCET toolkit (zie Sectie 6.2.7).

Wat betreft SBE ten opzichte van een traditionele manier van softwareontwikkeling, is de manier waarop de PLC-code tot stand komt anders. Bij SBE wordt hiervoor gemodelleerd, gesynthetiseerd en code gegenereerd. Bij traditioneel ontwikkelen wordt de software handmatig gemaakt. Zodra de PLC-code er is, wordt echter in beide gevallen een PLC-omgeving van de betreffende PLC-fabrikant gebruikt om de PLC-code te compileren, en op de PLC te zetten.

Voor het aspect wordt in deze sectie vooral de Eclipse ESCET toolkit besproken, omdat hier het verschil zit ten opzichte van de huidige manier van werken. Waar relevant wordt echter ook de PLC-omgeving van de PLC-fabrikant bij de discussie betrokken.

7.11.1 Eclipse ESCET en CIF

De *Eclipse Supervisory Control Engineering Toolkit* (ESCET) [24] is een toolkit voor de ontwikkeling van supervisory controllers volgens een model-gebaseerde aanpak. De toolkit heeft naast een focus op model-gebaseerd ontwikkelen, ook een focus op supervisory controller synthesis en industriële toepasbaarheid. Voor dit rapport is versie 0.7 bekeken, de op het moment van schrijven meest recente versie van de toolkit.

Eclipse ESCET biedt meerdere modelleertalen, waarin modellen kunnen worden opgesteld, en programma's om met die modellen te werken. Eén daarvan is de CIF taal met bijbehorende tools [71], die specifiek worden ontwikkeld om SBE te ondersteunen. CIF is een op toestandsmachines gebaseerde taal, uitgebreid met diverse concepten zoals data en tijd, en concepten voor besturingssynthese en het maken van modellen voor grotere, complexere en geparametriseerde systemen. De bijbehorende toolset heeft als doel het gehele ontwikkelproces van besturingen te ondersteunen, inclusief specificatie, synthese, validatie middels simulatie en visualisatie, verificatie, real-time testen en codegeneratie. Uit de samenwerking tussen Rijkswaterstaat en de TU/e is gebleken dat CIF gebruikt kan worden voor het ontwikkelen van besturingssoftware voor Rijkswaterstaat objecten (zie Sectie 7.9).

Voor specificatie heeft CIF diverse concepten die het mogelijk maken om intuïtieve (begrijpbare en leesbare) plant en requirement modellen te maken, zoals diverse vormen om requirements te formuleren en het rechtstreeks kunnen verwijzen naar locaties van automaten. Ook heeft CIF de mogelijkheid (delen van) plants en requirements apart te modelleren, en ieder een eigen naam te geven. Dit helpt in traceerbaarheid van bijvoorbeeld de requirements. Verder biedt CIF hergebruik van (deel)modellen, onder andere door het eenmalig definiëren van componenten en die vervolgens meerdere keren te instantiëren, maar ook door middel van het 'import' mechanisme. CIF heeft ook concepten die het mogelijk maken simulatie modellen te maken, zoals de mogelijkheid om modellen met tijd te maken. Hierbij kunnen alle modellen dus in een enkele taal worden gemodelleerd, en hoeft er niet steeds een andere taal gebruikt te worden. Dit heeft als voordeel dat er slechts een enkele taal geleerd hoeft te worden, en dat hetzelfde gedrag niet in meerdere talen hoeft te worden beschreven. Daarnaast detecteert de CIF tooling automatisch diverse modelleerfouten en rapporteert deze aan de gebruiker. Zo geeft bijvoorbeeld de type checker syntax- en typeringsfouten aan, en detecteert synthese events die in de plant niet geactiveerd kunnen worden, nog zonder dat de requirements in beschouwing worden genomen.

CIF biedt supervisory controller synthesis, waaronder een data-based synthesis tool die gebruik maakt van state-of-the-art symbolische synthese algoritmes, voor het synthetiseren van besturingsmodellen voor complexere systemen. CIF biedt momenteel echter alleen monolithische synthese. Andere vormen van synthese, zoals modulaire en multi-level synthese, zijn nog niet in de toolset beschikbaar. Multi-level synthese is wel reeds als prototype geïmplementeerd en is uitgebreid toegepast door Rijkswaterstaat en

TU/e [16] (zie ook Sectie 7.10). De TU/e is momenteel bezig dit prototype verder door te ontwikkelen en toe te voegen aan de open source Eclipse ESCET toolkit [72].

De CIF toolset bevat diverse tools gerelateerd aan codegeneratie. Zo bevat het een ‘controller property checker’ tool, die diverse eigenschappen kan verifiëren op het gesynthetiseerde besturingsmodel (zie Sectie 7.2.3), die moeten gelden om correcte (PLC-)code te kunnen genereren. Momenteel kunnen de *confluence* en *finite* response eigenschappen worden geverifieerd, maar ontbreekt het verifiëren van de *nonblocking under control* nog. Als de eigenschappen gelden, kan code worden gegenereerd. CIF heeft hiervoor meerdere codegeneratoren, onder andere om Java, C en PLC-code te genereren. De huidige PLC-codegenerator is al jaren veelvuldig door studenten gebruikt, zowel bij vakken als bij toepassing in de industrie in afstudeerprojecten [73]. Verder is deze ook in Rijkswaterstaat context ingezet [56, 25]. Momenteel kan PLC-code worden gegenereerd voor PLCopen XML, IEC 61131-3, TwinCAT en Siemens S7 (1500, 1200, 400 en 300). Die zijn echter niet allemaal even lang aanwezig, en ook niet evenveel toegepast. De codegenerator is echter uitbreidbaar en codegeneratie voor PLC’s van andere fabrikanten kan dus worden toegevoegd. De TU/e werkt momenteel aan een volgende generatie codegenerator voor PLC-code [17, 74], met als voornaamste doel dat er traceerbaardere PLC-code uitkomt. Het idee hierbij is om de structuur van het besturingsmodel (inclusief de plants) waar mogelijk over te nemen in de PLC-code. Hiermee is de PLC-code eenvoudiger te relateren aan het besturingsmodel, wat dus de traceerbaarheid verhoogt. Deze zal ook codegeneratie voor ABB PLC’s ondersteunen. Ongeacht welke PLC-codegenerator gebruikt wordt, dient de gegenereerde PLC-code in de PLC-omgeving van de desbetreffende fabrikant te worden geladen, te worden gecompileerd, en op de PLC te worden gezet. Het aanbrengen van de scheiding in safety en reguliere PLC-code (zie ook Secties 7.2.3 en 7.9) is nog niet geïmplementeerd in de CIF PLC-codegenerator in het ESCET project. Ook het inbrengen van de gegenereerde code in de PLC-omgeving van de fabrikant vereist nu nog wat handmatig werk.

Verificatie is bij het gebruik van synthese niet voor alle eigenschappen nodig, omdat diverse eigenschappen al door constructie worden gegarandeerd (zie Sectie 7.2.4). Voor de overige eigenschappen biedt CIF automatische transformaties aan, die het mogelijk maken het CIF model om te zetten in een ander soort model, dat wordt begrepen door verificatie tools. Concreet bevat de CIF toolset transformaties naar mCRL2 [75] en UPPAAL [76], bekende state-of-the-art model checkers die formele verificatie ondersteunen.

Voor validatie biedt CIF een simulator aan. Deze kan worden gebruikt voor automatische simulatie, waarbij een aantal scenario’s kan worden doorlopen. De simulator kan dan aangeven of deze wel of niet door het besturingsmodel worden toegestaan. De simulator staat echter ook interactieve simulatie toe, waarbij handmatig het scenario gekozen kan worden. Samen met visualisaties kan op deze manier inzicht gekregen worden in het gedrag van diverse scenario’s (zie ook Sectie 7.2.5).

De Eclipse ESCET toolkit is beschikbaar voor Windows, Linux en macOS. Naast de Eclipse ESCET IDE, waarin alle tools zijn geïntegreerd, zijn de tools ook beschikbaar voor de command line, zodat deze *headless* op een server, bijvoorbeeld een build server, kunnen draaien.

7.11.2 Toolkeuze en alternatieven

Er zijn tal van tools, zowel commerciële tools als open source tools, die diverse aspecten van model-gebaseerd ontwikkelen (MBE) ondersteunen. Veel model-gebaseerde tools zijn opgebouwd rond UML/SysML, met wisselende mate van mogelijkheden en formele basis. De toolsuite SCADE [77] (Safety Critical Application Development Environment) van ANSYS kan worden gezien als een state-of-the-art voorbeeld van een dergelijke tool. Het ondersteunt model-gebaseerd ontwerpen van embedded software en virtual prototypes, met onder andere simulatie, verificatie en codegeneratie. Het wordt ingezet bij diverse industrieën, waaronder voor auto’s, vliegtuigen, defensie, spoorwegen en nucleaire installaties. Ook wat betreft certificatie kan SCADE gezien worden als state-of-the-art. De codegeneratoren van SCADE zijn door meerdere instanties uit meerdere domeinen gecertificeerd, onder andere op diverse *Safety Integrity Level* (SIL) niveaus. Een voorbeeld van een Nederlands bedrijf dat een op UML gebaseerde toolsuite levert is de Cordis Suite van Cordis Automation BV [78, 79, 80]. De Cordis Suite ondersteunt eveneens

modelleren, simulatie en codegeneratie. Met de TU/e wordt gewerkt[2] aan het integreren van verificatie [81]. Er zijn ook bedrijven met tools die verificatie-gebaseerd ontwikkelen (VBE) ondersteunen, waaronder Nederlandse bedrijven, zoals Verum met de Dezyne tool [82] en Cocotec met het Coco Platform [83].

