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 (NIOC2023, gehouden op donderdag 30 maart 2023 jl. en georganiseerd door NHL Stenden Hogeschool). Bij elkaar bijna 1500 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.

Het eerstvolgende NIOC vindt plaats op donderdag 27 maart 2025 in Zwolle en wordt dan georganiseerd door Hogeschool Windesheim. Kijk op www.nioc2025.nl voor meer informatie.

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.

De interne opleiding voor software engineers bij Philips

J.J. van Amstel, V.M. Ronteltap, R. Vader
Philips, Natuurkundig Laboratorium
Postbus 80000
5600 JA Eindhoven

Samenvatting

Een opleiding voor software engineers wordt beschreven. Daartoe wordt eerst het begrip software engineering gedefinieerd. In de opleiding komen onderwerpen aan bod uit de informatica en de software engineering. Bovendien is er aandacht voor methoden en technieken.

1 Inleiding

In het begin van de jaren tachtig werd door twee externe consultants, A. Macro en J. Buxton, voor Philips een interne opleiding voor software-ontwikkelaars gedefinieerd. De opleiding duurde zeven weken en de nadruk lag op praktische richtlijnen voor software engineers.
In dezelfde tijd bezon een interne commissie zich op de gewenste opleiding voor software-ontwikkelaars. Deze commissie deed een rapport het licht zien met als titel 'A model for software education for software developers within Philips' (Bourgonjon e.a., 1984). In dit rapport werd de nadruk gelegd op de fundamentele, formele kennis die een software-ontwikkelaar nodig heeft. De inleiding van het rapport stelt: "For all but some trivial software systems, the most important characteristic of software is its inherent complexity. Software design has therefore to do with the mastering of this complexity. Formal techniques and abstraction (as applied e.g. in mathematics) are most essential tools."
Op basis van dit rapport werd bij Philips in Hilversum de CSO (Cursus voor Software Ontwerp) gedefinieerd. Het centrale thema in de cursus was het gebruik van formalismen in de Informatica. De Software Engineering kwam er in de cursus wat bekaaid vanaf.
In 1985 werd binnen Philips de afdeling OIT (Opleidingen Informatie Technologie) opgericht. De groep SWE (Software Engineering) van het OIT kreeg de opdracht om een curriculum te maken voor software-ontwikkelaars. Bij het opzetten van het curriculum werd dankbaar ge-

bruik gemaakt van de bouwstenen die voorhanden waren. Dit hield in, dat de CSO werd overgenomen van Hilversum en dat als een vervolg de CSE (Cursus voor Software Engineering) werd gedefinieerd, die enige gelijkenis vertoonde met de cursus van Macro en Buxton, maar met als grote verschil dat er nu werd voortgebouwd op de bij cursisten aanwezige kennis van formele methoden en technieken. Door het OIT werden ook enkele cursussen gegeven op het gebied van methoden en technieken. Om ervoor te zorgen dat de cursusgevers en de cursussen zelf zouden blijven aansluiten bij de praktijk van de software-ontwikkeling, werd in 1988 besloten om de software engineeringgroep onder te brengen bij het in 1987 opgerichte Centrum voor Software Technologie (sedert 1991 opgenomen in het Natuurkundig Laboratorium). Daardoor werd het pakket praktische cursussen ook aanzienlijk uitgebreid. Dit pakket duiden we hier verder aan met CMT (Cursussen op het gebied van Methoden en Technieken). Het curriculum heeft daardoor nu de globale structuur zoals in figuur 1.

Figuur 1
Globale opzet van het curriculum

Voordat we in 3 ingaan op de diverse cursussen, zullen we in 2 het begrip software engineering beschrijven, zoals dat gebruikt wordt in het curriculum.

