Verslag Tinlab Advanced Algorithms

J. I. Weverink
...

15 april 2021

Inhoudsopgave

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 geplaats. De requirements voor de verwaring 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 van het perspectief van de opdrachtgever. De opdrachtgever kan de requirtements 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 overscheden 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 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 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
- Enviromental 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 denk 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 interpetatie, bij specificaties is die ruimte voor interpetatie er niet.

De specificaties geven geen ruimte voor interpetatie, 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

2.3.1 Monitored variabelen

Monitored variabelen zijn waarnemingen die kunnen worden gemeten vanuit de omgeving. De waarnemingen worden gemeten door sensoren, de gemeten data wordt als input gebruikt. Voorbeelden van zulke waarnemingen zijn:

- Temperatuur
- Licht intensiteit
- Luchtvochtigheid

2.3.2 Controlled variabelen

Controlled variablen zijn waarnemingen die in zekere zin beïnvloed kunnen worden. Zo kunnen de controlled variabelen 'bestuurd' worden, er is controle over.Voorbeelden van controleerbare waarnemingen zijn:

- Temperatuur
- Licht intensiteit

2.3.3 Input variabelen

Input variablen zijn de waardes, of de data, die een sensor doorstuurt naar de software die het systeem bestuurd. Deze data staat voor de waardes die de sensor heeft gemeten vanuit de omgeving. Omdat de sensor de gemeten waardes heeft omgezet in data kan dit worden gebruikt in de software, om bijvoordbeeld berekeningen mee te maken.

2.3.4 Output variabelen

Output variabelen zijn de 'uitkomsten' van de software die worden uitgelezen door de actuatoren. De actuatoren handelen aan de hand van de output van de software.

2.4 Rampen

2.4.1 Therac-25

Beschrijving

Datum en plaats

Oorzaak

3 Modellen

Een model is een schematische weergaven van de werking van een systeem. Elk soort systeem kan gemodelleerd worden. of het ingewikkeld of heel simpel is. Een model wordt altijd zo realistisch mogelijk gemaakt. Dat wil zeggen dat de belangrijke onderdelen in het model ook terug komen het "echte" systeem.

3.1 De Kripke structuur

Een kripke structuur bestaat uit binaire overgangsrelaties tussen statussen, die de werking van een systeem weergeven.[3] Aan deze overgangen en handelingen kunnen eigenschappen gekoppeld worden. Op deze manier kunnen de handelingen en eigenschappen gecontroleerd en verifieerd worden en het systeem getest worden of het op juiste wieze handelt. De acties binnen een kripke structuur lopen altijd synchroon.

3.2 Soorten modellen

Andere soorten modellen zijn discrete and continuous -time Markov chains, deze zogehete "Marko chains" zijn een vorm van stochastic model checking. Bovendien kosten continuous chains enorm veel kracht om te berekenen. Zoals A. Philippe en C. Robert het verwoorden in hun boek *Linking Discrete and Continuous Chains*[5] hebben deze vormen een gebrek aan intuïtieve basis achter de theorie van Markov chains.

3.3 Tijd

Tijd kan een belangrijke rol spelen in het modelleren van een systeem. De tijd die jij en ik bijhouden op een 24-uurs klok die is net zo van invloed op een systeem. De handellingen gebeuren immers in de loop van de tijd. In UPPAAL kan met het verloop van tijd ook gewerkt worden. De tijd verloop is altijd synchroon. Dit houdt in dat er geen 2 of meerdere acties tegelijk uitgevoerd kunnen worden.

3.4 Guards en invarianten

Guards zijn voorwaarden waaraan moet worden voldaan voordat een transitie genomen kan worden. Het is als het ware een ïf statement", is de uitkomst niet true dan kan de transitie niet genomen worden.

Invariant theory is al heel oud en komt oorspronkelijk uit de wiskunde. Over de jaren heen heeft het verschillende betekenissen gekregen. Hermann Weyl legt dit uitgebreid uit in zijn boek *The Classical Groups*. [2] G. Rota haal uit het boek van Weyl 2 beweringen. De eerste äll geometric facts are expressed by the vanishing of invariants". en de tweede äll invariants are invariants of tensors". Echter denk ik dat deze beweringen niet van toepassing zijn in deze context.

