Van onderwijsvisie naar realisatie

Verwerkelijking van de uitgangspunten binnen vakken en projecten

Inleiding

In hoofdstuk 5 van het “Opleidingsprofiel Technische Informatica” (verder aan te duiden met OTI) wordt een visie op onderwijs geschetst, gebaseerd op de volgende vijf uitgangspunten:

1. Studeren met plezier
2. Studeren met nieuwsgierigheid
3. Studeren binnen een contextrijke leeromgeving
4. Studeren door te leren van en met andere studenten en docenten
5. Studeren met eigenaarschap en verantwoordlijkheid

In hoofdstuk 6, 7 en 8 wordt deze visie verder uitgewerkt op niveau van leerlijnen, namelijk de praktijklijn, de kennislijn en de studentgestuurde lijn. Daarbij komt ook toetsing aan de orde.

Dit document gaat over de volgende stap, het concreet maken van de genoemde visie binnen projecten, vakken, stage en afstuderen. Om deze stap inderdaad zo concreet mogelijk te maken, wordt voorbeelden ontleend aan het huidige onderwijsaanbod. Dit onderwijsaanbod is een bewegend doel. De voorbeelden kunnen echter worden vertaald naar andere projecten en vakken.

De voorbeelden zijn vrij ver uitgewerkt, ook weer om ze zo concreet mogelijk te maken. Dat houdt niet in dat de details in steen gebeiteld staan.
De bedoeling is ideeën aan te dragen voor een haalbaar pad. Dat pad kent uiteraard vele variaties.

Voor een heldere structuur wordt als indeling van dit document de splitsing in leerlijnen aangehouden. Echter zijn ook juist de dwarsverbindingen tussen met name de praktijklijn en de kennislijn van belang, met name voor uitgangspunt 3: Studeren binnen een contextrijke leeromgeving. De praktijklijn kan een deel van de context verschaffen die studenten nieuwsgierig maakt naar de kennislijn. Daarmee wordt ook uitgangspunt 2 geraakt.

Opmerking: In dit document zijn consequent woorden als *hij* en *hem* gebruikt. Hiermee worden alle genders bedoeld.

Lift-project

Wat is het

Het lift-project is op dit moment voor alle HR-TI studenten de eerste kennismaking met een stukje praktische vakinhoud. Studenten bouwen met hun halve stamgroep een lift. (Stamgroepen, bestaande uit ca. 8 studenten, zijn een middel om studenten te helpen aansluiting vinden bij studiegenoten.) Iedere student bouwt een verdieping, bestuurd door een Arduino Uno. Het gaat daarbij om een numerieke display, knoppen en het detecteren van de liftpositie.

Daarnaast is er een centrale besturing, eveneens een Arduino, die de verdiepingen coördineert en de liftmotor aanstuurt. De verdiepings-Arduino's en de centrale Arduino communiceren met elkaar via een eenvoudig serieel protocol. Programmering gebeurt in principe in C, sommige studenten gebruiken de C++ faciliteiten die in de Arduino IDE aanwezig zijn.

De studenten gaan dit project in met een minimum aan voorkennis. De meesten kunnen net een beetje programmeren in C, netwerken en protocollen zijn nieuw voor ze en op een paar hobbyisten na weten de meesten weinig van hardware. De werkwijze is bij de meesten verkennend en ad-hoc. Goed kijken naar het logica en mechanica van een echte lift gebeurt weinig tot niet. Met een laser-cutter worden de wanden van de schacht en de liftkooi uitgesneden, waarna één en ander in elkaar wordt gezet, vaak met behulp van een lijmpistool.

Wat betreft de electronica gaan studenten aan de slag met de zogenaamde brooddoos, een bak met hardware die iedere TI student aan het begin van z'n studie krijgt. Als het geheel niet blijkt te werken wordt er veel gepiekerd en vaak lukraak geprobeerd. Soms helpen ze elkaar, soms zitten ze in hun eentje te modderen. Systematisch zoeken naar fouten, door hard- en software afzonderlijk te testen en de mogelijke foutoorzaken één voor één te elimineren is voor de meesten onbekend terrein.

Er wordt vooral veel gekeken naar pin-nummers en draadjes, bij dat laatste soms maar lang niet altijd geholpen door systematisch kleurgebruik. Het verschil tussen stroom en spanning, hoe je deze grootheden zou kunnen meten en waarom je dat zou doen is bij velen onbekend. Dat er ergens in het gebouw universeelmeters liggen te wachten om dichter bij de foutoorzaak te komen is bij vrijwel geen enkele student in beeld. Dat ook een ledje met een serieweerstand je hierbij zou kunnen helpen is een goed bewaard geheim.

Enerzijds zou je kunnen zeggen dat de uitgangspunten 4 en 5 hier vollop aan de orde kunnen komen. Anderzijds zitten sommigen wel erg lang in een impasse. Als je niet weet dat er systematische manieren bestaan om fouten te traceren en dat er spullen en truukjes bestaan die je daarbij kunnen helpen, blijf je misschien worstelen zonder op het idee te komen dat andere mensen je verder zouden kunnen helpen. Je weet immers nog niet dat er een "verder" is. Enige worsteling en spontaan ontwaken het eigen initiatief om een medestudent of docent om hulp te vragen is leerzaam. Een te lange

worsteling kan zowel ontmoedigend als tijdverspillend zijn. Het is een kwestie van balans.