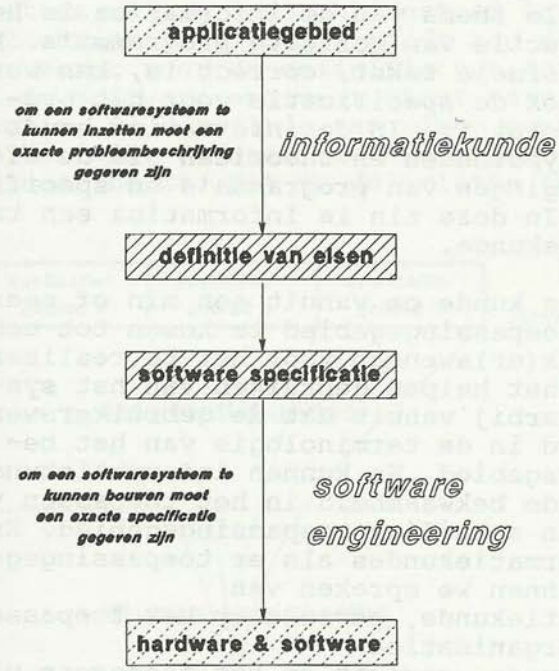
2 Software Engineering

Een korte en bondige definitie van 'software engineering' als kunde zou kunnen luiden: het effectief definiëren, ontwerpen en realiseren van softwareprodukten. We willen wat langer stilstaan bij dit vakgebied, vooral ook om de relaties te schetsen met begrippen als informatiekunde en informatica. In het geheel van activiteiten behorende bij de ontwikkeling van een softwareprodukt kunnen we een tweedeling maken:
activiteiten die te maken hebben met

- het doel en het gebruik
- de realisatie en de constructie

van het softwareprodukt.

Een dergelijke tweedeling komen we tegen bij de realisatie van allerlei produkten. Denk maar aan de realisatie van een huis, waarbij de architect de gebruikerswensen vertaalt in ruimten en de aannemer zorgt voor de fysieke realisatie. Voor een softwareprodukt kunnen we de tweedeling in beeld brengen als in figuur 2.



Figuur 2
Software Engineering en Informatiekunde

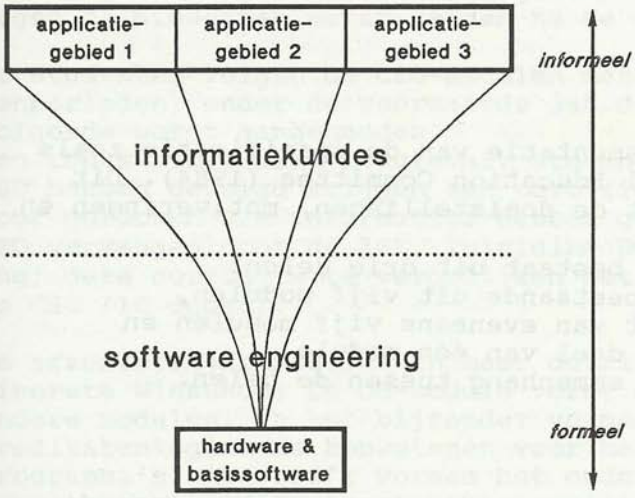
Het gaat om twee clusters van activiteiten van verschillende aard. In de kennis en kunde, die nodig zijn voor deze activiteiten, komen -naast overeenkomsten- ook grote verschillen voor. De kunde die te maken heeft met het doel en het gebruik van het software-produkt noemen we Informatiekunde. De kunde die te maken heeft met de realisatie van het softwareprodukt noemen we Software Engineering. Om de informatiekunde en de software engineering te beschrijven gaan we uit van de Informatica. Informatica is het kennisgebied, de wetenschap, die ten grondslag ligt aan het ontwerpen, bouwen en in bedrijf houden van informatiesystemen.
Zo gedefinieerd is de informatica een zeer breed kennisgebied, omdat in informatiesystemen niet alleen hardware en software een rol kunnen spelen, maar bij-

voorbeeld ook organisatorische en menselijke aspecten. Als we ons wat beperkter opstellen, kunnen we informatica omschrijven als de wetenschap die zich bezighoudt met het beschrijven en ontwerpen van in (hardware en) software uitgevoerde gegevensverwerkende systemen en met het ontwikkelen van theorie, methoden en technieken voor het construeren van dergelijke systemen.
Kruseman Aretz (1985) zegt dan ook dat de informatica zich bezighoudt met het bestuderen van het ontwerpproces van software en als zodanig een technische wetenschap is. Het centrale thema van de informatica is het ontwerp en de constructie van correcte programma's. Dat een programma, een formele tekst, correct is, kan worden aangetoond als ook de specificatie voor het programma een formele tekst is. In de informatica houdt men zich bezig met hypothesen en theorieën die de eigenschappen en gedragingen van programma's en specificaties beschrijven. In deze zin is informatica een tak van de toegepaste wiskunde.

