

Verslag Tinlab Advanced Algorithms

J. I. Weverink

...

9 april 2021



Inhoudsopgave

1	Inleiding	3
2	Requirements	3
2.0.1	Mode confusion?	3
2.0.2	Automatisering paradox	3
2.1	Requirements	3
2.2	specificaties	4
2.3	Het vier variabelen model	5
2.3.1	Monitored variabelen	5
2.3.2	Controlled variabelen	5
2.3.3	Input variabelen	5
2.3.4	Output variabelen	5
2.4	Rampen	6
2.4.1	Ramp 1	6
2.4.2	Ramp 2	6
2.4.3	Ramp 3	6
2.4.4	Ramp 4	6
2.4.5	Ramp 5	6
2.4.6	Ramp 6	6
3	Modellen	6
3.1	De Kripke structuur	7
3.2	Soorten modellen	7
3.3	Tijd	7
3.4	Guards en invarianten	7
3.5	Deadlock	7
3.6	Zeno gedrag	7
4	Logica	7
4.1	Propositielogica	7
4.2	Predicatenlogica	7
4.3	Kwantoren	7
4.4	Dualiteiten	7
5	Computation tree logic	7
5.1	De computation tree	7
5.2	Operator: AG	7
5.3	Operator: EG	7
5.4	Operator: AF	7
5.5	Operator: EF	8
5.6	Operator: AX	8
5.7	Operator: EX	8
5.8	Operator: $p \cup q$	8
5.9	Operator: $p \cap q$	8

5.10 Fairness	8
5.11 Liveness	8

1 Inleiding

Zie hier een referentie naar Royce [?] en nog een naar Clarke [?]. . .

2 Requirements

2.1 Requirements

Requirements zijn beschrijvingen over hoe een product zou moeten functioneren. Zo verandert de betekenis van een requirement als de machine in een andere omgeving wordt geplaatst. De requirements voor de verwarming van een ruimte bijvoorbeeld: Binnen moet het altijd warm zijn. In Nederland kunnen we zeggen dat 25°C als warm wordt aangezien. Terwijl op de noordpool dat op een lager punt zal zijn.

Anders gezegd zijn requirements geen harde eisen. Dit komt doordat de requirements zijn geformuleerd vanuit het perspectief van de opdrachtgever. De opdrachtgever kan de requirements geven zonder kennis te hebben van de machine die het moet gaan uitvoeren. De requirements die zijn opgesteld geven dan ook geen grenzen aan die overschreden kunnen worden.

Onder requirements zijn er verschillende soorten requirements. Zo zijn system requirements opgesteld voor het hele systeem en bevatten subsystemen die die kunnen bestaan uit software en hardware. Hier moet uiteindelijk alles ervoor zorgen dat deze requirement wordt gehaald. Software requirement zijn niet bedoeld voor het hele systeem, maar behappen alleen de software van het systeem. Software requirement zijn niet bedoeld voor het hele systeem, maar behappen alleen de software van het systeem. De software requirements kunnen gaan over de functionele eisen, gebruikers eisen en zakelijke vereisten. Requirements zijn onder te verdelen in verschillende delen:

- Functional Requirement
- Performance requirement
- Usability requirement
- User requirement
- Interface requirement
- Modes requirement
- Adaptability requirement
- Physical requirement
- Design requirement
- Environmental requirement
- Logistical requirement

Onder deze verschillende requirements zijn er nog twee soorten, functionele en niet-functionele requirements. Functionele requirements geven aan wat het systeem moet doen en kunnen. Niet-functionele requirements geven de eigenschappen aan van het systeem, zoals snelheid, veiligheid en bruikbaarheid. Met andere woorden functionele requirements geven informatie over het "wat". niet-functionele requirements geven informatie over het "hoe".

2.1.1 Mode confusion?

De naam van het begrip zegt het eigenlijk allemaal. Bij mode confusion maakt de gebruiker een vergissing in de huidige of geactiveerde modus van het systeem. De gebruiker denkt dat het systeem in modus A staat terwijl het werkelijk in modus B staat.

2.1.2 Automatisering paradox

Automatiseringsparadox. Wanneer een systeem is dat volledig geautomatiseerd moet worden is er altijd een wel een stap die dat nog niet is. Wanneer er een stap is geautomatiseerd moet er weer iets anders geautomatiseerd worden, om het hele systeem automatisch te krijgen.