De meeste MBE tools ondersteunen echter geen besturingssynthese, waarmee de keuze aanzienlijk kleiner wordt. Maar, er zijn wel andere tools die besturingssynthese ondersteunen, zoals Supremica [84, 85, 86]. Voor een overzicht, zie onder andere [87].

Rijkswaterstaat is in aanraking gekomen met SBE vanuit de samenwerking met de TU/e, en heeft daarbij logischerwijs ook kennisgemaakt met de CIF toolset, en recentelijk de Eclipse ESCET toolkit. Er is echter ook voor de langere termijn voor deze optie gekozen (zie Sectie 6.2.7). Hier zijn een aantal redenen voor.

Een van de redenen is de combinatie van gewenste eigenschappen. Zo ondersteunt ESCET het gehele ontwikkelproces van besturingssoftware inclusief besturingssynthese, via een enkele modelleertaal, en heeft die taal concepten om leesbare en herbruikbare modellen te creëren. Een tweede reden is de focus in het Eclipse ESCET project op het ontwikkelen van een industrieel toepasbare toolkit. Diverse andere synthese tools ondersteunen of alleen synthese of synthese met een beperkt aantal andere stappen van het ontwikkelproces, zoals bijvoorbeeld verificatie en simulatie, in plaats van het gehele ontwikkelproces. Ook zijn de meeste andere synthese tools academisch ingestoken, en besteden daarbij doorgaans geen of minder aandacht aan zaken als het gemak waarmee kan worden gemodelleerd, controles op veel gemaakte fouten en begrijpbare foutmeldingen.

Een derde reden is de kernwaardes van de Eclipse Foundation: transparantie, openheid, meritocratie en leveranciersonafhankelijkheid. De Eclipse Foundation streeft naar open ecosystemen, waarin diverse partijen kunnen samenwerken, en niemand wordt buitengesloten. Zo kan in het ecosysteem (zie Sectie 7.14) samen worden gewerkt aan verdere innovaties rondom de toolset, en kan iedereen hier aan bijdragen, inclusief onder andere Rijkswaterstaat, de TU/e en marktpartijen. Verder is het binnen het open ecosysteem mogelijk voor meerdere commerciële partijen om commerciële ondersteuning aan te bieden voor de toolkit. Hiermee wordt leveranciersafhankelijkheid voorkomen, en wordt niemand buitengesloten omdat alle partijen mee kunnen en mogen doen, een belangrijke eis voor een overheidsinstantie zoals Rijkswaterstaat.

7.11.3 Kwaliteit van de Eclipse ESCET toolkit

Voor de garanties die formele methoden als supervisor synthese en codegeneratie geven, is het belangrijk dat de wiskundig bewezen algoritmen juist worden geïmplementeerd in de tools, zodat deze garanties ook gelden voor de tools. Daarnaast is het belangrijk voor toepassing in de industrie dat er hoogwaardige tools worden gebruikt, die goed werken en weinig bugs bevatten.

Hoewel Eclipse ESCET van oorsprong een academische toolkit is, ontstaan uit werk van de TU/e, is het doel hierbij altijd geweest om een dergelijke kwalitatief hoogwaardige toolkit te ontwikkelen, die geschikt is om in te zetten in de industrie. Zowel in de jaren dat de TU/e de ontwikkeling voor rekening nam, als momenteel bij de Eclipse Foundation, wordt daarom elke wijziging zorgvuldig bekeken en bekritiseerd door medeontwikkelaars alvorens deze wordt opgenomen in de tool. Verder heeft de CIF toolset een uitgebreide set tests, zowel unit tests als integration tests. Deze dienen er ten eerste voor om te zorgen dat de functionaliteit juist is geïmplementeerd, en dat bijvoorbeeld algoritmen correct functioneren. Maar ze zorgen er ook voor dat eventuele wijzigingen alleen worden toegelaten als daarbij de bestaande functionaliteit nog blijft werken. Daarnaast zijn er codestijlregels om de leesbaarheid en onderhoudbaarheid van de code te waarborgen, en worden er tal van automatische controles gedaan tijdens het bouwproces. De uitgebreide reviews, grote hoeveelheid tests, stijlregels en automatische controles zijn voorbeelden van processen en regels die worden ingezet om de kwaliteit van de software hoog te houden.

[2] Deze werkzaamheden vinden plaats in de context van het Europees onderzoeksproject ITEA Machinaide, waaraan ook TNO deelneemt [101].

De CIF toolset wordt sinds 2007 ontwikkeld door de TU/e en is sinds 2020 een onderdeel van de Eclipse ESCET toolkit, ontwikkeld in het Eclipse ESCET open source project bij de Eclipse Foundation. Daarmee zijn het tools die al jaren bestaan en waaraan al jaren is ontwikkeld. Ook is het door de jaren heen in diverse contexten ingezet (zie Sectie 7.9). Verder is de ervaring van de TU/e dat het gebruik van de tools in het onderwijs vaak problemen kan blootleggen, aangezien studenten die voor het eerst in aanraking komen met de tools ze dikwijls op verrassende manier inzetten. Studenten zijn daarmee een goede toevoeging op de bestaande testen in de CIF toolset. De tools zijn in het afgelopen decennia door vele honderden studenten gebruikt.

Hoewel er in het Eclipse ESCET project veel aandacht is voor het hoog houden van de kwaliteit van de tools, is het volledig garanderen van afwezigheid van fouten in niet-triviale software tools een utopie [9]. Dit geldt echter niet alleen voor Eclipse ESCET, maar net zo goed voor PLC-compilers.

7.11.4 Certificatie van tools

Afhankelijk van het gewenste veiligheidsniveau, kan het nodig zijn tools te certificeren voor bijvoorbeeld ISO of NEN normen, of bepaalde SIL niveaus. Dit kan extra zekerheid geven over de correctheid van de tools. Te denken valt bijvoorbeeld aan het certificeren van Eclipse ESCET, waaronder de synthese tools en codegeneratoren, maar ook aan de certificering van PLC-omgevingen van PLC-fabrikanten. De Eclipse ESCET toolkit, en daarmee de CIF toolset, zijn momenteel niet gecertificeerd. Er zijn wel andere codegeneratoren die zijn gecertificeerd (zie Sectie 7.11.2).

Rijkswaterstaat hanteert voor onder ander beweegbare bruggen en sluizen doorgaans SIL niveau 1 of 2. Uit de analyse van CGI [4] komt geen noodzaak naar boven om tools te certificeren (zie ook Sectie 7.6.4). In de TOPAAS methode die Rijkswaterstaat gebruikt voor faalkansbepaling van softwaremodules leidt het gebruik van gecertificeerde compilers tot lagere faalkansen [41]. Daarbij wordt door de auteurs van die methode opgemerkt dat volgens de IEC-61508 norm uit 1998 een gecertificeerde compiler verplicht is voor SIL niveau 3 en 4. Daarmee lijkt certificatie van tools vooralsnog voor Rijkswaterstaat geen harde eis. TNO heeft echter geen compleet onderzoek gedaan naar de eisen die gesteld worden in de diverse richtlijnen en normen waar Rijkswaterstaat en de objecten van Rijkswaterstaat aan moeten voldoen.

Mocht het alsnog gewenst zijn tools te certificeren, dan zullen specifieke versies gecertificeerd worden. Vervolgens moet dan alleen die gecertificeerde versie worden gebruikt. Echter, tools worden constant aangepast, met verbeteringen en nieuwe functionaliteit. Bij keuze voor certificatie, moet worden bekeken hoe om te gaan met volgende versies, of er dan hercertificering nodig is, wie daar verantwoordelijk voor is, en hoe dat wordt ingeregeld. Ook de rol van eventuele open source projecten en/of commerciële bedrijven die de tools aanbieden of ondersteunen, zou daar in moeten worden meegenomen.

Certificatie kan een lastig en langdurig proces zijn. Als er voor certificatie gekozen wordt, is het zaak de juiste balans te vinden tussen de moeite die certificeren kost en de extra zekerheden die ermee kunnen worden verkregen. Hierbij kan worden gekeken naar welke softwaretools certificatie nodig hebben, en in welke mate. Voor tools die automatisch iets genereren (zoals synthese tools, codegeneratoren, en compilers) zouden strengere eisen gesteld kunnen worden, om het vertrouwen hierin te verhogen. Dit in tegenstelling tot tools die niet direct iets genereren als uitvoer wat in vervolgstappen van het ontwikkelproces weer als invoer wordt gebruikt. Zo is het certificeren van simulatoren wellicht niet of in mindere mate nodig. Voor tools die later in het ontwikkelproces worden gebruikt, kunnen ook strengere eisen gesteld worden, omdat de uitvoer ervan niet of in mindere mate in latere stappen nog geverifieerd en gevalideerd zal worden, waardoor het risico bij de aanwezigheid van fouten in dergelijke tools hoger is.