Informatiekunde is de kunde om vanuit een min of meer vage vraag uit een toepassingsgebied te komen tot een definitie van gebruik(er)swensen voor het te realiseren softwaresysteem (en het helpen gebruiken van het systeem). We gaan er daarbij vanuit dat de gebruikerswensen zijn geformuleerd in de terminologie van het betreffende toepassingsgebied. We kunnen informatiekunde ook beschrijven als de bekwaamheid in het toepassen van de informatica in een specifiek toepassingsgebied. Er zijn dus zoveel informatiekundes als er toepassingsgebieden zijn en zo kunnen we spreken van
bestuurlijke informatiekunde, gericht op het toepassen van informatica in organisaties
medische informatiekunde, gericht op het toepassen van informatica in de medische wetenschappen, enzovoort.

Software engineering is de bekwaamheid in het realiseren van softwareprodukten, ervan uitgaande dat de definitie van de gebruikerswensen voorhanden is. De software engineer zorgt voor de specificatie, het ontwerp en de implementatie van grotere programma's (die ten nutte zijn van anderen). Bovendien onderhoudt de software engineer deze programma's, daar ze aangepast moeten worden aan veranderingen in de eisen. We zouden het drietal wiskunde - informatica - software engineering kunnen vergelijken met het drietal wiskunde - natuurkunde - elektrotechniek.
Informatica is het kennisgebied dat ten grondslag ligt aan de software engineering (zoals de natuurkunde ten grondslag ligt aan de elektrotechniek). Het werkmateriaal van de informatica bestaat uit programma's die kunnen worden afgeleid uit hun (formele) specificaties. De software engineer houdt zich bezig met de construc-

tie en het onderhoud van evoluerende programma's in een zich wijzigende omgeving. De produkten van de software engineer dienen echter opgebouwd te zijn uit programma's in de zin van de informatica. De indruk bestaat dat er een afstemmingsprobleem bestaat tussen de informaticaopleidingen en de praktijk. Universiteiten leiden te veel informatici op die in de praktijk als software engineer moeten functioneren en er is een verschil tussen die twee, zoals er ook een verschil is tussen een natuurkundige en een elektrotechnicus. Op de HIO's wordt te weinig aandacht besteed aan formele methoden en technieken. Figuur 2 geeft aan dat de informatiekundige zorgt voor de definitie van de gebruikerswensen en dat de software engineer start vanuit de formele specificatie. De definitie van eisen beschrijft wat de gebruiker wil, de specificatie beschrijft wat de software engineer bouwt. Wat is de relatie tussen die twee?



Figuur 3
Relatie tussen informatiekunde en software engineering

We gaan ervan uit dat de specificatie het formele model is van de definitie van eisen. De verantwoordelijkheid voor de formele specificatie ligt bij de informatiekundige en de software engineer samen. Zij zullen er samen voor moeten zorgen dat de specificatie een juiste afspiegeling vormt van de gebruikerswensen. Daarbij moeten we ons wel realiseren dat de specificatie een formeel model is. Bij de overgang van de definitie van eisen naar de formele specificatie vindt er bij de meeste toepassingen een overgang plaats van de informele naar de formele wereld. Daarbij gaat altijd informatie verloren. Informatiekundige en software engineer moeten het eens zijn over het model. Om ervoor te zor-

gen dat de informatiekundige en de software engineer bij de realisatie van het formele model elkaar begrijpen, zullen zij iets van elkaars vakgebied moeten kennen. De opleidingen voor beide kundigheden zouden er dan ook uit kunnen zien als

