Samenvatting

Bij gebruik van Must voor simulaties worden in verschillende toepassingen vaak vergelijkbare objecten gebruikt. Dergelijke objecten lijken zich dan ook te lenen voor generalisatie. Wanneer deze standaard-objecten in units worden opgenomen kan dit dubbel werk besparen en bovendien zorgen voor een eenvoudige structuur van het hoofdprogramma. Object-georiënteerd programmeren is hierbij een goede methode: dit maakt het mogelijk in een toepassing te werken met afstammelingen van de gegeneraliseerde objecten, waarbij de afstammelingen dan precies kunnen worden aangepast aan de specifieke eisen van deze toepassing.

Uitwerking van een voorbeeld waarin een kassa en een meervoudig kassa-systeem zijn gegeneraliseerd gaf het gewenste resultaat: deze blijken zich goed te lenen voor generalisatie; definities van deze objecten zijn in units opgenomen. Bij initiatie van deze objecten kunnen enkele parameters worden meegegeven, zoals bijvoorbeeld rijdisciplines en aantallen wachtrijen. De objecten zijn dan ook in diverse toepassingen inzetbaar. Bovendien leidt de generalisatie tot een relatief eenvoudige structuur van het hoofdprogramma. Wanneer ook een klant (waarvan een attribuut voor de kassa toegankelijk moest zijn) op hetzelfde niveau wordt gedefinieerd - d.w.z als een gegeneraliseerde klant binnen dezelfde unit als de gegeneraliseerde kassa - wordt het hoofdprogramma nog eenvoudiger. Het genoemde hoofdprogramma betreft de simulatie van een eenvoudig supermarktmodel: klanten worden aangemaakt door een generator en verdwijnen weer na behandeling door de juiste kassa.

Generalisatie kan dus leiden tot een breed-inzetbaar object en een eenvoudig hoofdprogramma. De eisen breed-inzetbaar en eenvoudig-hoofdprogramma zijn echter vaak strijdig. Een complicatie is namelijk dat een in een unit gedefinieerd object niet kan beschikken over attributen van een op een ander niveau (bijvoorbeeld het hoofdprogramma) gedefinieerd object. Door gebruik te maken van virtuele methoden kan dit opgelost worden. Dit maakt het hoofdprogramma complexer. Het probleem kan omzeild worden door ook dat andere object in diezelfde unit te definiëren. Dit maakt een en ander echter minder flexibel: het geeft slechts een oplossing voor een specifieke toepassing.