7.11.5 Commerciële ondersteuning

Voor bedrijven is commerciële ondersteuning van de software over het algemeen een vereiste. Als er een probleem is met een tool, moet er iemand kunnen worden benaderd die dat binnen een redelijke termijn oplost. Hoewel er voor model-gebaseerd ontwikkelen (MBE) en verificatie-gebaseerd ontwikkelen (VBE)

keuze is uit diverse commerciële bedrijven die hier hun eigen oplossing voor aanbieden (zie Sectie 7.11.2), is er op dit moment nog geen commercieel ondersteunde tool voor besturingssynthese (SBE).

Een open source project biedt geen commerciële ondersteuning. Mocht Rijkswaterstaat SBE breder in willen gaan zetten, dan zal een partij gevonden moeten worden welke bereid is commerciële ondersteuning aan te bieden voor de te gebruiken tools. Voor Eclipse ESCET is dit in een community meeting besproken [88]. Hierbij waren naast Rijkswaterstaat, TU/e en TNO, onder andere ook de ICT Group [89] en Capgemini Engineering [90] aanwezig. Die laatste twee partijen hebben aangegeven open te staan hier verder naar te kijken. Uiteraard is het hierbij essentieel voor een dergelijke partij dat er marktpotentie is, waarbij onder andere gekeken zal worden naar het ecosysteem rondom een bepaalde tool (zie Sectie 7.14).

Het lijkt er op dat bedrijven momenteel de kat uit de boom kijken, en wachten totdat een ander bedrijf SBE volledig omarmt en commerciële ondersteuning regelt. Hiermee ontstaat een kip-ei probleem, in de zin dat geen enkel bedrijf SBE omarmt omdat er geen commerciële ondersteuning is, en er geen commerciële ondersteuning komt omdat geen enkel bedrijf SBE omarmt en er daarmee dus onvoldoende markt is voor commerciële ondersteuning. Sterk gerelateerd hieraan is ook het feit dat marktpartijen die dienstverlening bieden op het gebied van ontwikkeling van (besturings)software nog niet gebruikmaken van SBE en Eclipse ESCET en daarom ook nog niet over de benodigde kennis en kunde beschikken (zie Sectie 7.12). Mocht Rijkswaterstaat met marktpartijen verder willen gaan met SBE, dan is het raadzaam om actief de commerciële ondersteuning te regelen en daarmee deze kip-ei situatie te doorbroken. Er kan bijvoorbeeld contact worden gezocht met een softwarehuis of softwaredienstverlener. Die moet dan dus wel de benodigde kennis en kunde hebben om de commerciële ondersteuning te bieden. Rijkswaterstaat kan hierbij eventueel ook samen optrekken met andere bedrijven die SBE willen adopteren.

Rijkswaterstaat heeft aangegeven momenteel op zoek te zijn naar een partij die commerciële ondersteuning kan bieden. Een eerste offerteaanvraag is reeds verstuurd.

7.11.6 Conclusie rondom tooling

Als er voor SBE wordt gekozen (zie ook Sectie 7.15), dan kan Eclipse ESCET een goede keuze zijn, vanwege de combinatie van eigenschappen rondom modelleren en het ondersteunen van het gehele ontwikkelproces, de focus op het creëren van een industrieel toepasbare toolkit, en het open ecosysteem. Hierbij moet dan nog wel worden gekeken naar vooral de commerciële ondersteuning, die momenteel nog ontbreekt. Rijkswaterstaat is momenteel op zoek naar een partij die commerciële ondersteuning kan bieden, en dient daarbij de overwegingen uit Sectie 7.11.5 mee te nemen. Verder dienen de aanbevelingen van CGI omtrent toolondersteuning geadresseerd te worden (zie Sectie 7.6.4).

Het zou zo kunnen zijn dat het Rijkswaterstaat en/of marktpartijen om de een of andere reden niet lukt om Eclipse ESCET (volledig) te introduceren, bijvoorbeeld omdat het niet lukt commerciële ondersteuning te regelen, of vanwege een van de andere aspecten die in dit rapport worden besproken. Dan is het een mogelijk alternatief om niet volledig in te zetten op Eclipse ESCET, maar dit te combineren met een of meerdere andere tools, en eventueel (gedeeltelijk) terug te schakelen naar MBE of VBE in plaats van volledig SBE in te zetten (zie ook Sectie 7.15). Een voorbeeld hiervan zou kunnen zijn om voor het modelleren en/of codegeneratie een toolsuite zoals SCADE te gebruiken.

7.12 Kennis en kunde

Rijkswaterstaat voorziet SBE toe te passen samen met marktpartijen, om het proces schaalbaar te maken en het grote aantal objecten te kunnen renoveren en vervangen (zie Sectie 6.2.1). Er wordt een grote rol voorzien voor marktpartijen, maar Rijkswaterstaat zal, zeker de eerste jaren, de regie moeten voeren (zie Secties 6.2.3 en 6.2.4). Hiervoor is het essentieel dat zowel Rijkswaterstaat zelf als marktpartijen hiervoor voldoende kennis en kunde hebben, dus weten wat SBE is en hoe het werkt, maar ook voldoende ervaring hebben met het op de juiste manier toepassen van SBE. Wat betreft Rijkswaterstaat zelf geldt dit in eerste instantie voor het Centraal SBE team (zie Sectie 6.2.8). Maar ook als de taken van dat team later belegd

worden in de lijnorganisatie, en dus verdeeld zijn over een groter aantal personen, zal dit onverminderd van toepassing zijn.

Het toepassen van SBE is iets dat om een bepaalde mindset vraagt. Te denken valt aan het gebruik van model-gebaseerd ontwerpen, het modelleren van gedragsspecificaties en besturingseisen, en het gebruik van formele methoden. Maar zeker ook het denken in restricties (wat moeten requirements verbieden ten opzichte van de plants) in plaats van wat er moet, kan of mag gebeuren (acties in een bepaalde volgorde of vanuit een bepaalde toestand), is echt een andere manier van denken dan wat men vaak gewend is. Voldoende kennis is nodig om dit juist uit te voeren. Daarnaast is voldoende ervaring nodig met onder andere het modelleren, synthetiseren, simuleren, etc, om te weten waar de aandachtspunten liggen, en hoe hiermee om te gaan. De personen die SBE uitvoeren moeten hier dus voldoende kundig in zijn. Ze moeten voldoende vertrouwen hebben in hun eigen kennis en kunde, maar ook in de tools en technieken die worden gebruikt, wat ook weer een zekere ervaring vereist.

Bij een verandering van werkwijze is het belangrijk dat het juiste personeel met de juiste kennis en kunde aanwezig is in het bedrijf. Hierbij is extra training en/of omscholing voor (een gedeelte van) het personeel dan ook geen uitzondering. Rijkswaterstaat is bezig diverse personen die kennis en ervaring te geven, onder andere door ze zelf SBE toe te laten passen als onderdeel van het project waarvoor TNO dit rapport oplevert (zie ook Sectie 6.3.1). Om de benodigde kennis en kunde op te doen is de TU/e bezig een meerdaagse online cursus op te zetten. Deze cursus bevat diverse modules, waarmee de principes en het proces van SBE geleerd kunnen worden, en er op diverse (simpele) situaties geoefend kan worden. Het bevat onder andere modules omtrent modelleren, synthetiseren en simuleren. De TU/e is voornemens meer modules toe te voegen. Onderzocht wordt of deze cursus onderdeel kan worden van het Eclipse ESCET project. De ontwikkeling van trainingsmateriaal voor online zelfstudie is echter slechts een eerste stap. Er is nog geen cursus waarbij samen met een trainer het materiaal eigen gemaakt kan worden. Daarmee kan uiteindelijk een hoger niveau van kennis en kunde kan worden vergaard, omdat dit de mogelijkheid geeft tot het stellen van vragen, maar ook het leren uit feedback op gemaakte modellen. Hiervoor zouden dan wel trainers moeten worden opgeleid. Verder is de online cursus nog niet af, en is deze nog niet of amper getest op geschiktheid. Hierbij valt verder op te merken dat voor marktpartijen wellicht een ander soort cursus nodig is dan voor studenten van de universiteit, en dat de cursus dus ook op geschiktheid voor die doelgroep moet worden getest.

Verder voorziet Rijkswaterstaat in een groeipad (zie Sectie 6.2.5), waarbij marktpartijen in proefprojecten, met ondersteuning en begeleiding vanuit Rijkswaterstaat, de benodigde kennis en kunde in de praktijk kunnen opdoen. Op die manier kan een marktpartij ook aantonen voldoende deskundig te zijn in SBE om dit succesvol te kunnen toepassen. Rijkswaterstaat voorziet ook een kwalificatiesysteem (zie Sectie 6.2.5), waarmee dergelijke marktpartijen dan voor SBE gekwalificeerd kunnen worden, om het vervolgens zelfstandig in te zetten bij volgende projecten. Rijkswaterstaat zorgt hierbij voor een open systeem, zodat alle geïnteresseerde partijen kunnen meedoen. Er zijn op dit moment onvoldoende marktpartijen met substantiële SBE kennis en kunde. Daarnaast is er nu slechts voorzien in drie proefprojecten voor verschillende toepassingen van SBE, met ieder een evaluatie voorafgaand aan bredere invoering (zie Sectie 6.2.6). Er zal goed gekeken moeten worden hoeveel marktpartijen dit traject gaan doorlopen, hoeveel deskundige personen benodigd zijn voor de VenR uitdagingen van Rijkswaterstaat, hoeveel proefprojecten nodig zijn om voldoende personen bij de marktpartij de kennis en kunde te laten opbouwen, binnen welk tijdsbestek dat dient plaats te vinden, en hoe groot de investeringen zijn die marktpartijen daarvoor moeten doen. Het is in dit stadium nog niet vastgesteld of dit haalbaar en realistisch is, en of daarmee de grote VenR uitdaging kan worden beslecht.