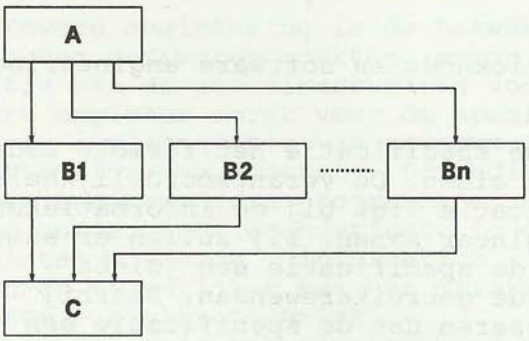
informatiekunde	65% toepassingsgebied
	35% informatica
software engineering	75% informatica
	25% toepassingsgebied en informatietechniek)

Kennis van het toepassingsgebied is voor de software engineer niet essentieel; deze kennis kan in een aantal gevallen natuurlijk wel goed van pas komen. Figuur 3 illustreert de samenhang tussen de beide disciplines. De informatiekunde richt zich grotendeels op de toepassingsgebieden, de software engineering richt zich grotendeels op software en hardware.

3 De cursussen.

3.1 De CSO

De CSO is de implementatie van de specificatie zoals verwoord in Pascal Education Committee (1984). Dit rapport beschrijft de doelstellingen, motiveringen en inhoud van de CSO.
Een volledige CSO bestaat uit drie delen:
A. Een kerndeel, bestaande uit vijf modulen,
B. een keuzepakket van eveneens vijf modulen en
C. een afsluitend deel van één module.
Figuur 4 geeft de samenhang tussen de delen.



Figuur 4
De CSO

Het kerndeel bestaat uit de modulen:

DM : Discrete wiskunde (7 dagen)
PP1: Programmeerprincipes 1; algoritmeontwerp (8)
PP2: Programmeerprincipes 2; datastructuurontwerp (8)
PP3: Programmeerprincipes 3; parallel programmeren (5)
PP4: Concepten van programmeertalen (5)
Men moet de modulen in deze volgorde volgen.

Op dit moment is één keuzepakket beschikbaar. Het bestaat uit de modulen:
OS : Operating Systems (5 dagen)
DB : Databases (5)
SA1: Systeemarchitectuur (4)
SA2: Netwerken (3)
CO : Compilers (5)
De volgorde waarin men deze modulen volgt is vrij.

De afsluitende module
PM: programmeermethodologie (10 dagen)
is een inleiding in Software Engineering. Deze module mogen de studenten eerst volgen na de andere modulen.

De studenten volgen de CSO-modulen met zelfgekozen tussenperioden, onder de voorwaarde dat de voorgeschreven volgorde wordt aangehouden.
Een toets sluit elke module af. Studenten die de gehele CSO hebben gevolgd krijgen een certificaat.
Voor personen die informatica hebben gestudeerd is de CSO vervangen door de ISE, Inleiding Software Engineering; deze cursus is te vergelijken met de PM-module van de CSO (10 dagen).

De afzonderlijke modulen in meer detail zijn:
Discrete Wiskunde; De DM-module vormt een basis voor de andere modulen. In het bijzonder vormen propositie- en predikatenlogica de bouwstenen voor het redeneren over programma's. Algebra's vormen het onderliggende concept voor abstracte datastructuren, terwijl formele talen noodzakelijke voorkennis vormen voor de CO-module.
In het algemeen geldt dat de DM-module de abstracte en formele wijze van denken bevordert die zo essentieel is voor het ontwerpen van software.

Programmeerprincipes 1; Sequentieel programmeren, het ontwerpen van algoritmen. De cursisten zien in dat programmeren meer is dan het op intuïtieve wijze aaneenschakelen van een verzameling statements in een favoriete programmeertaal. Ze ontdekken dat programmeren een professioneel vak is waarin systematiek en precisie een grote rol spelen en geen bezigheid waarbij trial-and-error de boventoon voert. Het ontwerpen van algoritmen en het aannemelijk maken van de correctheid ervan gaan hand-in-hand. Een belangrijk aspect is dat de studenten vooral het verschil inzien tussen operati-

onele en statische beschrijvingen van algoritmen. PP1 behandelt geen specifieke programmeertaal, hoewel meestal een Pascal-achtige notatie wordt gehanteerd. Evenmin is er een computer-practicum. De cursisten dienen in staat te zijn hun algoritmen op voorhand aannemelijk correct te ontwerpen.
Het ontwerpen van algoritmen is een aspect in elke CSO-module, dus is PP1 een basis voor de resterende modulen. De belangrijkste onderwerpen zijn de axiomatische semantiek van elementaire constructies en het afleiden van repetities aan de hand van invarianten.

