Combining Monitoring with Run-time Assertion Checking

Gouw, C.P.T. de

Samenvatting

Fouten in software veroorzaken jaarlijks een schade van 312 miljard dollar volgens een recent onderzoek aan Cambridge University. Type Checking, Static Verification en Run-time Checking zijn bekende manieren om softwarefouten te voorkomen, te isoleren, en op te lossen. Alle drie methoden bepalen of de software naar verwachting werkt op basis van annotaties: een formele beschrijving van de door de gebruiker gewilde werking van de software.

In dit proefschrift hebben we een nieuwe techniek voor Run-Time Checking voor twee object-georiënteerde talen ontwikkeld: Java en de ABS. De ABS is een taal die ontwikkeld is in het Europese HATS project en concurrency ondersteund in de vorm van groepen van objecten, waarbij meerdere groepen tegelijkertijd actief kunnen zijn. In object-georiënteerde talen communiceren objecten door elkaar berichten te sturen. Het gedrag van objecten is volledig bepaald door de volgorde en inhoud van deze berichten. Traditionele methoden voor Runtime Checking focussen *ofwel* exclusief op het beschrijven en testen van deze volgordes (Monitoring), *ofwel* op de beschrijving en het testen van de de gegevens in de berichten (Run-time Assertion Checking, Design by Contract). De methode geïntroduceerd in dit proefschrift **combineert** Monitoring met Runtime Assertion Checking.

Het basisidee van onze techniek is dat het gedrag van objecten formeel beschreven kan worden door een attribuutgrammatica uitgebreid met asserties. De onderliggende (contextvrije) grammatica specificeert de toegestane volgordes van de berichten, de attributen definiëren eigenschappen van de inhoud van de berichten, en de asserties beschrijven de toeges-

tane waarden van deze eigenschappen. Als de asserties geen kwantoren bevatten dan is het beslisbaar of een executie van een programma voldoet aan de specificatie gegeven door zo een attribuutgrammatica. Wij hebben een nieuwe Run-time Checker voor attribuutgrammatica's ontwikkeld in de vorm van een meta-programma in de taal Rascal. Vervolgens hebben we de Run-time Checker toegepast op een industriële case van het bedrijf Fredhopper. Op basis van deze case study hebben we de efficiëntie van de Run-time Checker onderzocht en met succes een aantal fouten in de Fredhopper software ontdekt en opgelost.