Gezien de toepassing van formele methoden, en de gewenste mindset welke fundamenteel anders is dan wat bijvoorbeeld PLC-programmeurs gewend zijn, is het in huis hebben van voldoende (universitair) geschoolde experts die kunnen assisteren en sturen een belangrijke factor voor succes. Deze zullen wellicht aangetrokken moeten worden. Rijkswaterstaat heeft hier al stappen in gezet, door personen met ervaring met SBE aan te nemen, en op te nemen in het Centraal SBE team. Echter, in de toekomst worden

deze taken belegd in de lijnorganisatie. Er zal goed gekeken moeten worden hoeveel personen dan de kennis en kunde met betrekking tot SBE dienen te hebben, hoeveel van hen universitair geschoold dienen te zijn, in welke mate ze die kennis en kunde moeten hebben, hoeveel projecten Rijkswaterstaat (gedeeltelijk) zelf denkt te gaan uitvoeren om die kennis en kunde op te bouwen en te behouden, en of dit haalbaar en realistisch is. Daarbij moet ook rekening worden gehouden met of er genoeg van dergelijk geschoold en om te scholen personeel is, voor zowel Rijkswaterstaat als marktpartijen. De TU/e is momenteel de enige Nederlandse universiteit met SBE in het curriculum, en zoals aangegeven zijn er nog geen cursussen die gevolgd kunnen worden. Rijkswaterstaat zal een visie omtrent (om)scholing en de arbeidsmarkt moeten opstellen.

Concluderend kan gesteld worden dat er wat betreft het opbouwen van kennis en het opdoen van ervaring (kunde) nog diverse stappen gezet moeten worden. De eerste plannen rondom opleidingsmateriaal zijn gezet, maar vormen slechts een eerste stap. Zeker wat betreft de kennis en kunde bij marktpartijen is er nog werk te verzetten. Om de kennis en kunde bij zowel Rijkswaterstaat zelf als bij marktpartijen op het gewenste niveau te krijgen moeten gedegen plannen worden gemaakt en uitgevoerd.

7.13 Veranderen van bedrijfsprocessen en bedrijfscultuur

Het adopteren van model-gebaseerd ontwikkelen in het algemeen, en SBE in het bijzonder, vereist nogal wat veranderingen in het ontwikkelproces van besturingssoftware, waaronder aan het gebruik van model-gebaseerd ontwerpen, het modelleren van gedragsspecificaties en besturingseisen, en het gebruik van formele methoden. Het kan lastig zijn voor een bedrijf om over te gaan op een andere werkwijze en bijbehorende bedrijfscultuur. Voor iemand die zijn of haar werk al tientallen jaren op een bepaalde manier doet en gewend is dat op die manier te doen, is het namelijk niet altijd eenvoudig om iets nieuws te leren, want het doorbreken van oude patronen gaat vaak niet vanzelf. Daarnaast zit wellicht niet iedereen te wachten op een andere manier van werken, of zijn er mogelijk zelfs een aantal personen in het bedrijf die helemaal niet bereid zijn om te veranderen. Het creëren van voldoende draagvlak in de organisatie, bij zowel Rijkswaterstaat als bij marktparijen, is dus essentieel [1].

Zowel Rijkswaterstaat als marktpartijen dienen daarom aan goed verandermanagement (*change management* in het Engels) te doen [91]. Het is daarbij belangrijk dat de gevolgen voor het bedrijfs- en ontwikkelproces alsmede de bedrijfscultuur goed in kaart gebracht worden. Hier moeten de voors en tegens (die per organisatie kunnen verschillen) goed worden afgewogen, en eventuele pijnpunten worden geïdentificeerd en weerlegd [44, 92].

Maar verandermanagement bevat tal van andere aspecten. Zo is het belangrijk duidelijke doelen te stellen, voldoende meetpunten in te bouwen in het proces, en bij (deel)evaluaties de juiste evaluatiecriteria te hanteren [1]. Daarnaast moet er goed en open worden gecommuniceerd over de veranderingen, en de te verwachte effecten ervan, zowel de voordelen als de nadelen. En er moet worden nagedacht over wat te doen als de gewenste verandering niet slaagt, of niet geheel slaagt. Wat doe je met die werknemers die niet in staat blijken mee te komen met de verandering, of niet mee willen in de verandering.

Een ander belangrijk aspect van verandermanagement is dat een organisatie genoeg tijd moet nemen voor een dergelijke ingrijpende verandering. Als onderdeel daarvan moet iedereen voldoende tijd gegeven worden om aan de verandering te werken, onder andere om kennis en kunde op te bouwen (zie Sectie 7.12). De ervaring in de samenwerking tussen TU/e en Rijkswaterstaat leert dat studenten SBE snel oppikken. Ze kunnen zonder veel domeinkennis van infrastructurele systemen in een kort tijdsbestek van een paar maanden zich deze techniek eigen maken, en toepassen op objecten van Rijkswaterstaat, om tot een gedegen besturing voor (een deel van) een object te komen. Hierbij moet wel worden opgemerkt dat deze studenten in het algemeen al kennis en kunde hebben opgedaan via vakken die ze hebben gevolgd aan de TU/e, en dat ze worden begeleid door deskundige medewerkers van de universiteit. De vraag is of deze ervaring 1-op-1 is door te zetten naar Rijkswaterstaat zelf, en naar marktpartijen. Studenten hebben nog geen jarenlange werkervaring en vaste gewoontes. Het zal duidelijk moeten worden in hoeverre medewerkers van Rijkswaterstaat en marktpartijen in staat zijn om een dergelijke stap te maken, en of dat

ook in een aantal maanden kan. SBE is namelijk een vorm van model-gebaseerd ontwikkelen, en de introductie daarvan is een veranderproces dat meerdere jaren kan duren. Zo leert althans de ervaring van zowel TNO in de Nederlandse high-tech industrie als ook de literatuur [44, 93].

Gedeeltelijk zal het een en ander duidelijk worden in de drie door Rijkswaterstaat voorziene proefprojecten (zie Sectie 6.2.6). Hierbij kan de gefaseerde introductie van SBE, zoals door Rijkswaterstaat is voorzien, de risico's significant beperken. Echter, het is in dit stadium nog onduidelijk in hoeverre Rijkswaterstaat en marktpartijen in staat zullen zijn dit veranderproces te laten slagen. Het niet van de grond komen van de benodigde proces- en organisatieverandering is ook een belangrijk punt van aandacht in het BIT-advies rondom 3B bouwblokken [1]. Het is dan ook essentieel dat Rijkswaterstaat het veranderproces voor SBE verder uitwerkt, waarbij de hier besproken aandachtspunten dienen te worden meegenomen. Daarbij dient dan ook naar de tijdslijn voor dit veranderproces te worden gekeken, en of er voldoende marktpartijen zijn die deze transitie in het door Rijkswaterstaat voorziene tijdspad kunnen doorlopen.

Rijkswaterstaat heeft in het verleden vaker een overstap gemaakt naar alternatieven manieren van werken, door het adopteren van nieuwe technologieën. Mogelijk kan Rijkswaterstaat ook lering trekken uit de lessen die destijds zijn geleerd. Hoewel de introductie van SBE ongetwijfeld grote verschillen heeft met deze technologieën, kan hierbij bijvoorbeeld worden gedacht aan de overgang van schaalmodellen op wiskundige modellen, en de introductie van PLC's.

Het gedegen insteken van het veranderproces is in ieder geval essentieel voor het slagen ervan. Rijkswaterstaat kan naast de geplande drie proefprojecten (zie Sectie 6.2.6) overwegen om SBE nog meer gefaseerd te introduceren (zie Sectie 7.1.3). Ook kan het samen met marktpartijen bekijken hoe de adoptie van SBE het beste kan worden ingestoken. Verder kan het delen van ervaringen met andere partijen die een dergelijke transitie hebben doorlopen van grote waarde zijn, om daar lering uit te trekken, en niet in dezelfde valkuilen te vallen. Ook kan contact worden gezocht met andere partijen die voor een soortgelijke keuze staan, om samen op te trekken. Voor zowel partijen die de keuze al gemaakt hebben, als die nog voor de keuze staan, kan gedacht worden aan andere partijen met kritieke infrastructuur, zoals het spoor, maar ook aan andere domeinen, zoals de high-tech industrie. Daarnaast kan zowel op landelijk, provinciaal als gemeente niveau worden gezocht naar dergelijke partijen, en zowel in binnen als buitenland. Ten slotte kan worden overwogen een externe partij in te schakelen die is gespecialiseerd in het begeleiden van bedrijven tijdens dergelijke transities, om de risico's verder te beperken.

