Software Testing & Verifcation

Project stage 1

Jarno Le Conté, 3725154
Bas Meesters, 3700569

Dit rapport beschrijft de eerste stappen die wij hebben doorlopen bij het ontwikkelen van een adresboek-applicatie. Hierbij zullen we de requirements van de software nader bespreken en beschrijven we onze werkwijze omtrent testen. We zullen zowel ingaan op het plan voor unit-testing als de uitvoering daarvan, ook het plan voor system-testing in fase 2 komt aan de orde.

Requirements:

1. Het is in de applicatie mogelijk om meerdere adresboeken aan te maken, te bewerken en in te zien. In dit adresboek staan nul of meerdere contacten waarbij bij elk contact een naam staat en een emailadres. Een adresboek heeft een naam en een lijst met contacten die bij het aanmaken nog leeg is. Er kan één adresboek tegelijk worden weergegeven.
2. Het is mogelijk om per adresboek de contacten te weergeven. Het is ook mogelijk om alle contacten die gezamenlijk in alle adresboeken staan te weergeven.
3. Het is mogelijk om in het adresboek dat op dit moment wordt weergegeven contacten te zoeken. De contacten die aan de zoekopdracht voldoen zullen op het beeldscherm worden weergegeven. Hierna heeft de gebruiker de mogelijkheid deze contacten te verwijderen of aan een ander adresboek toe te voegen. Het speciale adresboek met 'alle contacten' wordt apart bijgewerkt zodra een contact wordt toegevoegd of verwijderd.
4. Het is mogelijk om een contact toe te voegen aan het adresboek dat op dat moment wordt weergegeven.
5. De applicatie valideert ingevoerde contacten op naam en e-mailadres. Daarbij zullen alle regels opgevolgd worden zoals die in de opdracht beschreven staan.

Overzicht unit-testing:

Aan de hand van de requirements hebben we een applicatie gemaakt waarin de data die verwerkt wordt gescheiden is van de user interface. De data die moeten worden verwerkt kan gescheiden worden in contacten, die weer bestaan uit een naam en een emailadres, en adresboeken, die weer bestaan uit een lijst van contacten.

Het unit testen kan dus gedaan worden op de onderdelen contacten, adresboeken en de user interface. Deze worden hier kort per onderdeel besproken.

Wanneer een contact wordt aangemaakt moeten twee strings worden meegeven, één voor de naam en één voor het emailadres. Deze strings worden vervolgens gevalideerd volgens de opgestelde regels. Vervolgens wordt het contact aangemaakt of wordt een foutmelding teruggegeven. Het unit testen van dit onderdeel bestaat uit het veelvoudig aanmaken van nieuwe contacten met daarbij verschillende strings die alle regels voor het valideren van de naam en emailadres langslopen. Daarnaast zullen ook ongeldige strings worden toegevoegd waarbij getest wordt of die geweigerd worden door het systeem. Ondanks dat het valideren wordt uitgevoerd door meerdere functies kiezen wij ervoor om unit-testing alleen uit te voeren op de overkoepelende validatiefunctie omdat bij het kiezen van goede test-strings elke validatiefunctie aangeroepen zal worden. Uit de resultaten kunnen we vervolgens opmaken dat de validatiefuncties correct (samen)werken.

Adresboeken zullen wanneer ze aangemaakt worden een string verwachten. Dit is de naam van het adresboek en zal alleen gevalideerd worden op de lengte, welke korter moet zijn dan 20 tekens. Verder bevat een adresboek een lijst met contacten welke leeg zal zijn bij het aanmaken van een nieuw adresboek. Het is vervolgens mogelijk om aan de lijst contacten toe te voegen en weer te verwijderen. Ook is het mogelijk om naar contacten te zoeken in de lijst. Het unit testen van deze klasse bestaat uit veelvoudig aanmaken van adresboeken met verschillende toegestane en verboden namen. Vervolgens worden de acties getest met standaard en randgevallen, zoals bijvoorbeeld het verwijderen van contacten in een lege lijst.

Het unit testen van de Viewer, de klasse die verantwoordelijk is voor het weergeven van de user interface, wordt gedaan door de testen op de adresboeken en contacten te combineren en uit te laten voeren in verschillende volgordes waarin acties ook daadwerkelijk uitgevoerd kunnen worden. Er wordt vervolgens vergeleken of de weergave overheen komt met de verwachte weergave. Doordat het aantal uit te voeren acties in verschillende volgordes zo veel verschillende combinaties geeft is het niet mogelijk om alle mogelijke volgordes en randgevallen te testen. Door gebrek aan tijd hebben wij dit niet getest. Wij denken daarnaast ook dat het testen van de user-interface voor een groot deel buiten de scope van unit-testing valt en thuishoort bij system-testing.

Conclusie uit unit testing:

We hebben ons voornamelijk gericht op het unit-testen van de overkoepelende functies omdat daarmee ook alle hulpfuncties indirect worden getest. Het betekend dus ook dat we geen tests hebben uitgevoerd op het niveau van boolean logic, echter doordat wij uitgebreide testwaardes hebben gekozen én de tests slagen, denken wij dat wij unit-testing op adresboeken en contacten op de juiste manier hebben toegepast waarbij vrijwel alle mogelijke fouten zijn gedetecteerd. We verwachten dat daarom de unit-tests compleet genoeg zijn om door te gaan naar system-testing. Zoals eerder vernoemd is het testen van de user interface lastig en zal daarom ook een belangrijk onderdeel van system-testing worden.

Plan voor system testing fase:

(**a)** We zullen rekening houden met Observability en Controlability door het scheiden van ‘state’ en de methodes die werken op de ‘state’. Het programma zal dan observable zijn doordat we de ‘state’ opgevraagd kan worden al dan niet via een functie. Het programma zal ook controllable zijn doordat er publieke methodes zijn die door de test-module aangeroepen kunnen worden en die de ‘state’ zullen veranderen. Ook het gebruik van puur functionele functies zal observability en controlability gemakkelijker maken vanwege het ontbreken van side-effects.

(**b)** Een mogelijkheid zal zijn om een input-space coverage criterion te gebruiken zoals bijvoorbeeld Base Choice Coverage, omdat wij denken dat combinaties van input-waardes het meest cruciaal is voor system-testing. Zo zal bijvoorbeeld het invoeren van contacten verschillende resultaten geven afhankelijk van het actieve adresboek. Ook mag het niet mogelijk zijn dat er dubbele contacten komen en zullen contacten op de juiste wijze verwijderd moeten worden. Het is telkens afhankelijk van de huidige ‘state’ of anders gezegd input-space. Ook het overkoepelende adresboek met daarin alle contacten heeft afwijkend gedrag. Elke combinatie van handelingen is dus van belang.

(c) Om rekening te houden met de beperkte tijd voor de opdracht stellen wij de prioriteit om allereerst te testen met simpele test-cases en nog geen coverage criterion te hanteren. Op die manier kunnen wij snel testen of de integratie tussen de verschillende methodes en klassen correct is. Dat zal ook het moment zijn dat wij de user-interface goed kunnen testen. Daarna zullen we kijken of wij kunnen voldoen aan een coverage criterion zoals bijvoorbeeld Base Choice Coverage.