Programmeerprincipes 2; datatypen en -structuren. Het concept abstracte datastructuur vormt het belangrijkste onderwerp in PP2. Niet alleen zullen de cursisten de vaardigheden verwerven om datastructuren te specificeren maar ook om ze te implementeren met behulp van datatypen en -structuren die al beschikbaar zijn, waarbij op het behoud van de correctheid van de operaties een sterke nadruk ligt. De wiskundige theorie van algebra's vormt daarbij een ondersteunend hulpmiddel. Aldus verkrijgen de studenten de eerste beginselen van Object-Oriented ontwerpen en programmeren. Net als voor PP1 geldt, dat PP2 een basis is voor de resterende CSO-modulen.

Programmeerprincipes 3; parallel programmeren. De studenten krijgen de vaardigheid om predikatenlogica toe te passen bij het verifieerbaar ontwerpen van (parallelle) oplossingen voor programmeerproblemen.
De nadruk bij PP3 ligt op vooral communicatie- en synchronisatie-primitiva. De predikatenlogica dient als hulpmiddel om 'safeness' en 'liveness' te garanderen. De onderwerpen die aan bod komen zijn semaforen, monitoren, channels en rendez-vous mechanismen.

Programmeerprincipes 4; concepten van programmeertalen. De nadruk ligt op de toepasbaarheid en bruikbaarheid van taalconcepten en zeker niet op de eigenschappen van verschillende talen in alle detail. PP4 behandelt verschillende manieren waarop basistructuren van PP1,2 en 3 zijn belegd in moderne programmeertalen. Onderwerpen die aan bod komen zijn onder andere: Het bindingsconcept, datatypen, besturingsstructuren in imperatieve programmeertalen en concepten in niet-imperatieve programmeertalen. In tegenstelling tot andere modulen bevat PP4 wel een computer-practicum met een niet-procedurele programmeertaal als LISP, Hope of PROLOG.

Het eerste keuzepakket van vijf modulen. De doelstellingen van de modulen in het keuzepakket liggen alle in de sfeer van 'het kunnen toepassen wat in het kerndeel is geleerd, waarbij tevens de noodzakelijke basiskennis

omtrent het onderwerp wordt verworven'.

Operating Systems; De onderwerpen die aan de orde komen zijn: Classificatie van operating systems, de 'nucleus', memory en Input-Output Management, het filing system, resource allocation en scheduling.

Systeemarchitectuur 1; SA1 bespreekt systeemarchitectuur vanuit het gezichtspunt van de software-ontwerper. Onderwerpen die aan bod komen zijn: De implementatie van elementaire operating system-functies samen met de hardware-software interface van een operating system, de invloed van programmeertalen en operating systems en parallelle systemen.

Systeemarchitectuur 2; datacommunicatie en netwerken. Ook de SA2-module behandelt netwerken en datacommunicatie vanuit het gezichtspunt van de software-ontwerper.

Compilers; De CO-module bespreekt het vertaalproces van enige programmeertaalconstructies. De module maakt gebruik van formele talen en het begrip semantiek uit de PP-modulen. Men moet het ontwerpen van compilers zien als syntax driven program design. In deze zin is de CO-module een behandeling van een aanvullende programmeermethode op de PP-modulen. Deze methode omvat de techniek van JSP (Jackson Structured Programming). Onderwerpen zijn: Lexicale analyse, contextvrije talen en parsing, attributengrammatica's, en codegeneratie.