7.14 Ecosysteem

Bij het adopteren van een nieuwe methode (zoals SBE) of tool (zoals Eclipse ESCET) is het goed om te kijken naar het ecosysteem rondom de methode/tool. Te denken valt hierbij aan hoeveel onderzoek er naar de methode wordt gedaan, hoeveel ontwikkelaars er zijn die de tool onderhouden en verder ontwikkelen, hoeveel gebruikers er zijn van de tool, hoe actief die gebruikers hun kennis en ervaring delen, hoeveel bedrijven de methode/tool inzetten in hun dagelijks werk, etc. Hoe meer een methode/tool wordt toegepast, en hoe meer mensen er mee bezig zijn, hoe meer kans dat het een goede methode/tool is. Het geeft een mate aan van volwassenheid. Het is daarmee ook voor SBE een indicatie van hoe kansrijk het is dat bredere toepassing ervan gaat slagen, niet alleen voor Rijkswaterstaat en marktpartijen, maar voor iedere partij die overweegt SBE te gaan inzetten.

De SBE methode, en besturingssynthese als uniek onderdeel van deze methode, hebben hun oorsprong in de jaren '80 [13, 14]. Er wordt in diverse groepen in de wereld onderzoek naar gedaan [42]. De TU/e is daarmee niet de enige groep die hier actief onderzoek naar doet. Het is dan ook te verwachten dat het onderzoek naar beide nog velen jaren zal voortduren. Er kan dus voldoende influx van nieuwe onderzoeksresultaten worden verwacht.

In de oorsprong is Eclipse ESCET een toolkit die door de TU/e werd ontwikkeld, voordat het in 2020 een Eclipse Foundation open source project werd (zie ook Sectie 7.11). De TU/e is nog altijd nauw bij het project betrokken, zowel in onderzoek als in verdere ontwikkeling van de toolkit. Echter, er zijn meer personen bij

het open source project betrokken [94]. Officieel heeft het project momenteel twee project leads, Bert van Beek van de TU/e, en Dennis Hendriks van TNO. De project leads zijn ook automatisch committer, en mogen dus code wijzigingen doorvoeren. Verder is Albert Hofkamp van de TU/e committer. En de vierde committer is Ferdie Reijnen, voormalig PhD student van de TU/e en voormalig medewerker van Rijkswaterstaat, die nu werkzaam is bij Vanderlande Industries en nog altijd op persoonlijke titel actief is in het project. Naast deze officiële rollen staat het iedereen vrij bijdragen te doen aan het project. Dat gebeurt dan ook. Dit is op het moment echter wel beperkt tot een zeer klein aantal contributies en het aanmaken van issues in het issue systeem met bugs of ideeën ter verdere verbetering van de toolkit. De ontwikkeling leunt dus momenteel op een gering aantal ontwikkelaars. Daarmee kan in de huidige staat nog niet gesproken worden van een levendige open source community, en dit vormt een risico voor de continuïteit.

Vanuit de oorsprong wordt Eclipse ESCET veel gebruikt bij de TU/e (zie ook Sectie 7.9). Het wordt gebruikt voor onderzoeksdoeleinden, door zowel de wetenschappelijke staf als door studenten en promovendi. Studenten passen de tools ook toe in projecten in de industrie. Daarnaast komen grote groepen studenten in aanraking met de tool tijdens vakken, waaronder tijdens het bachelorvak *Model-based systems engineering* [73] en het mastervak *Supervisory Control* (zie ook Sectie 7.11.3). De TU/e is niet de enige universiteit waar Eclipse ESCET in het onderwijs wordt gebruikt [95]. Een volledig overzicht van op welke universiteiten de toolkit wordt gebruikt is momenteel niet beschikbaar.

Eclipse ESCET, danwel CIF, is door studenten veelvuldig ingezet bij diverse bedrijven (zie ook Sectie 7.9). Hoewel de ervaringen veelal positief zijn, is er voor zover TNO bekend geen enkele bedrijf dat SBE, al dan niet met behulp van Eclipse ESCET, in het ontwikkelproces inzet om via modelleren, synthetiseren, en codegeneratie er besturingssoftware mee te ontwikkelen. Het ontbreken van commerciële ondersteuning (zie Sectie 7.11.5) en kennis en kunde bij marktpartijen (zie Sectie 7.12) zijn hierbij belangrijke aandachtspunten.

Het is overigens niet zo dat Eclipse ESCET in het geheel nergens wordt gebruikt in de industrie. Zo wordt Eclipse ESCET gebruikt in de Eclipse Logistic Specification and Analysis Toolkit (LSAT™) [96, 97], een ander Eclipse Foundation open source project[3]. Dit wordt momenteel op beperkte schaal ingezet door de industrie om logistiek te optimaliseren [98].

Gezien dat iedereen Eclipse ESCET kan downloaden en gebruiken zonder dat dit wordt geregistreerd, is er geen volledig overzicht van hoeveel en waar precies Eclipse ESCET wordt gebruikt.

Concluderend kan gesteld worden dat het ecosysteem rondom Eclipse ESCET nog erg mager is, zowel wat betreft ontwikkelaars als wat betreft gebruikers en ondersteunende partijen. Zoals het ecosysteem er momenteel bij staat vormt het een bedreiging voor de continuïteit op de langere termijn. Een bijkomend risico is dat Rijkswaterstaat (met de marktpartijen die het inschakelt) de kans loopt om de enige partij te zijn die de toolkit in het dagelijks gebruik toepast. Als Rijkswaterstaat wil overgaan tot bredere inzet van SBE, en dan met name Eclipse ESCET, is het noodzakelijk om te investeren in het uitbreiden van het ecosysteem. Daarnaast dienen de aanbevelingen van CGI omtrent het SBE ecosysteem geadresseerd te worden (zie Sectie 7.6.4).

7.15 Ontwikkelproceskeuze MBE/VBE/SBE

Rijkswaterstaat overweegt als alternatief ontwikkelproces de overstap naar SBE. SBE is een vorm van model-gebaseerd ontwikkelen (MBE) gecombineerd met computer-ondersteund ontwerpen. Het heeft als uitgangspunt om correct-door-constructie besturingen te vervaardigen, door zo veel mogelijk stappen in het ontwikkelproces te automatiseren (zie ook Sectie 5.6).

Dit rapport gaat, gezien de vraagstelling vanuit Rijkswaterstaat, vooral in op de verschillen tussen traditioneel ontwikkelen en SBE. Echter, diverse van de in dit rapport besproken aspecten, en de potentiële

[3] De LSAT toolkit is mede door TNO ontwikkeld en TNO is nog altijd bij de ontwikkeling ervan betrokken.

voordelen en aandachtspunten die daarbij aan bod komen, gelden in bepaalde mate ook voor MBE en VBE.

In het spectrum van mogelijke ontwikkelprocessen, waarvan er een aantal zijn geschetst in Sectie 5.2, staat traditioneel ontwikkelen het meest links en SBE het meest rechts. Van links naar rechts (traditioneel ontwikkelen, MBE, VBE, SBE) gebruiken ze steeds meer automatisering en worden ze steeds meer ondersteund door automatisering via computers. Wat dat betreft is SBE de meest vergaande vorm van model-gebaseerd ontwikkelen waarbij gebruik wordt gemaakt van formele methoden en computer-ondersteund ontwerpen. Het biedt dan ook onder andere de meest vergaande automatisering, en daarmee in potentie de hoogste efficiëntie (zie ook Sectie 7.7), en heeft van de beschreven alternatieven de meeste wiskundige garanties (zie ook Sectie 7.5), wat kan leiden tot nog hogere kwaliteit en veiligheid (zie ook Sectie 7.6). Ook de TU/e heeft traditioneel ontwikkelen vergeleken met MBE en SBE en geconcludeerd dat in de basis SBE de beste kwaliteit, de hoogste evolueerbaarheid, en de laagste variabiliteit heeft [43].

Rijkswaterstaat overweegt SBE specifiek vanwege de diverse voordelen. Veiligheid is erg belangrijk voor de objecten van Rijkswaterstaat. Daarnaast moeten veel objecten worden gerenoveerd of vervangen, wat een grote uitdaging is, en waar efficiëntie als essentieel wordt gezien (zie ook Sectie 3.1). SBE biedt hiervoor in potentie de grootste voordelen ten opzichte van de andere in dit rapport besproken alternatieven.

De vergaande automatisering van SBE kan echter ook een keerzijde hebben, door minder menselijke betrokkenheid bij diverse stappen (zie Sectie 7.7). En met de meest vergaande automatisering en computer-ondersteuning verschilt SBE uiteindelijk ook het meest van traditioneel ontwikkelen zoals dat nu nog wordt ingezet door Rijkswaterstaat en marktpartijen. Voor SBE is onder andere het meeste kennis en kunde nodig, die gezien de grotere verschillen ook lastiger is op te doen (zie ook Sectie 7.12). Verder is SBE van de alternatieven nog het minst gemeengoed, met het kleinste ecosysteem (zie ook Sectie 7.14), en is commerciële ondersteuning voor SBE-tooling nog niet geregeld (zie ook Sectie 7.11).

