# Testplan ATP
# Lemonator

Philippe Zwietering
Studentnummer: 1685431

8 juli 2019

# Inleiding

In dit testplan worden de tests uiteengezet die zullen worden uitgevoerd voor het voltooien van het Lemonator practicum. Hierbij zal vooral hoofdstuk 5 van de reader worden gebruikt, waarin de testcyclus voor technische, incomplete systemen is beschreven. Hoofdstuk 6 van de reader wordt gebruikt om de eisen en hardware-eigenschappen van de Lemonator te achterhalen.

Globaal gezien zijn er 4 teststadia: simulatie, prototyping, pre-production en post-development. Voor deze opdracht zijn de pre-production en post-development stadia niet relevant, omdat er nooit een volwaardig prototype ontwikkeld zal worden; het blijft altijd bij "laboratory quality". Dit heeft als gevolg dat de Lemonator nooit een product zal worden dat door klanten gebruikt kan worden. Het blijft altijd bij een proof of concept, dat gebruikt kan worden om het idee van een limonademengmachine te illustreren en om fundamentele tests mee uit te voeren.

Het simulatiestadium kan vrijwel geheel getest worden volgens de methoden zoals beschreven in de reader. Dit betekent dat er eerst een model moet worden ontwikkeld voor de omgeving van de Lemonator, die getest moet worden door middel van een one-way simulation. Daarna kan er een model worden opgezet voor de Lemonator zelf, die dan met het model van de omgeving kan wisselwerken. Dit kan vervolgens getest worden met behulp van een feedback simulation. Rapid prototyping kan niet toegepast worden in deze opdracht, omdat het niet mogelijk is de sensoren en actuatoren direct aan te sluiten op een pc om op die manier het model voor de Lemonator te testen.

Het prototypingstadium kan niet compleet volgens de omschrijving worden gedaan zoals in de reader staat beschreven. De hardware van de Lemonator staat namelijk al vast en is vrij experimenteel van aard. Ook is het niet mogelijk om bepaalde delen van de Lemonator apart van elkaar te beschouwen: het is al een compleet product, waarvan de verschillende onderdelen aan elkaar vastzitten en er dus geen garantie kan worden gegeven dat er geen onderlinge afhankelijkheid aanwezig is tussen de verschillende hardware componenten. Dit is natuurlijk omdat het een studieopdracht is en er ook andere mensen zijn die gebruik moeten kunnen maken van de hardware. Er zal dus om deze tekortkoming heen moeten gewerkt. Dit betekent dat het uitvoeren van environmental tests en system integration tests dus onmogelijk is. Environmental testing kan niet omdat er niet de juiste apparatuur voor beschikbaar is[1] en het risico niet gelopen mag worden dat onderdelen van de Lemonator beschadigd raken. System integration tests houdt in dat er op een prototype getest wordt met definitieve power supplies en I/O poorten, wat bij de Lemonator (hoop ik) niet het geval is. De tests die hier wel voor kunnen worden uitgevoerd zijn de software unit en software integration tests door middel van host testing en hardware/software integration testing.

# Testomschrijvingen

Zoals al besproken in de inleiding, zullen er verschillende soorten tests uitgevoerd gaan worden om de kwaliteit van de Lemonator in zijn huidige vorm te garanderen (voor zover dat mogelijk is). Het ontwikkeltraject zal er dan als volgt uitzien:

- Ontwikkelen Python model voor de Lemonator omgeving.
- Opstellen en uitvoeren van software unit tests voor de one-way simulatie van de Lemonator-omgeving.

[1] Er wordt in de reader zelfs gesproken over gespecialiseerde bedrijven voor dit soort tests. Het komt over als iets dat vergelijkbaar is met stress testing.

- Ontwikkelen van Python model en controller voor de Lemonator.
- Opstellen en uitvoeren van blackbox software integration tests voor de feedback simulatie van de Python controller voor de Lemonator.
- Ontwikkelen C++ controller voor Python omgeving van de Lemonator met behulp van de Python API[2].
- Uitvoeren blackbox systems tests voor de feedback simulatie van de C++ controller in de Python Lemonator-omgeving. Hiervoor kunnen als het goed is dezelfde tests gebruikt worden als bij de blackbox system tests voor de Python controller, want ze moeten natuurlijk hetzelfde gedrag vertonen.
- Volledig implementeren embedded C++ controller voor de Lemonator.
- Opstellen en uitvoeren van blackbox system acceptance tests voor de C++ controller op de Lemonator.

Het zou prettig zijn om ook nog ergens whitebox tests te kunnen doen op de hardware, omdat er wat dingen niet helemaal lekker werken op de Lemonator. Dat is helaas niet mogelijk, omdat er niet zomaar extra dingen op aangesloten kunnen worden om te meten wat voor signalen er worden doorgegeven.

Concreet betekent het ontwikkelen van het Python model van de Lemonator-omgeving dat er drivers en stubs gemaakt worden voor alle omgevingsvariabelen die relevant zijn voor de Lemonator. Hierbij kan gebruik worden gemaakt van de simulator die al beschikbaar is gesteld via Github. Deze moeten vervolgens getest worden op "logisch"gedrag, dus dat houdt in:

- De achtergrondtemperatuur. Die is belangrijk om dingen met het warmte-element te kunnen doen, omdat de vaten met vloeistof altijd dezelfde temperatuur zullen hebben als de achtergrondtemperatuur, maar misschien is het verstandiger om dit bij de vloeistofvaten in te programmeren in plaats van als een aparte variabele.
- De beker. Deze moet een bepaalde maximale inhoud hebben en moet weggehaald, geleegd en gevuld kunnen worden. Ook zal de beker een bepaalde warmtecapaciteit moeten hebben, de mate waarin iets opwarmt bij de hoeveelheid warmte-energie die erin gestopt wordt.
- De vloeistofvaten. Deze hebben ook een bepaalde inhoud en zullen leeg moeten gaan als er vloeistof uitgepompt wordt en gevuld moeten kunnen worden.

Deze unittests gaan vervolgens opgesteld en uitgevoerd worden met behulp van de `unittest` en `nose2` modules die geïnstalleerd kunnen worden met behulp van `pip`.

Daarna kan de Python controller voor de Lemonator worden ontwikkeld. Hierbij moet rekening worden gehouden met de functionele en niet-functionele eisen zoals die opgesteld zijn in de reader. Ook moeten er extra features worden toegevoegd om de opdracht te voltooien. Voor een beschrijving van de Lemonator, zie hoofdstuk 6 van de reader. Hierna worden blackbox integration tests uitgevoerd om te controleren of aan deze eisen is voldaan. Dit zal gebeuren door alle functionaliteit van de Lemonator langs te gaan met de verschillende klassen van input om zoveel mogelijk code te kunnen checken. Dus bijvoorbeeld als getest moet worden of de simulatie het pompen goed uitvoert, moet onderzocht worden wat voor gedrag wordt vertoond bij een volle tank, een lege tank en iets er tussenin, en bij een volle beker, een lege beker en iets ertussenin.

[2] `https://docs.python.org/3/extending/extending.html`

Vervolgens wordt in C++ de controller geschreven, die met behulp van de Python API als het goed is zou moeten kunnen communiceren met de Lemonator omgeving. Dit betekent dat dezelfde tests zouden moeten kunnen worden gebruikt als bij de blackbox integration tests voor de Python controller. Zodra al deze tests positief zijn, kan dit worden omgezet naar een embedded controller voor de Lemonator en kan de volledige Lemonator hardware door middel van acceptance testing bekeken worden.