Stichting NIOC en de NIOC kennisbank

Stichting NIOC (www.nioc.nl) stelt zich conform zijn statuten tot doel: het realiseren van congressen over informatica onderwijs en voorts al hetgeen met een en ander rechtstreeks of zijdelings verband houdt of daartoe bevorderlijk kan zijn, alles in de ruimste zin des woords.

De stichting NIOC neemt de archivering van de resultaten van de congressen voor zijn rekening. De website www.nioc.nl ontsluit onder "Eerdere congressen" de gearchiveerde websites van eerdere congressen. De vele afzonderlijke congresbijdragen zijn opgenomen in een kennisbank die via dezelfde website onder "NIOC kennisbank" ontsloten wordt.

Op dit moment bevat de NIOC kennisbank alle bijdragen, incl. die van het laatste congres (NIOC2018, gehouden op dinsdag 6 en woensdag 7 maart 2018 jl. en georganiseerd door CVI i.s.m. NHL Stenden Hogeschool). Bij elkaar bijna 1450 bijdragen!

We roepen je op, na het lezen van het document dat door jou is gedownload, de auteur(s) feedback te geven. Dit kan door je te registreren als gebruiker van de NIOC kennisbank. Na registratie krijg je bericht hoe in te loggen op de NIOC kennisbank.

NIOC 2023 wordt gehouden op donderdag 30 maart 2023 in Emmen en wordt georganiseerd door NHL Stenden Hogeschool.

Wil je op de hoogte blijven van de ontwikkeling rond Stichting NIOC en de NIOC kennisbank, schrijf je dan in op de nieuwsbrief via

www.nioc.nl/nioc-kennisbank/aanmelden nieuwsbrief

Reacties over de NIOC kennisbank en de inhoud daarvan kun je richten aan de beheerder:

R. Smedinga kennisbank@nioc.nl.

Vermeld bij reacties jouw naam en telefoonnummer voor nader contact.

Stapsgewijze verfijning expliciet gemaakt

M.R. Kramer, E.W. van Ammers
Vakgroep Informatica, Landbouwuniversiteit Wageningen

Inleiding

In het inleidend programmeeronderwijs staat het idee van 'gestructureerd programmeren' doorgaans centraal (wie zou zijn studenten ook anders willen leren programmeren?). Stapsgewijze verfijning is daarbij een gebruikelijke techniek, die in het onderwijs echter vaak impliciet blijft. Studenten moeten leren 'aanvoelen' hoe de methode werkt, door middel van voorbeelden en (in de praktijk) door schade en schande.
In Wageningen wordt in het onderwijs al zo'n tien jaar gewerkt met een expliciete vorm van stapsgewijze verfijning. Deze methode berust op twee principes: enerzijds worden de verfijningsstappen als zodanig herkenbaar gemaakt in het geproduceerde programma, anderzijds zijn de wisselwerkingen tussen acties geformaliseerd door middel van het begrip 'interface van een actie'.

Expliciete aanduiding van verfijningsstappen

De methode van stapsgewijze verfijning berust erop dat men een probleem kan oplossen door het op te splitsen in een aantal eenvoudiger deelproblemen, waarvoor oplossingen worden gepostuleerd. De oplossing van het oorspronkelijke probleem bestaat uit de deeloplossingen plus besturing om deze delen in een geschikte volgorde uit te voeren (met eventueel keuze en/of herhaling). De deelproblemen worden op dezelfde manier verder verfijnd totdat de deeloplossingen 'elementair' zijn.
Een verfijningsstap bestaat er nu uit dat men een (deel)probleem uitwerkt tot een aantal acties en bijbehorende besturing. Sommige acties kan men direct als 'harde code' opschrijven, andere moeten nog verder worden verfijnd. Deze laatste acties worden als *pseudostatements* in het programma opgenomen. Een pseudostatement vervult de rol van een gepostuleerde oplossing. Daarom beschrijft het niet alleen wat er op die plaats gebeurt, maar tevens welke variabelen daarbij betrokken zijn (zie 3).
De programmeerstijl die in Wageningen wordt onderwezen kent een prominente plaats toe aan de pseudostatements als neerslag van de stapsgewijze verfijning. Deze worden in de vorm van commentaar opgenomen in het programma. Daardoor kan men ook een gedeeltelijk uitgewerkt programma lezen en begrijpen. Bovendien wordt een pseudostatement niet verwijderd bij de verdere verfijning van de desbetreffende actie, maar blijft het staan als uitleg bij de resulterende code. Deze werkwijze wordt in vervolgvakken uitgebreid tot de techniek van 'literate programming', waarbij de verfijningsstappen als zelfstandige onderdelen gedocumenteerd worden.
Tenslotte wordt de hiërarchie van verfijningen expliciet aangegeven door de volgende conventies:

1. ieder pseudostatement wordt voorzien van een nummer overeenkomstig de plaats in de hiërarchie: acties die ontstaan bij de eerste verfijning worden aangeduid met nummers (1), (2), enzovoort; de verfijning van actie (3) levert bijvoorbeeld acties (3.1) tot en met (3.5) op
2. de uitwerking van een actie staat direct onder het corresponderende pseudostatement
3. elke verfijning wordt afgesloten met een speciale regel commentaar, waarin het nummer van het pseudostatement herhaald wordt.

Het interface van een actie

Het succes van stapsgewijze verfijning staat of valt met de mate waarin acties onafhankelijk uitgewerkt kunnen worden. Acties dienen onderling zo min mogelijk koppeling te vertonen. Deze koppeling bestaat eruit dat resultaten van de ene actie als gegevens dienen voor volgende acties. Elke actie levert dus bepaalde gegevens op en gebruikt resultaten van elders. De variabelen die deze gegevens dragen, vormen samen het *interface* van die actie (vergelijk de pinnen van een IC).

onderwijsinhoud informatica
onderwijsuitvoering curriculum, didactiek
onderwijssector hoger onderwijs, scholing en training

Tijdens het verfijningsproces spelen interfaces een belangrijke rol. Het interface en de beschrijving van het deelprobleem bevatten samen alle informatie die nodig is om de actie onafhankelijk uit te kunnen werken. Daarom nemen wij interface-variabelen ook expliciet op in de pseudostatements.
Door het interfaces van elke actie te benoemen, kan men eenvoudig de consistentie van een opsplitsing controleren: iedere variabele die in de uitwerking van een pseudostatement voorkomt moet òf in het interface voorkomen, òf buiten de actie überhaupt geen rol spelen. Bovendien zullen bij een goede verfijning de interfaces van deel-acties klein blijven (de-coupling).
Bij de methode die in Wageningen wordt gehanteerd, worden de interfaces in de pseudostatements opgenomen door, in de lopende tekst, de namen van de desbetreffende variabelen te markeren. Op deze manier is het interface van een actie goed zichtbaar terwijl pseudostatements toch tamelijk informeel blijven. Een bijkomend voordeel is dat zo'n beschrijving vaak zinvolle namen voor variabelen oplevert.

Slot

Op de gepresenteerde *vorm* van stapsgewijze verfijning zijn allerlei variaties mogelijk. De *methode* als zodanig is echter zeer geschikt om programmeurs (in dit geval studenten) bewust te maken van de structuur van hun programma's. De resulterende code is bovendien een eerste stap op weg naar documentatie van de programma-structuur.
Het expliciet maken van verfijningen wordt in het begin soms als overhead ervaren. Dat geldt met name voor kleine programma's (die nog als geheel te overzien zijn). Wanneer men de methode echter eenmaal beheerst, is het ook voor grotere programma's een goede leidraad om tot een zinvolle opsplitsing te komen.