Daarnaast is de ervaring van zowel TNO vanuit de samenwerking met de Nederlandse high-tech industrie, als vanuit de literatuur, dat de introductie van model-gebaseerd ontwikkelen (MBE), dus de stap van traditioneel ontwikkelen naar MBE, een grote stap is voor veel bedrijven [51]. Dit grotendeels vanwege de kennis en kunde die moet worden opgebouwd (zie ook Sectie 7.12), en vanwege de verandering in bedrijfsprocessen en bedrijfscultuur (zie ook Sectie 7.13). Als deze stap eenmaal is gezet, dan is de stap om meer formele methoden te gaan gebruiken, zoals formele verificatie met model checking, en daarmee van MBE naar VBE, relatief kleiner. De stap van VBE naar SBE is vermoedelijk kleiner dan van traditioneel ontwikkelen naar MBE, maar groter dan van MBE naar VBE. Bij SBE moet op een andere manier gedacht worden, in restricties die de requirements opleggen ten opzichte van de plants, wat specifieke kennis en kunde vereist (zie ook Sectie 7.12). Daarnaast heeft het grote impact op het ontwikkelproces (zie ook Secties 5.2 en 7.2). Er dient dus hoe dan ook zorgvuldig te worden overwogen hoe een nieuwe aanpak geïntroduceerd wordt in de organisatie (zie ook Sectie 7.13).

Mocht Rijkswaterstaat er echter niet (geheel) in slagen SBE en Eclipse ESCET te introduceren (voor het gehele ontwikkelproces), dan kan worden overwegen SBE beperkter in te voeren, of op bepaalde vlakken terug te schakelen naar VBE of MBE. Synthese is bijvoorbeeld niet altijd noodzakelijk (zie ook Sectie 7.10). Bij synthese voor de Oisterwijksebaanbrug [26] zijn de gespecificeerde requirements al voldoende om de brug te besturen. Synthese hoeft niets extra te verbieden ten opzichte van wat al in de modellen is gespecificeerd, wat te zien is doordat alle gesynthetiseerde extra condities in het supervisor model ‘true’ zijn [99]. Dit wordt uiteraard wel pas duidelijk nadat synthese is toegepast. Maar, hierbij is het dus ook mogelijk formele verificatie op de gespecificeerde modellen toe te passen, dus VBE toe passen, en zo dezelfde formele ‘*safety’* garanties (zie Sectie 5.6.3) te verkrijgen [16]. Bij bijvoorbeeld het Algeracomplex (sluis/brug combinatie) heeft synthese overigens wel een extra effect [100], en biedt daarmee ondersteuning bij het ontwerp, aangezien deze extra effecten dan weer als expliciete requirements gemodelleerd kunnen worden. Er kan dan dus ook worden gekozen voor een beperktere introductie van SBE, bijvoorbeeld wel voor verhogen van de kwaliteit van de requirements, maar niet voor codegeneratie

(zie ook Secties 6.2.2, 6.2.6 en 7.1.3). Uiteindelijk is het belangrijk altijd een gefaseerde introductie van SBE te gebruiken, keuzes te maken die goed passen bij de organisatie en de gestelde doelen (zie ook Sectie 7.13), en daarbij de voor- en nadelen van een aanpak goed mee te wegen.

8 Verantwoording

TNO heeft voor dit rapport gebruik gemaakt van diverse bronnen, die waar mogelijk zijn geciteerd. De niet geciteerde bronnen betreffen voornamelijk informatie die is verkregen uit rechtstreekse gesprekken die hebben plaatsgevonden met Rijkswaterstaat, de TU/e en CGI. Ook is hergebruik gemaakt van delen van een eerder rapport dat door de auteur van dit rapport is geschreven voor Rijkswaterstaat [53], en van publiek beschikbare informatie rondom SBE zoals van de Eclipse ESCET website [24], eveneens (grotendeels) geschreven door de auteur van dit rapport. Dit betreft met name Hoofdstuk 5, maar ook andere delen van het rapport, en is niet overal specifiek met bronvermelding aangegeven. Ook is voor met name Hoofdstuk 6 gebruikt gemaakt van de opdrachtomschrijving van Rijkswaterstaat aan TNO [3], en van de door Rijkswaterstaat als achtergrondinformatie aan TNO ter beschikking gestelde oplegmemo ‘marktinschakeling bij toepassing van SBE in projecten’ [22]. Verder is gebruik gemaakt van de eigen kennis en ervaring.

TNO heeft voor dit rapport een deel van de werkzaamheden uitbesteed bij de Technische Universiteit Eindhoven (TU/e). Specifiek is er een beroep gedaan op de expertise van prof. dr. Wan Fokkink en dr. Asia van de Mortel-Fronczak. Beiden zijn gespecialiseerd in model-based system engineering, en werkzaam in de Control Systems Technology groep van de faculteit Werktuigbouwkunde van de TU/e. Concreet gaat het hierbij om expertise voor de beantwoording van de hoofdvraag, met focus op het beschouwen van de onderliggende algoritmes en een onderbouwing geven over de betrouwbaarheid van het resultaat van de software synthese. Als onderdeel hiervan deelt TU/e, vanuit de jarenlange samenwerking met Rijkswaterstaat, ook specifieke ervaring met toepassing van software synthese bij Rijkswaterstaat en op infrastructurele systemen. Verder heeft de TU/e het TNO rapport gereviewd. Dit rapport blijft echter een TNO rapport, waarbij TNO verantwoordelijk is voor de uiteindelijke inhoud. Het rapport is ook door meerdere personen binnen TNO gereviewd.

De antwoorden vanuit CGI met betrekking tot vragen 4-8, en de aanbevelingen en overige conclusies uit het CGI rapport [4], zijn door TNO deels in dit rapport meegenomen (zie ook Secties 6.3.2 en 6.4). CGI heeft ook kennis genomen van draft versies van dit TNO rapport, en de inhoud daarvan meegenomen bij het schrijven van het CGI rapport. Deze samenwerking was expliciet vooraf voorzien [3]. Lezers van dit TNO rapport wordt aangeraden ook het CGI rapport te lezen, aangezien deze complementair zijn en samen alle door Rijkswaterstaat gestelde vragen beantwoorden.

De auteur van rapport, Dennis Hendriks, heeft informatica gestudeerd aan de TU/e. Hij heeft vervolgens ruim acht jaar gewerkt voor de TU/e, faculteit Werktuigbouwkunde. Daarbij heeft hij gewerkt in de groep en met de personen van die groep, waarmee ook Rijkswaterstaat de afgelopen jaren heeft samengewerkt rondom SBE. Hij heeft daarbij ruime ervaring opgedaan rondom model-gebaseerd en synthese-gebaseerd ontwikkelen, was een van de grondleggers van de CIF tooling, en was een van de twee ontwikkelaars die CIF namens de TU/e ontwikkelde. Sinds 2016 werkt hij voor TNO. Hij heeft het voorstel gedaan CIF als open source project onder te brengen bij de Eclipse Foundation, en als project manager dit open sourcing traject getrokken, met betrokkenheid van de TU/e, Rijkswaterstaat, TNO en ASML. Dit heeft geresulteerd in het Eclipse ESCET project zoals dat er nu is. Hij heeft nog altijd een leidende rol in dit open source project en is er daarmee nauw bij betrokken. Verder werkt hij, ook buiten het Eclipse ESCET project om, nog altijd samen met de TU/e.

Gezien de relaties tussen de auteur van dit rapport, de TU/e en het Eclipse ESCET project, behelst dit rapport een expertoordeel van TNO. TNO benadrukt dat gezien deze relaties, die vooraf aan Rijkswaterstaat bekend zijn gemaakt, en in tegenstelling tot wat gebruikelijk is voor TNO, dit rapport daarom geen (volledig) onafhankelijk oordeel betreft.

9 Referenties

[1] Adviescollege ICT-toetsing, „Definitief BIT-advies programma IA Sourcing," 2021.

[2] De minister van infrastructuur en waterstaat, „Kamerbrief bij adviesrapport Adviescollege ICT-Toetsing over programma Industriële Automatisering Sourcing," 2021.

[3] Rijkswaterstaat, „Opdrachtomschrijving TNO-deel Proof of Concept SBE 30 mei 2022," 2022.

[4] W. Geurts en A. Verhoeven, „Eindrapport Veiligheid Synthesis Based Engineering, Oisterwijksebaanbrug," CGI Nederland B.V., Versie 1.0, 2022.

[5] M. A. Goorden, L. Moormann, F. F. H. Reijnen, J. J. Verbakel, D. A. van Beek, A. T. Hofkamp, J. M. van de Mortel-Fronczak, M. A. Reniers, W. J. Fokkink, J. E. Rooda en L. F. P. Etman, „The Road Ahead for Supervisor Synthesis," in *International Symposium on Dependable Software Engineering: Theories, Tools, and Applications*, 2020.

[6] Eclipse Foundation, „Synthesis-based engineering," 2022-11-08. [Online]. Available: https://eclipse.org/escet/cif/synthesis-based-engineering.

[7] Wikipedia, Wikimedia Foundation, „Electronic design automation," 2022-10-17. [Online]. Available: https://en.wikipedia.org/wiki/Electronic_design_automation.

[8] Wikipedia, Wikimedia Foundation, „High-level synthesis," 2022-10-17. [Online]. Available: https://en.wikipedia.org/wiki/High-level_synthesis.

[9] S. McConnell, Code complete: a practical handbook of software construction, Design, 2004.