In de context van de opdracht zijn invarianten voorwaarden die kunnen worden vastgesteld aan een status. De state moet worden verlaten zodra de invariant niet

meer van kracht is(als de voorwaarde niet meer behaald is) en zal geforceerd worden een beschikbare transitie te nemen.

3.5 Deadlock

Een deadlock ontstaat als er een proces in groep processen een geheugen locatie bezet houden en bij een andere geheugen locatie wilt komen, maar deze wordt al bezet gehouden door een ander proces in de groep. Dit is een deadlock in de ogen van een programmeur.[6]

Een deadlock heeft echter een hele andere definitie als het neerkomt op modelleren. In de wereld van modelleren heeft een deadlock niets te maken met gebruik van geheugen locaties. Een deadlock kan alleen ontstaan als er ristricties zijn gelegd aan transities. Deze restricties zorgen ervoor dat niet elke transitie altijd genomen kan worden. Als er een status is waar geen volgende transitie meer mogelijk is preken we van een deadlock.

3.6 Zeno gedrag

Zeno gedrag is een wiskundige term. het beschrijft een situatie waarin in een bepaalde tijd oneindig veel transities ontstaan.[7] Doordat er oneindig veel stappen worden gezet in een korte (of lange) tijd, moeten de stappen extreem klein zijn. Dat moet wel want er is immers geen einde aan te krijgen.

4 Logica

4.1 Propositielogica

De propositielogica is een tak van logica die zich bezighoudt met het redeneren met proposities. Proposities zijn uitspraken of beweringen die ofwel waar, ofwel onwaar zijn.

4.2 Predicatenlogica

Predicatenlogica is wiskundig-formele logica waarin expliciet predicaten voorkomen, waarmee eigenschappen van en relaties tussen verzamelingen objecten worden beschreven. Vaak wordt vooral de eerste-orde-predicatenlogica bedoeld

4.3 Kwantoren

Een kwantor (soms wordt ook quantor gebruikt) is een taalelement in de wiskunde, in het bijzonder in de logica. Kwantoren binden variabelen.

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 AF is makkelijk te onthouden A = Always, F = Eventually.

5.5 Operator: EF

De betekenis van AF is makkelijk te onthouden E = Exists, F = Eventually.

********** EXAMPLES **********

Referenties

[1] Knuth: Use of Tabular Expressions for Refinement Automation, https://www.researchgate.net/figure/4-Variable-Model-of-Parnas-Madey_fig3_270733268

[2] WEYL, H (1966). *The Classical Groups: Their Invariants and Representations.*. PRINCETON, NEW JERSEY: Princeton University Press.

[3] V. Gupta, V Pratt (1993) *Concurrent Kripke Structures*. Stanford University.

[4] Kwiatkowska M., Norman G., Parker D. (2007) *Stochastic Model Checking*. Bernardo M., Hillston J. Springer, Berlin, Heidelberg.

[5] Philippe A., Robert C.P. (1998) *Linking Discrete and Continuous Chains*. Robert, Christian P., Springer, New York, NY.

[6] S. Isloor, T. Anthony Marsland (1980) *The Deadlock Problem: An Overview*.

[7] A. D. Ames, A. Abate and S. Sastry (2005) *Sufficient Conditions for the Existence of Zeno Behavior*. University of Californioa, Berkeley.

[8] Michel Goossens, Frank Mittelbach, and Alexander Samarin. *The LaTeX Companion*. Addison-Wesley, Reading, Massachusetts, 1993.

[9] Rota G. -C. (2001). *Algebraic Combinatorics and Computer Science: A Tribute to Gian-Carlo Rota*. Crapo, H. and Senato, D., Springer Milan.

[10] Albert Einstein. *Zur Elektrodynamik bewegter Körper*. (German) [*On the electrodynamics of moving bodies*]. Annalen der Physik, 322(10):891–921, 1905.

[11] Knuth: Computers and Typesetting, http://www-cs-faculty.stanford.edu/~uno/abcde.html