Programmeer-curriculum voor HR-TI

Een verkenning

Bijdrage Jacques de Hooge, 3 nov 2022

Al lange tijd is de inhoud van het programmeer-curriculum onderwerp van overdenking en discussie. In de wereld van de programmeertalen is zowel snelle verandering als een behoudende trend te vinden.

Zo'n 30 jaar geleden leek het erop dat de taal C++ z'n langste tijd achter zich had. Multimedia en Internet, dat ging het worden en daar hoorden talen bij zoals JavaScript, HTML en CSS (don't blame me, ik citeer slechts de toenmalige hype). Web-programmer, dat moest je worden, daar waren de banen van de toekomst te vinden. Met het maken van een eenvoudige website hengelde je zo maar 5000 Euro binnen.

Het liep anders. HTML kon iedereen eenvoudig leren, CSS kon niemand leren en JavaScript bleek een snel bewegend doel, dat elke 3 jaar compleet van gedaante veranderde. Uiteindelijk kon elke middelbare scholier met wat vrije tijd voor 150 Euro een website maken. C++ daarentegen bleek niet weg te branden, goed voor een stevig inkomen, en staat samen met C nog steeds aan de basis van bijna elke technology stack.

Nieuwe talen kwamen op. Python veranderde van een simpele scripttaal in een breed ingezette OO taal, populair voor van alles en nog wat, van eenvoudige besturingen tot ingewikkelde AI en DataScience. Daarmee verloor de taal wel heel wat simplicity, en kreeg er dubieuze toeters en bellen bij. Zo gaat dat, altijd maar weer. Talen beginnen simpel, maar de drang tot optuigen is onstuitbaar.

Java werd onder leiding van Microsoft kundig de browser uitge"scared", waarmee de ABM coalitie (Anything But Microsoft, namelijk Sun, IBM en Digital) effectief een hak werd gezet. C# daarentegen, in feite Java++, werd met veel succes de markt in geduwd. Apple ging z'n eigen gang, met o.a. Objective C, opgevolgd door Swift. Rust diende zich aan als solide, maar niet zo backward compatible opvolger van C++. Google zag het gat en werkt op dit moment aan Carbon, waarbij backward compatibility met de enorme codebase in C++ een primair design goal is.

En dan zijn er Julia, Occam, F#, Scala en vele anderen, allemaal elegant, allemaal met hun eigen aanhang. De voordelen liegen er niet om, maar welke gaat het nu worden? Dat weet niemand.

Wat betekent dat voor het vierjarig curriculum van HR TI? Er is een beperkte tijd, waar het één zit kan het ander niet zitten en de breedte van het vakgebied blijft toenemen. What to do?

Wat is Technische Informatica?

TI heeft daarin enkele jaren geleden een keuze gemaakt: TI speelt zich af op de grens van hard- en software. Die keuze staat niet voor eeuwig in steen gebeiteld. Om niet alles tegelijk op losse schroven

te zetten, is het echter verstandig deze keuze for the time being in grote lijnen te eerbiedigen.

Welke talen horen eigenlijk bij die “grens tussen hard- en software”? Dat hang er vanaf wat je onder hardware verstaat. Voor de één is dat een single board controller en liggen C, C++, Rust, Carbon en Python voor de hand. Voor de ander is het een heel bedrijfsnetwerk en zijn Java, C# en SQL prime candidates. Voor weer iemand anders draait het om the Internet of Things, waarbij JavaScript en Python in beeld komen. Voor de vierde draait het om het besturen van zware industriële hardware zoals productielijnen en haveninstallaties, een taak die vrijwel altijd door PLC’s wordt gedaan.

De delete-knop

Op dit moment blijkt bij een ruime meerderheid van de C++ tentamen-deelnemers vrijwel geen enkele kennis van het werken met pointers meer aanwezig te zijn, zoals dat in C is aangeleerd. Deze waarneming heb ik de afgelopen dagen gedaan tijdens het samen met collega’s afnemen van C++ assessments. Het gaat hierbij vaak om serieuze, slimme studenten. Elke assessment opnieuw heb ik mensen gevraagd het adres van een floating point variabele op te bergen in een pointer en dan via deze pointer de betreffende floating point variabele te wijzigen. De meesten konden het niet, hoe zeer ik ze ook op het goede spoor probeerde te zetten. Weg. Deleted. En, nogmaals, het gaat om slimme studenten met een goede instelling. De kennis daalt niet in, maar vloeit weg. Elk concept dat niet bij herhaling door de studenten practisch wordt toegepast is reddeloos verloren. Dat kost tijd, en we hebben maar 4 jaar. Er moeten dus keuzen worden gemaakt.

Wie maakt de keuzen

Wij kunnen keuzen maken voor de student. We kunnen zeggen: C(++) zit het dichtst bij de hardware, dus daar leggen we het accent. Of we kunnen zeggen: Veel werkgevers werken met C# op Azure dus dat gaan we doen (of moeten we dat aan INF overlaten?). Of we kunnen zeggen: Kijk eens hoe Python in de lift zit! Veel Python, beetje C erbij voor als het echt snel moet, that’s it. Of we kunnen het Internet of Things serieus nemen en zeggen: Bewegend doel of niet, JavaScript hoort erbij. Of we kunnen zeggen, laten we eindelijk eens kiezen voor mathematical rigour: Functional Programming. Of laten we een toekomstige winnaar pakken: Rust of Carbon. Of we kijken naar de haven en industrie: PLC’s. Mijn voorstel is dat we dat niet doen. Mijn voorstel is dat we de student laten kiezen, maar wel uit een aantal samenhangende clusters met een gemeenschappelijk basis, zodat men niet voor een pretpakket kiest en uiteindelijk met een lappendeken van niet op elkaar aansluitende kennis en vaardigheden komt te zitten. Daarnaast kunnen desgewenst vakken uit een andere programmeer-cluster worden gekozen, maar niet “in plaats van”.

Voorstel voor samenhangende programmeer-clusters met een gemeenschappelijke “foundation”.

Studenten moeten mininaal alle vakken uit de Foundation en alle vakken uit 1 cluster behalen.

Foundation:

Introduction to C

Introduction to Python

Cluster 1: Embedded programming

Advanced C (esp. pointers)

Advanced Python (esp. classes, e.g. MicroPython)

C++

Rust

Cluster 2: Industrial controls programming

PLC-programming

Advanced C (esp. pointers)

C++

Advanced Python (esp. classes, for SCADA systems)

Cluster 3: Technical applications programming

Java en/of C#

Introduction to Functional Programming

SQL

Internet Programming

Tot slot

De vakken per cluster kunnen en zullen blijven veranderen. IT is nu eenmaal een vakgebied in snelle beweging. Belangrijk is vooral de keuze van de individuele studenten voor een cluster in plaats van “one size fits all”. Dit maakt het mogelijk dieper op het geleerde in te gaan en te zorgen dat de kennis ook werkelijk beklijft door oefening in uiteenlopende projecten.

Zo houden we de hoeveelheid stof behapbaar binnen 4 jaar.

Zo kan iedere student een keuze maken die bij aanleg en interesse past.

Zo kan iedere werkgever een student vinden die bij de behoefte past.