[10] M. E. Gooden, „Return on Investment for Complex Projects Utilizing Model Based Systems Engineering (MBSE)," 2016.

[11] J. Woodcock, P. G. Larsen, J. Bicarregui en J. Fitzgerald, „Formal methods: Practice and experience," *ACM computing surveys (CSUR),* vol. 41, nr. 4, pp. 1-36, 2009.

[12] J. Baeten, J. M. van de Motel-Fronczak en J. E. Rooda, „Integration of supervisory control synthesis in model-based systems engineering," in *Complex systems*, 2016.

[13] P. J. Ramadge en W. M. Wonham, „Supervisory control of a class of discrete event processes," *SIAM journal on control and optimization,* vol. 25, nr. 1, pp. 206-230, 1987.

[14] W. M. Wonham en P. J. Ramadge, „On the supremal controllable sublanguage of a given language," *SIAM Journal on Control and Optimization,* vol. 25, nr. 3, pp. 637-659, 1987.

[15] F. F. H. Reijnen, M. A. Goorden, J. M. van de Mortel-Fronczak en J. E. Rooda, „Modeling for supervisor synthesis - a lock-bridge combination case study," *Discrete Event Dynamic Systems,* vol. 30, nr. 3, pp. 499-532, 2020.

[16] M. A. Goorden, „Supervisory control synthesis for large-scale infrastructural systems," PhD thesis, Eindhoven University of Technology, 2019.

[17] F. Reijnen, *Putting supervisor synthesis to work,* PhD thesis, Eindhoven University of Technology, 2020.

[18] L. Moorman, *Logische besturingen voor tunnels: specificatie van componenten en eisen,* Presentatie, Eindhoven University of Technology, 2019.

[19] W. M. Wonham en K. Cai, *Supervisory control of discrete-event systems,* Springer, 2019.

[20] C. G. Cassandras en S. Lafortune, *Introduction to discrete event systems,* Springer, 2008.

[21] Eclipse Foundation, „Synthesis-based engineering example," 2022-11-14. [Online]. Available: https://www.eclipse.org/escet/cif/synthesis-based-engineering/example.html.

[22] Werkgroep Roadmap SBE, Rijkswaterstaat, „Oplegmemo marktinschakeling bij toepassing van SBE in projecten," 2022.

[23] F. Erens en K. Verhulst, „Architectures for product families," *Computers in industry,* vol. 33, nr. 2-3, pp. 165-178, 1997.

[24] Eclipse Foundation, „Eclipse ESCET," 2022-10-17. [Online]. Available: https://eclipse.org/escet.

[25] F. F. H. Reijnen, J. M. van de Mortel-Fronczak, J. E. Rooda en M. J. T. van Dinther, „Synthesis and implementation of supervisory control for infrastructural systems," in *Benelux Meeting on Systems and Control*, 2019.

[26] F. Reijnen, E.-B. Leliveld, J. van de Mortel-Fronczak, J. Dinther, J. Rooda en W. Fokkink, „Synthesized fault-tolerant supervisory controllers, with an application to a rotating bridge," *Computers in Industry,* vol. 30, p. 103473, 2021.

[27] F. F. H. Reijnen, T. R. Erens, J. M. van de Mortel-Fronczak en J. E. Rooda, „Supervisory controller synthesis and implementation for safety PLCs," *Discrete Event Dynamic Systems,* vol. 32, nr. 1, pp. 115-141, 2022.

[28] J. F. Groote, J. Keiren, A. van Rooij, V. Saralaya en A. J. Wijs, „Verificatie van de correctheid van de PLC-software van de Algerabrug," 2013.

[29] M. Leemans, R. P. J. Koolen, S. Cranen, S. E. Jurgens en J. F. Groote, „Analyse van besturingssystemen voor beweegbare bruggen (analysis of the control system for movable bridges)," 2014.

[30] B. Boehm en V. R. Basili, „Software defect reduction top 10 list," *Software engineering: Barry W. Boehm's lifetime contributions to software development, management, and research,* vol. 34, nr. 1, 2007.

[31] J. D. Thomson, „Images of e-fficiency: Integrative Applications of Transaction Costs and e-Procurement Delivery," in *9th International Business Research Conference Melbourne*, 2008.

[32] J. van Hegelsom, „Development of a 3D Digital Twin of the Swalmen Tunnel in the Rijkswaterstaat Project," arXiv:2107.12108v2, 2021.

[33] L. Moormann, J. van Hegelsom, J. M. van de Mortel-Fronczak, P. Maessen, W. J. Fokkink en J. E. Rooda, „Digital Twins for the Validation of Road Tunnel Controllers," in *ITA-AITES World Tunnel Congress, WTC2022 and 47th General Assembly*, 2022.

[34] F. F. H. Reijnen, J. M. van de Mortel-Fronczak en J. E. Rooda, „Data Logging and Reconstruction of Discrete-event System Behavior," in *ICARCV*, 2020.

[35] J. J. Verbakel, M. E. W. Vos de Wael, J. M. van de Mortel-Fronczak, W. J. Fokkink en J. E. Rooda, „A configurator for supervisory controllers of roadside systems," in *CASE*, 2021.

[36] L. Moorman, J. M. van de Mortel-Fronczak en J. E. Rooda, „Design of a Parameter-based Modeling Platform for Road Tunnel Supervisory Controllers," in *CCTA*, 2021.

[37] F. F. H. Reijnen, J. M. van de Mortel-Fronczak, M. A. Reniers en J. E. Rooda, „Design of a Supervisor Platform for Movable Bridges," in *CASE*, 2020.

[38] S. C. M. Knippenberg, L. F. P. Etman, J. E. Rooda, E. J. Houwing en J. A. Vogel, „Development of a Product Platform for Ship Locks Using DSM Methods," in *Internataional DSM Conference*, 2019.

[39] S. C. M. Knippenberg, L. F. P. Etman, J. E. Rooda, T. Wilschut en J. A. Vogel, „Towards A Module-based Product Platform For Ship Locks Using DSM Methods," in *Internataional DSM Conference*, 2020.

[40] S. C. M. Knippenberg, W. J. Pennings, L. F. P. Etman, J. E. Rooda en J. A. Vogel, „Configuring Ship Locks Using A Product Platform Based On DSM Methods," in *International DSM Conference*, 2021.

[41] E. Brandt en et al., „TOPAAS: Een structurele aanpak voor faalkans analyse van software intensieve systemen," 2011.

[42] W. M. Wonham, K. Cai en K. Rudie, „Supervisory control of discrete-event systems: A brief history," *Annual Reviews in Control,* vol. 45, pp. 250-256, 2018.

[43] L. Moormann, P. Maessen, M. A. Goorden, J. M. van de Mortel-Fronczak en J. E. Rooda, „Design of a Tunnel Supervisory Controller using Synthesis-Based Engineering," in *ITA-AITES World Tunnel Congress, WTC2020 and 46th General Assembly*, 2020.

[44] P. Baker, S. Loh en F. Weil, „Model-driven engineering in a large industrial context—Motorola case study," in *MODELS*, 2005.

[45] P. Balasubramanian en K. Prasad, „A fault tolerance improved majority voter for TMR system architectures," arXiv:1605.03771, 2016.

[46] Europese Unie, „Richtlijn 2006/42/EG," 2006.

[47] International Electrotechnical Commission, „IEC 62061: Safety of machinery - Functional safety of safety-related control systems," 2021.

[48] International Electrotechnical Commission, „IEC 61508-3: Functional safety of electrical/electronic/programmable electronic safety-related systems - Part 3: Software requirements," 2010.

[49] T. R. Erens, „Design and Implementation of Supervisory Controllers for Safety PLCs," in *MSc thesis, Eindhoven University of Technology*, 2019.

[50] E. R. Carroll en R. J. Malins, Sandia National Laboratories, 2016.

[51] J. Wesselius, J. van den Aker, R. Doornbos, T. Hendriks, J. Marincic en W. T. Suermondt, „MBSE in the High-Tech Equipment Industry," TNO Report 2022 R11504, 2022.

[52] T. Alameda en T. Tritsch, „Deployment of MBSE processes using SysML," U.S. Army Research, Development and Engineering Command, 2010.

[53] E. Langius, K. Helmholt, H. Hildmann, R. van Heijster, D. Hendriks en A. Smulders, „TNO-rapport 2019 R11759: Onafhankelijk strategisch onderzoek naar leveranciersonafhankelijkheid voor het 3B bouwblok voor bedienen, besturen en bewaken," TNO, 2019.

[54] M. Goorden, J. van de Mortel-Fronczak, M. Reniers en J. Rooda, „Structuring multilevel discrete-event systems with dependency structure matrices," in *CDC*, 2017.

[55] M. Goorden, J. van de Mortel-Fronczak, M. Reniers, W. Fokkink en J. Rooda, „Structuring multilevel discrete-event systems with dependence structure matrices," *IEEE Transactions on Automatic Control,* vol. 65, nr. 4, pp. 1625-1639, 2019.

[56] F. F. H. Reijnen, M. A. Goorden, J. M. van de Mortel-Fronczak en J. E. Rooda, „Supervisory control synthesis for a waterway lock," in *CCTA*, 2017.