Databases; DB presenteert een formele beschrijvingsmethode voor specificaties van (en de relatie tussen) waardenverzamelingen in het algemeen. Onderwerpen zijn het relationele model, het ontwerpen van een database, datamanipulatie en Data Base Management Systemen.

De afsluitende module. Programmeermethodologie; Als een verplichte afsluiting van de CSO dient de PM-module. In de laatste module worden kenmerken van programmeren-in-het-groot ingeleid. Een belangrijk deel van de tijd wordt ingenomen door het practicum. Groepjes van ongeveer drie studenten pakken een niet te klein en niet-triviaal programmeerproject aan. Het project doorloopt alle fasen van de life-cycle met een nadruk op persoonlijke planning, design en implementatie. Het project past onderwerpen toe uit veel voorgaande CSO-modulen.

3.2 De cursus voor Software Engineering (CSE)

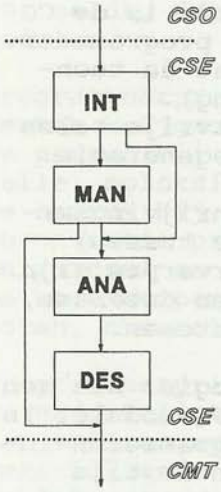
De CSO biedt onvoldoende opleiding voor een volleerde software engineer. De kennis en vaardigheden van de CSO, moeten worden aangevuld met kennis en kunde betreffende programmeren-in-het-groot.

De doelstelling van de CSE is dan ook:

> Na afloop van de CSE, hebben de studenten software engineering als een ingenieursdiscipline aangeleerd, gebaseerd op een managementaanpak (die alle delen van de software ontwikkel cyclus omvat), op traditionele en nieuwe specificatie- en ontwerptechnieken en op software engineering-omgevingen met een nadruk op formele methoden en abstracties.

Een rigoureuze aanpak van software-ontwikkeling is noodzakelijk. Uitgaande van een informeel document met de wensen van de gebruiker voor een project van middelgrote omvang, zijn de cursisten in staat om een projectplan en een formele specificatie voor het produkt te maken. Verder zijn ze in staat een software-produkt te realiseren, uitgaande van een formele specificatie.

De deelnemers in de cursus (12 tot 16 per cursus) zijn ervaren software-ontwikkelaars, die werken in projecten voor de realisering van complexe software.



Figuur 5
De CSE

De cursus duurt zeven weken en is verdeeld over vier blokken: INT, MAN, ANA and DES. Een blok komt overeen met een bepaald onderdeel van het praktisch werk. In elk blok zijn er twee aandachtspunten: lezingen en een praktische opdracht. In het begin van de CSE ligt de nadruk op de lezingen, terwijl naar het einde van de cursus de nadruk komt te liggen op het praktische werk. Een belangrijk gedeelte van de cursus (ca 70%) wordt

besteed aan praktisch werk. De opdracht omvat een programmeerproject, waaraan de cursisten in groepen van circa vier personen werken. Dit is één opdracht voor de gehele CSE.
Een informeel document met gebruikerswensen wordt aan de cursisten ter hand gesteld. De groepen moeten dan een projectplan, een (zo formeel mogelijke) software-specificatie, een software-ontwerp en een implementatie maken. Verder moet een (eenvoudig) introductiedocument voor de eindgebruiker worden gemaakt. De specificatie, het ontwerp en de implementatie zijn object-georiënteerd. Aan het einde van de cursus presenteert elke groep zijn werk aan de 'klant'. Dit omvat ondermeer het uitvoeren van een acceptatietest. Het project moet in de daarvoor gestelde tijd worden gerealiseerd, maar deze tijdlimiet moet de kwaliteit van het werk niet nadelig beïnvloeden; kwaliteit heeft de hoogste prioriteit. Elke groep is verantwoordelijk voor zijn eigen projectorganisatie, keuze van methoden en technieken, zolang dat past binnen de doelstellingen van de cursus. Elke dag moet de groep met de cursusleiding overleggen over de dagelijkse planning.
De cursusleiding speelt in deze opzet diverse rollen. Eén van de begeleiders vervult de rol van klant, maar ook de gebruiker wordt uitgebeeld. Deze klant/gebruiker is beschikbaar (op bepaalde momenten en na afspraak) voor uitleg betreffende de wensen of andere extra informatie. Met de klant moet ook worden overlegd indien stukken van de functionaliteit op verzoek van de groep niet worden gerealiseerd. Verder voert de klant de acceptatietest uit. De cursusleiding heeft dagelijks contact met de groepen om het leerproces te begeleiden. De cursusleiding treedt ook op als adviseur in geval van technische problemen of andere onduidelijkheden in de gepropageerde methode.

