Samenvatting

Het ontwikkelproces van welke software dan ook is heden ten dage uitermate belangrijker dan voorheen, niet alleen in de IT sector maar tevens in het bedrijfsleven en vrijwel alle onderzoeksrichtingen. Met oog op het verbeteren van dit ontwikkelproces wat betreft schakelsnelheid, efficientie, betrouwbaarheid, en automatiseerbaarheid, is er in de afgelopen decennia een werkwijze ontstaan, waarin software stapsgewijs opgeleverd wordt, gebaseerd op de competenties van de ontwikkelaar en terugkoppeling van de eindgebruikers.

Softwaremodelering en modeleringstalen hebben als doel het ontwikkelprocess te vergemakkelijken door middel van ontwerpbeschrijvingen van correcte en betrouwbare softwaretoepassingen. Een invloedrijk voorbeeld van softwaremodelering, wat betreft het verbeteren van de betrouwbaarheid en weerbarstigheid van softwareproducten, is het 'concurrency' model van de Abstract Behavioral Specification (ABS) taal. Deze modeleringstaal heeft als kenmerken de ondersteuning voor asynchrone programma's en coĂ¶peratieve uitvoering van programma's. Door van implementatiedetails, programmacomplexiteit en inhoud van programmabilbiotheken te abstraheren, zijn formele analyse technieken en bijbehorende bewijstechnieken aanzienlijk eenvoudiger toe te passen op software modellen dan op software an sich, en dit geldt specifiek ook voor ABS. Echter blijft er hierdoor afstand tussen de gebruikte software modellen en de daadwerkelijke software: het ontwikkelproces bewandelt twee losse paden, de een en ander overeenkomstig aan softwaremodelering in een modeleringstaal en implementatie in een programmeertaal. Dit leidt tot foutgevoeligheid en dubbel werk.

Het uiteindelijke doel van het onderzoek zoals beschreven in dit proefschrift is het overbruggen van de afstand tussen softwaremodellen en software zoals uitgedrukt in programmeertalen. Dit biedt een soepele integratie van formele methoden en twee van de meest bekende en gebruikte programmeertalen voor softwareontwikkeling: Java en Scala. Het onderzoek richt zich voornamelijk op sequentiele en parallelizeerbare applicaties, maar een apart onderdeel richt zich ook op een theoretische beschouwing van gedistribueerde applicaties. Dit onderzoek is een eerste stap naar een programmeertaal met grondige ondersteunig voor formele methoden. Het gepresenteerde werk is verdeeld in drie delen, die elk een onderdeel vormen van het gemeenschappelijke doel.

Het eerste deel beschrijft de ontwikkeling van een programmaomgeving, dat het voorgestelde gestrengelde gelijktijdigheidsmodel van ABS implementeert, doelgericht op de efficiĂ«ntie en schaalbaarheid ervan. Deze omgeving benut optimaal het beheer van de zogenaamde parallellopende leidraden ('threads' in het Engels) in de Java programmeertaal en integreert zowel acteursobjecten als toekomstobjecten met het asynchrone programmeren.

Het tweede deel van dit onderzoek stelt de constructen van hoger niveau, in het bijzonder de constructen die asynchroniciteit en parallellisme modelleren in ABS, beschikbaar binnen de programmeeromgeving van de Java programmeertaal door middel van een applicatieprogrammeringsinterface (API). De basis van deze API, genaamd JAAC, is gegeven als een Java programmaonderdeel en stelt daarmee constructen bloot aan vrij gebruik, veelal niet beperkt door types te checken van asynchrone aanroepen, en is daardoor vatbaar voor de mogelijkheid ongewenste programmatuur uit te voeren binnen acteursobjecten. Deze API is vervolgens uitgebreid, genaamd ASCOOP, als Scala programmaonderdeel, zodanig dat alle (a)synchrone

methode-aanroepen nu wel bij het type checken door de Scala compilator gecontroleerd worden, met als resultaat dat alleen de toegestane programmatuur kan worden uitgevoerd binnen acteursobjecten.

Het derde deel richt zich op de ontwikkeling van een compilator van softwaremodellen met als doeltaal Java, en als uitgebreide doeltaal Scala, dat een formeel correcte vertaling van programmagedrag vertoont. Deze vertaalslag ondersteunt volledig de semantiek van de kern van de modelleringstaal, inclusief acteursobjecten en groepen van acteursobjecten als softwarecomponenten. De ondersteuning van de compilator includeert de constructies voor asynchrone communicatie, het suspenderen van co-routines, en de continuering daarvan. Ten slotte vertaalt de compilator taal extensies van ABS voor tijdsafhankelijke modellen en modellen met consumptiemetingen van (geheugen)bronnen.