2.2 specificaties

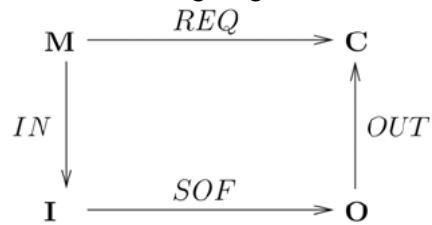
Specificaties zijn eigenlijk niet heel veel anders dan requirements. ze beschrijven beide een systeem of een deelsysteem. Het grote verschil tussen de twee is de grenzen die ze opleggen. Bij requirements is er ruimte voor interpretatie, bij specificaties is die ruimte voor interpretatie er niet.

De specificaties geven geen ruimte voor interpretatie, omdat ze 'meetbare' informatie bevatten. In specificaties worden meetbare eenheden gebruikt, zoals 10 meter of 10°C. Door dat de eisen een meetbare eenheid bevatten kan hiervan niet worden afgewezen. Deze specificaties zullen dan ook niet veranderen als het wordt gebruikt in een ander land, doordat de eenheden zijn gegeven.

Stel we nemen het eerder genoemde requirement voorbeeld: "Binnen moet het altijd warm zijn." Als we dit vertalen naar een specificatie wordt het: "Binnen moet het altijd minimaal 20°C zijn."

2.3 Het vier variabelen model

Sensoren Software Actuatoren Omgeving



M - Monitored Variable statespace **C** - Controlled Variable statespace
I - Input Variable statespace **O** - Output Variable statespace

[?]

2.3.1 Monitored variabelen

Wat gemeten wordt vanuit de omgeving:

- Temperatuur
- Licht intensiteit
- Luchtvochtigheid
- Wat je allemaal met sensoren kunt meten

2.3.2 Controlled variabelen

Kunnen worden "bestuurd" door actuatoren:

- Temperatuur
- Licht intensiteit
- Wat je allemaal kan beïnvloeden

2.3.3 Input variabelen

De data, die staan voor de gemeten waardes vanuit de omgeving, die als input door de software worden gebruikt.

2.3.4 Output variabelen

De data die de software levert als output. Waar de actuatoren op moeten handelen.

2.4 Rampen

2.4.1 Ramp 1

Beschrijving

Datum en plaats

Oorzaak

2.4.2 Ramp 2

Beschrijving

Datum en plaats

Oorzaak

2.4.3 Ramp 3

Beschrijving

Datum en plaats

Oorzaak

2.4.4 Ramp 4

2.4.5 Ramp 5

2.4.6 Ramp 6

3 Modellen

Een goed model heeft een duidelijk object dat gemodelleerd moet worden, er is duidelijk **wat** er beschreven moet worden.

Een goed model heeft een duidelijk doel. -waarom modelleren we? (voor communicatie of verificatie, analyse, etc.)

Een goed model is traceerbaar: elk onderdeel is te herleiden tot de onderdelen van het "echte" systeem.

Een goed model is waarheidsgetrouw: relevante onderdelen van het model komen terug in de werkelijkheid.

een goed model is eenvoudig, maar niet te eenvoudig

Een goed model is uitbreidbaar en herbruikbaar: in de toekomst is het eenvoudig verder te werken met dit model en kunnen zelfs *klassen* van vergelijkbare systemen gemaakt worden

Een goed model deelt geen jargon/semantiek met andere documenten en modellen.

Richtlijnen (tegenstrijdig heden:

Waarheidsgetrouw vs simpelheid duidelijkheid vs. gedeeld jargon/semantiek

3.1 De Kripke structuur

3.2 Soorten modellen

3.3 Tijd

3.4 Guards en invarianten

Guards zijn voorwaarden waaraan moet worden voldaan voordat een status kan worden gemaakt.

3.5 Deadlock

3.6 Zeno gedrag

4 Logica

4.1 Propositielogica

4.2 Predicatenlogica

4.3 Kwantoren

4.4 Dualiteiten

5 Computation tree logic

5.1 De computation tree

5.2 Operator: AG

De betekenis van AG is makkelijk te onthouden A = Always, G = Globally. Dit houdt in dat het niet uit maakt waar je bent, je zal altijd van welke positie dan ook bij een gedefinieerd punt uitkomen.

5.3 Operator: EG

De betekenis van AG is makkelijk te onthouden E = Exists, G = Globally.

5.4 Operator: AF

De betekenis van AG is makkelijk te onthouden A = Always, F = Eventually.

- 5.5 Operator: EF
- 5.6 Operator: AX
- 5.7 Operator: EX
- 5.8 Operator: $p \text{ U } q$
- 5.9 Operator: $p \text{ R } q$
- 5.10 Fairness
- 5.11 Liveness