Samenvatting

In een traditioneel softwareontwikkelingsproces bouwen programmeurs gewoonlijk de volledige applicatie in één keer. Deze applicatie is het eindresultaat van een driedelige procedure die begint met een functionele analyse. Vervolgens wordt de applicatie geïmplementeerd en ten slotte grondig getest. Elk van deze stappen neemt een aanzienlijke hoeveelheid tijd in beslag, waardoor softwareontwikkeling een zeer dure en langdradige aangelegenheid is. Sinds de wereldwijde economische crisis zijn ontwikkelaars echter genoodzaakt om drastisch te besparen op hun uitgaven. De gemakkelijkste manier om dit te bereiken is om zo snel mogelijk een minimale versie van de applicatie uit te brengen met enkel de essentiële functionaliteit en deze vervolgens geleidelijk aan uit te breiden. Dit idee staat bekend als Agile Softwareontwikkeling, waarbij “agile” duidt op flexibiltieit.

Hoewel deze aanpak op korte termijn de financiële risico’s vermindert, doet er zich op de langere termijn een ander probleem voor. Om een frequente ontwikkelingscyclus te kunnen hanteren, is er nood aan een mogelijkheid om tests automatisch uit te voeren, zonder menselijke tussenkomst. Dit is mogelijk met behulp van Continue Integratie, maar dit is geen wondermiddel. Om het testproces volledig te kunnen automatiseren, dienen er voldoende en vooral adequate tests aanwezig te zijn.

Softwareontwikkeling vandaag de dag gebeurt razendsnel. Als gevolg van het tempo waaraan ontwikkelaars applicaties uitbreiden, neemt ook het aantal tests superlineair toe. Immers, elke verandering aan de code van de applicatie leidt tot de toevoeging van ten minste één extra test. Dit heeft een negatief effect op de uitvoeringstijd van de tests, waardoor de voordelen van een korte ontwikkelingscyclus tenietgedaan worden.

Dit probleem van schaalbaarheid kan opgelost worden door gebruik te maken van optimalisatietechnieken. Hiertoe stelt deze masterproef drie technieken voor: Test-Minimalisering, -Selectie en -Prioritering. De eerste twee technieken proberen te voorspellen welke tests zullen slagen en voeren deze redundante tests bijgevolg niet uit. De derde techniek voert wel elke test uit, maar bepaalt een ideale uitvoeringsvolgorde zodanig dat tests met een hoge kans op falen eerder worden uitgevoerd.

Deze masterproef presenteert een implementatie van deze techniek met een nieuw en drie bestaande prioriteringsalgoritmen. Het effect van deze techniek is geëvalueerd op twee bestaande applicaties. De resultaten zijn veelbelovend. De eerste falende test wordt gemiddeld dertig keer sneller gedetecteerd dan zonder deze techniek.