De vier blokken zijn in meer detail:
CSE-introductie; Dit blok duurt vier dagen en vereist CSO-kennis als ingangsniveau. In dit blok wordt geen praktisch werk verricht, maar wordt de basis gelegd voor de rest van de cursus. Geprobeerd wordt om een gemeenschappelijk begrippenkader te vormen. In die zin fungeert dit blok als een introductie tot software engineering. De lezingen in dit blok zijn erg verschillend, van documentatietechnieken tot programmeerprincipes, van testtechnieken tot expertsystemen, enz.

CSE-management; De managementaspecten van een project komen aan bod in het tweede blok van de cursus, dat zeven dagen duurt. Het belangrijkste doel van het praktische gedeelte is het produceren van een projectplan, dat gebruikt gaat worden door de groep in de rest van het practicum. De lezingen in dit blok hebben dan ook

voornamelijk betrekking op dit aspect van het werk. Er wordt ingegaan op projectmatig werken, configuratiebeheer, object-oriented werken, kosten van software-ontwikkeling, enz. Aan het eind van dit blok wordt het geproduceerd plan met elk van de groepen besproken en worden suggesties aangedragen voor verbetering. Over het algemeen blijken de cursisten goed in staat een plan te maken waarmee de rest van het practicum kan worden doorlopen. Cursisten die aan dit blok deelnemen moeten het introductieblok hebben gevolgd.

CSE-analyse; In dit blok wordt van de cursisten verwacht dat zij, op basis van de informele beschrijving van de wensen van de klant en het projectplan uit het vorige blok, een (formele, object georiënteerde) analyse maken en een software-specificatie schrijven. Ook het introductiedocument moet in dit blok worden geschreven. De lezingen zijn hierop afgestemd: software-specificatie, informatiemodellering, object-oriented werken, temporele logica, koppeling en cohesie, enz. Het blijk dat dit gedeelte door de cursisten als het moeilijkste wordt ervaren. Met name het modelleren van de wereld van de gebruiker in objecten en de daarvoor te gebruiken notatietechnieken leveren veel problemen op. Dit wordt mede veroorzaakt door het verschil in achtergrond van de cursisten (zowel wat betreft opleiding als werkkring). Ook hier wordt met elk van de groepen de documenten besproken. Dit kan meestal pas plaatsvinden aan het begin van het vierde cursusblok (mede vanwege tijdproblemen van de cursisten). Voor deelname aan dit blok is het niveau vereist van het introductieblok.

CSE-design; In dit laatste blok van de cursus moeten de cursisten een werkend systeem opleveren, gebaseerd op de specificatie uit het vorige blok. Dit houdt in dat men dit blok ook daadwerkelijk moet hebben gevolgd. Het systeem moet worden afgeleverd te zamen met testspecificatie en testrapporten. Verder wordt er door de klant een acceptatietest uitgevoerd, waarna deze het systeem al dan niet accepteert. De lezingen in dit blok hebben betrekking op te kiezen talen, de (on-)mogelijkheden van gangbare talen, te gebruiken abstracties, enz.

3.3 De CMT

De CSO en de CSE zijn consistente en volledige cursussen. CMT bestaat uit een collectie van min of meer onafhankelijke cursussen over methoden en technieken (M_1, M_2, ..., M_k) en over (T_1, T_2, ..., T_n). M-cursussen zijn gebaseerd op (delen van) de CSE; elke T-cursus is gebaseerd op een specifieke M-cursus, op een andere T-cursus of op delen uit de CSE.

