

## 0.1 OO Enquete

dinsdag 12/9/95

### Kenmerken/eigenschappen:

- nadruk op ontwerp
- data: opslag + interface
- nieuwste trend
- overerving
- overloading
- op aanvraag – event-driven
- manier van programmeren die verder gaat dan ADT
- polymorphisme
- verdelen in klassen van objecten met welgedefinieerde operaties
- klassen, objecten en polymorphisme
- van wat?
- classes = types
- ondersteuning van ontwerpen op hoog niveau
- focus verschuiving van algoritmiel naar data
- object = data + operaties
- collecties van objecten in hiërarchische structuren
- het opsplitsen van een probleem in samenhangende subdomeinen
- dingen uit de werkelijkheid worden als object gemodelleerd
- communicerende processen
- objecten van classes zijn representaties van entiteiten
- manier om bibliotheken te ontwikkelen

### Voordelen:

- onderhoudbaarheid (personeelsverloop)
- hergebruik

- information hiding
- abstractie
- aanpassing van bestaande code
- overzichtelijk – hierarchische ordening
- polymorfisme – compilerondersteuning
- uitbreidbaarheid
- geleidelijke overgang tussen ontwerp en implementatie
- van wat?
- modulariteit/data-abstractie
- inheritance bevordert hergebruik
- fouten kunnen lokaal gedebugged worden
- voordelig voor SE
- ontwerpproces: vergroting modulariteit en information hiding
- afscherming bevordert hergebruik
- polymorfisme bevordert hergebruik
- uitbreiding: hergebruik van eigen werk
- vereenvoudigd onderhoud
- De ontwikkelaar van code heeft de mogelijkheid ongemerkt veranderingen in de code door te voeren.
- het is eenvoudig voorgeprogrammeerde code in een eenvoudige vorm aan te bieden
- makkelijk te structureren – overzichtelijk
- betere coördinatie tussen delen
- natuurlijkheid
- implementatie is gescheiden van specificatie

**Nadelen/problemen:**

- nieuw paradigma (moet geleerd worden)
- grote verschillen in ondersteuning (talen)

- cultuuromslag – van lineair naar OO
- bouwen van hiërarchieën kost tijd
- snelheid neemt af
- leren van nieuwe technieken
- code is moeilijk terug te vinden (nadeling voor hergebruik)
- (van wat?) OO is voor mij heel vaag
- overhead dynamic binding
- sluit niet aan bij intuïtie (?)
- soms zijn implementatie details nodig voor performance
- Het grootste nadeel van OOP is dat niemand weet wat het is, en dat degenen die het wel weten dat niet duidelijk kunnen vertellen, en dat degenen die het wel kunnen het met elkaar oneens zijn.
- moeilijk om te schakelen van algoritmisch denken naar denken over het 'gedrag' van objecten.
- (te succesvol): OO toepassen op problemen die niet conceptueel zijn op te delen in objecten is zinloos, maar het gebeurt toch.
- onvermijdelijke overhead
- ontbreken van standaards
- non-intuïtieve werkwijze (als de intuïtie is gevormd door imperatief programmeren)
- er zijn wellicht 'ondeelbare' domeinen/problemen
- nadeel inheritance: In zeer complexe modeleringen kan een wirwar van gerelateerde objecten ontstaan waarbij het basis object niet meer te identificeren is.
- niet altijd even flexibel
- overhead voor programmeur
- moeilijker detectie van run-time errors
- parallel programmeren is moeilijk (voor de mens) te bevatten
- veel dirty programming
- langzaam
- sommige dingen zijn moeilijker / minder efficiënt implementeerbaar

- ieder brokje informatie uit een object vereist een functie (dat maakt de interface omslachtig)
- omscholen van programmeurs van de 'oude school'
- hergebruik is vaak moeilijk in de praktijk
- Hoe vang je de werkelijkheid in objecten?
- exception handling

**Vertel meer over:**

- manieren om OO toe te passen (zonder dat het op de oude manier beter zou kunnen)
- praktijkgericht
- onderwerpen voor de paper
- geschiktheid van C++ voor OO
- hoe organisaties veranderen door het gebruik van OO
- (van wat?) Vertel meer over OOP! Bewijs het nut van OOP!!
- theoretisch achtergronden, wiskundige onderbouwing, correctheidstheorie
- modellering van complexe problemen: technieken of natte vinger werk?
- Vertel meer over C++. (In college C/C++ is te weinig over de OO aspecten van C++ vermeld.)
- onderwerpen die me aanspreken: bv OO modeleren
- eerste het globale idee – principes eerst dan talen
- Is programma correctheid bewijsbaar voor OO?
- parallel programming mbv OOT
- technische aspecten (implementatie van objecten, compilers)
- (vervelende vraag): OO operating systems
- OO multimedia en hypermedia systems
- praktijkvoorbeelden (waargebeurde projecten – waarom ze vastliepen)
- wat er in het boek staat, en daarover dan uitweiden
- Zij er papers uit het verleden te vinden?
- Hoe ontwerp je een goede klassenstructuur?

- programmeerstijl adhv voorbeelden

**Vertel minder over:**

- Smalltalk – al die details

**Opmerkingen:**

- C++ moet Modula-2 niet vervangen
- Hoe laat begint het college eigenlijk?
- het lijkt dat de docent zelf niet goed weet wat hij moet vertellen op het college. (Hij is nota bene verantwoordelijk voor SE!)
- er zullen nog wel meer kenmerken zijn (komen in het college?)
- (van wat?) Ik weet nog bijna niets van OOP, daarom volg ik dit college.
- Ik zie alleen maar classes die messages naar elkaar sturen, maar geen harde code. Er zal toch nog een 'traditioneel' programma geschreven moeten worden?
- het is goed dat we 'gedwongen' worden iets te schrijven. Dat is een betere afspiegeling van de academische werkelijkheid.
- Mooi boek!
- Vertel dat de gelezen informatie gebruikt gaat worden. Dat verhoogt de prioriteit.
- Misschien kunt u concrete voorbeelden in C++ gebruiken om de aspecten van OOP te verduidelijken.
- Een soortgelijke enquête aan het eind van de cursus zou misschien ook wel zin hebben.
- Minder stemmen: dat haalt de snelheid uit het college!
- De opmerking dat OOP andere programmeertalen overbodig maakt lijkt mij gezien de problemen en nadelen enigszins voorbarig
- Als student informatica moet je kennis hebben van OO, naast logisch, functioneel en imperatief programmeren.
- En omdat het in de "praktijk" gebruikt wordt.
- Naar mijn mening is het college slecht gestructureerd, en is het redelijk vaag wat nu precies de bedoeling is. Beetje rommelig.

- Het schrijven van een paper is een goede manier om met een vak bezig te zijn. (In de studie zou meer aandacht besteed moeten worden hoe een goede paper geschreven dient te worden.)
- Ga liever door met het college. Op hogere snelheid. Discussies over wat overerving etc is duren te lang. U kunt toch beter zelf wat bijzaken ter verduidelijking noemen dan de studenten de tijd geven even wat missers te plaatsen. Zonde van de tijd.
- Mogelijk zou een educatieve visualisatie meer impact hebben.