[57] F. F. H. Reijnen, M. A. Reniers, J. M. van de Mortel-Fronczak en J. E. Rooda, „Structured synthesis of fault-tolerant supervisory controllers," *IFAC-PapersOnLine,* vol. 51, nr. 24, pp. 894-901, 2018.

[58] F. F. H. Reijnen, J. J. Verbakel, J. M. van de Mortel-Fronczak en J. E. Rooda, „Hardware-in-the-loop set-up for supervisory controllers with an application: the Prinses Marijke complex," in *CCTA*, 2019.

[59] M. Goorden, J. van de Mortel-Fronczak, K. van Eldik, W. Fokkink en J. Rooda, „Lessons learned in the application of formal methods to the design of a storm surge barrier control system," *IFAC-PapersOnLine,* vol. 25, nr. 28, pp. 93-99, 2022.

[60] C. Vissers, „Exploring supervisory control theory on the Maeslant barrier locomobiles," MSc thesis, Eindhoven University of Technology, 2022.

[61] W. Fokkink, M. Goorden, J. van de Mortel-Fronczak, F. Reijnen en J. Rooda, „Supervisor Synthesis: Bridging Theory and Practice," Computer, 2022.

[62] S. T. J. Forschelen, „Application of supervisory control theory to theme park vehicles," *IFAC Proceedings,* vol. 10, nr. 12, pp. 293-299, 2010.

[63] S. T. J. Forschelen, J. M. van de Mortel-Fronczak, R. Su en J. E. Rooda, „Application of supervisory control synthesis to theme park vehicles," in *Discrete Event Dynamic Systems: Theory and Applications*, 2012.

[64] R. J. M. Theunissen, M. Petreczky, R. R. H. Schiffelers, D. A. van Beek en J. E. Rooda, „Application of supervisory control synthesis to a patient support table of a magnetic resonance imaging scanner," *IEEE Transactions on Automation Science and Engineering,* vol. 11, nr. 1, pp. 20-32, 2013.

[65] B. van der Sanden, M. Reniers, M. Geilen, T. Basten, J. Jacobs, J. Voeten en R. Schiffelers, „Modular model-based supervisory controller design for wafer logistics in lithography machines," in *MODELS*, 2015.

[66] C. R. C. Torrico, A. B. Leal en A. T. Y. Watanabe, „Modeling and supervisory control of mobile robots: A case of a sumo robot," *IFAC-PapersOnLine,* vol. 49, nr. 32, pp. 240-245, 2016.

[67] L. Ouedraogo, R. Kumar, R. Malik en K. Akesson, „Nonblocking and safe control of discrete-event systems modeled as extended finite automata,” *IEEE Transactions on Automation Science and Engineering,* vol. 8, nr. 3, pp. 560-569, 2011.

[68] M. Goorden en M. Fabian, „No synthesis needed, we are alright already,” in *CASE*, 2019.

[69] M. Goorden, J. van de Mortel-Fronczak, M. Reniers, W. Fokkink en J. Rooda, „The impact of requirement splitting on the efficiency of supervisory control synthesis,” in *International workshop on formal methods for industrial critical systems*, 2019.

[70] M. Goorden, J. van de Mortel-Fronczak, M. Reniers, M. Fabian, W. Fokkink en J. Rooda, „Model properties for efficient synthesis of nonblocking modular supervisors,” *Control Engineering Practice,* vol. 112, p. 104830, 2021.

[71] D. A. van Beek, W. J. Fokkink, D. Hendriks, A. Hofkamp, J. Markovski, J. M. van de Mortel-Fronczak en M. A. Reniers, „CIF 3: Model-based engineering of supervisory controllers,” in *TACAS*, 2014.

[72] Eclipse Foundation, „Overview issue for multi-level synthesis,” 2022-10-17. [Online]. Available: https://gitlab.eclipse.org/eclipse/escet/escet/-/issues/318.

[73] Eindhoven University of Technology, „4TC00 Model-based systems engineering,” 2022-10-17. [Online]. Available: https://cstweb.wtb.tue.nl/4tc00/index.html.

[74] Eclipse Foundation, „ESCET Plcgen development step plan,” 2022-10-17. [Online]. Available: https://gitlab.eclipse.org/eclipse/escet/escet/-/issues/397.

[75] O. Bunte, J. F. Groote, J. J. A. Keiren, M. Laveaux, T. Neele, E. P. de Vink, W. Wesselink, A. Wijs en T. A. C. Willemse, „The mCRL2 toolset for analysing concurrent systems,” in *TACAS*, 2019.

[76] G. Behrmann, A. David, K. Larsen, J. Håkansson, P. Pettersson, W. Yi en M. Hendriks, „Uppaal 4.0,” Los Alamitos, CA: IEEE Computer Society, 2006.

[77] Ansys, „Embedded Software Development,” 2022-10-17. [Online]. Available: https://www.ansys.com/products/embedded-software.

[78] Cordis Automation BV, „Cordis Suite,” 2022-10-17. [Online]. Available: https://www.cordis-suite.com/.

[79] N. Roos, „Using low-code to create control software for a system you’ve never seen,” in *Bits & Chips*, 2022.

[80] N. Roos, „Smit Thermal Solutions goes low-code with Cordis Suite,” in *Bits & Chips*, 2022.

[81] A. Stramaglia en J. J. A. Keiren, „Formal Verification of an Industrial UML-like Model using mCRL2,” in *FMICS*, 2022.

[82] Verum, „Discover Dezyne,” 2022-10-20. [Online]. Available: https://www.verum.com/DiscoverDezyne.

[83] Cocotec, „Cocotec - Formal verification,” 2022-10-20. [Online]. Available: https://cocotec.io/.

[84] K. Akesson, M. Fabian, H. Flordal en R. Malik, „Supremica-an integrated environment for verification, synthesis and simulation of discrete event systems,” in *8th International Workshop on Discrete Event Systems*, 2006.

[85] R. Malik, K. Akesson, H. Flordal en M. Fabian, „Supremica--an efficient tool for large-scale discrete event systems," *IFAC-PapersOnLine,* vol. 50, nr. 1, pp. 5794-5799, 2017.

[86] supremica.org, „Supremica," 2022-10-17. [Online]. Available: http://supremica.org/.

[87] M. A. Reniers en J. M. van de Mortel-Fronczak, „An engineering perspective on model-based design of supervisors," *IFAC-PapersOnLine,* vol. 51, nr. 7, pp. 257-264, 2018.

[88] D. Hendriks, „Re: [escet-dev] Eclipse ESCET community meeting #1," 2022-10-17. [Online]. Available: https://www.eclipse.org/lists/escet-dev/msg00338.html.

[89] „ICT Group," 2022-10-17. [Online]. Available: https://www.ict.eu/nl.

[90] „Capgemini Engineering," 2022-10-17. [Online]. Available: https://capgemini-engineering.com/.

[91] R. Smith, D. King, R. Sidhu, D. Skelsey en others, The effective change manager's handbook: Essential guidance to the change management body of knowledge, Kogan Page Publishers, 2014.

[92] B. Akesson, J. Hooman, R. Dekker, W. Ekkelkamp en B. Stottelaar, „Pain-mitigation techniques for model-based engineering using domain-specific languages," in *Proceedings of the 6th International Conference on Model-Driven Engineering and Software Development - MOMA3N*, 2018.

[93] R. Raaijmakers, „MBSE isn't an overnight activity, it's an evolution," in *Bits & Chips*, 2022.

[94] Eclipse Foundation, „Eclipse ESCET™ (Supervisory Control Engineering Toolkit) - Who's Involved," 2022-10-17. [Online]. Available: https://projects.eclipse.org/projects/technology.escet/who.

[95] D. A. van Beek, „[escet-dev] Fwd: from Tiziano and Matteo on class experience using ESCET," 2022-10-17. [Online]. Available: https://www.eclipse.org/lists/escet-dev/msg00375.html.

[96] L. J. van der Sanden, Y. Blankenstein, R. R. H. Schiffelers en J. P. M. Voeten, „LSAT: Specification and analysis of product logistics in flexible manufacturing systems," in *CASE*, 2021.

[97] Eclipse Foundation, „Eclipse LSAT," 2022-10-17. [Online]. Available: https://eclipse.org/lsat.

[98] B. J. C. van Putten, L. J. van der Sanden, M. A. Reniers, J. P. M. Voeten en R. R. H. Schiffelers, „Supervisor synthesis and throughput optimization of partially-controllable manufacturing systems," *Discrete Event Dynamic Systems,* vol. 31, pp. 103-135, 2021.

[99] F. Reijnen, „OBB generated supervisor," 2022-10-17. [Online]. Available: https://github.com/ffhreijnen/OBB/blob/435af9e83ca9ad84ec36d431e3e9911a2f932c52/CIF-ESCET-Version/generated_files/1_Supervisor.cif.

[100] F. Reijnen, „Algera complex generated supervisor," 2022-10-17. [Online]. Available: https://github.com/ffhreijnen/AlgeraComplex/blob/cd7d387f1bd577199d364def0ee5913d150a0c3d/CIF%20Models/LockBridge/generated_files/Supervisor.cif.

[101] A. Pil, „Continue optimalisatie begint bij low code," in *Mechatronica & Machinebouw*, 2020.

[102] Merriam-Webster, „Plant," 2022-10-17. [Online]. Available: https://www.merriam-webster.com/dictionary/plant.