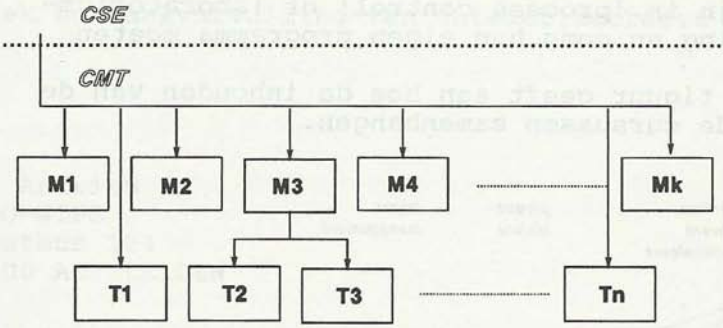


Figure 6
De CMT

Voorbeelden van cursussen zijn:
M: COLD, Object Oriented Design (OOD), Structured Analysis/Structured Design (Pirbhai), InfoMod, Functional Specification & Design (FSD)
T: Pascal, Teamwork-workshop, C-workshop, Modula-2.

Vooropleidingseisen en duur van de cursussen:

	vooropleiding	duur (in dagen)
COLD	CSE-DES	5
OOD	CSE-ANA	5
SASD	CSE-ANA	5
FSD	CSE-ANA	5
Pascal	CSO-PP1/PP2	3
Teamwork	SASD	3
ProMod	SASD	3
C	CSO-PP1/PP2	3
C-workshop	C	3
Modula-2	CSO-PP1/PP2	3
C^{++}-workshop	C	5
SDL-workshop	CSO	5

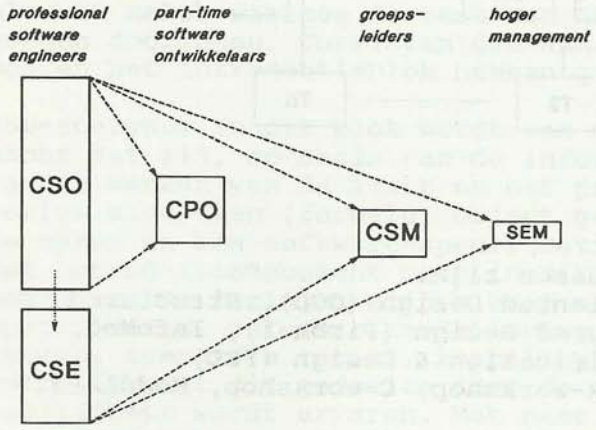
3.4 Andere cursussen

Voor software-projectleiders is er een speciale cursus, de CSM. Deze cursus met een duur van 5 x 3 dagen, geeft een overzicht van de onderwerpen van de CSO en de CSE, met een nadruk op de managementaspecten.

Voor het general management is er een cursus SEM. Deze cursus geeft basisbegrip in de aard van software, het software-ontwikkelproces en de uitwerking van software voor Philips en duurt 4 dagen.

Voor personen voor wie software-ontwikkeling een 'secondary skill' is, is er een cursus CPO van 20 dagen. Deze is bestemd voor personen die bijvoorbeeld

werkzaam zijn in 'process control' of laboratorium-automatisering en soms hun eigen programma moeten schrijven.
De volgende figuur geeft aan hoe de inhouden van de verschillende cursussen samenhangen.



Figuur 7
Andere cursussen

4 Afsluiting

Dit stuk beschrijft een coherent cursuspakket voor software-ontwerpers en software engineers binnen Philips in verschillende vestigingen, te weten Hilversum, Apeldoorn en Hengelo. Het pakket is al enige jaren in gebruik en heeft sinds 1983 tot heden ongeveer 750 CSO-cursisten en ongeveer 200 CSE-cursisten alleen in Eindhoven opgeleverd. Zoals beschreven vindt er een verschuiving plaats van CSO naar CSE. De CSO is tevens in licentie bij Cap Gemini/Pandata-opleidingen.

Gebruikte Literatuur

Pascal Education Committee (1984) A model for software education for software developers within Philips. Philips Eindhoven.

Kruseman Aretz (1985) Aard en wezen van software, Informatie jaargang 27 nr.4, april 1985.