Veel groepen maken een soort taakverdeling, op basis van veronderstelde aanleg. Vaak is er wel een student in het groepje die al eerder geprogrammeerd heeft. Deze neemt dan vaak de software voor z'n rekening. Soms is dat alleen de centrale software, soms programmeert diezelfde student ook alle verdiepingen. Bij vragen van een docent over de voortgang van de software wordt dan naar deze persoon verwezen, soms met enig ontzag. Ook hier zitten weer twee kanten aan. De taken verdelen is goed. Ieder op z'n eigen eiland is, blijkens uitgangspunt 4, niet de bedoeling.

Het communicatieprotocol tussen de Arduino's is een verhaal apart. Studenten focussen vrijwel zonder uitzondering op de technische details van I^2C. De vraag *welke informatie* de verdiepingen zouden moeten uitwisselen wordt pas in laatste instantie gesteld. Bij aanvang van de studie is dit begrijpelijk. Het heeft ook te maken met de instroom. We krijgen nu eenmaal niet alleen studenten met een (in aanleg) goed abstractievermogen. Deze "details eerst" aanpak blijkt echter ook in hogere leerjaren nog door veel studenten te worden gevolgd, waardoor men zich bijvoorbeeld bij het afstuderen helemaal commit aan een bepaald merk of type hardware, die soms niet of niet snel genoeg leverbaar blijkt. Wat dat betreft zou er binnen de studie een constante aansporing moeten zijn, ontwerpproblemen wat meer vanuit een (zich ontwikkelende) helikopterview te benaderen.

En dan is er nog het aspect van de geïnvesteerde aandacht, tijd en energie. Het liftproject is niet de enige studieactiviteit. Dat een bepaalde studieactiviteit even alle aandacht opeist is een bekend fenomeen. Het antwoord van een willekeurige student op de vraag waarom zij dit of dat nog niet gedaan heeft, is vaak: eind van de week moet project zus of zo klaar zijn. Plannen blijkt iets dat geleerd moet worden. Positief punt: Al doende leert men vaak ook dit. Negatief punt: Indien een project of vak te veel energie wegzuigt bij andere studieactiviteiten kan uitval het gevolg zijn. Ook hier weer de noodzaak van balans.

Uiteindelijk ontstaat een lift die het wel of niet doet. Daarbinnen zijn er nuances. Liften die een enigszins efficiente volgorde van passagiers oppikken hanteren zijn in de minderheid. Liften die het helemaal niet doen gelukkig ook. Al met al leren de meesten heel wat van dit project. Een mooi project dus, met zowel technische als professional en interpersonal skills aspecten. De ideeën die nu volgen betreffende het nog meer verwerken van de besproken onderwijsvisie kunnen er een nog mooier project van maken.

Het project door de bril van de vijf uitgangspunten

1. Studeren met plezier

Als je de studenten op onze verdieping aan dit project bezig ziet, valt de informele sfeer en het contact tussen de studenten op. Het is een soort zoemende bijenkorf, zowel op het onderwijsplein als in de lokalen er omheen en ook in het stadslab. Een mooi eikpunt is de situatie tijdens de Corona lockdowns. Wat waren ze blij elkaar af en toe te zien. De meesten varen wel bij fysieke aanwezigheid en contact

met studiegenoten en docenten. Voor studenten die niet te ver weg wonen is ook het dagritme een positief element: uit je bed komen en ergens heen gaan waar het gezellig is.

Daarnaast zijn velen gefascineerd door het knutselen met al die mysterieuze blokjes, busjes, draadjes en lampjes en knopjes. Ook de aansturing hiervan met software heeft voor velen z'n bekoring. Plezier in de inhoud van het vak, al is de diepte en dagelijkse werkelijkheid van dat vak nog grotendeels aan het oog onttrokken. Degenen die dat helemaal niet hebben? Misschien komt het nog, misschien past een andere studie beter bij ze. Niet alles is beïnvloedbaar.

Tenstlotte is er het plezier van de overwinning: Moeilijke problemen tegenkomen, ze één voor één oplossen en tenslotte een werkend systeem. Zelf gemaakt, samen met je team. Een vreugdevolle ervaring en positieve bekrachtiging. Yes, I can!

En er zijn ook zaken die beter kunnen. Urenlang worstelen en er niet uitkomen, waarna iemand anders het wel even in orde maakt doet afbreuk aan het plezier. Het project is voor veel eerstejaars hoog gegrepen, zelfs in z'n eenvoudigste vorm. Iets meer structuur en begeleiding zal helpen. Dat hoeft niet persee altijd 1 op 1 persoonlijke begeleiding te zijn. Ook een reader of video waarin wordt getoond hoe je zoiets nu eigenlijk aanpakt is begeleiding.

Het gaat daarbij niet om een stap voor stap "handleiding" maar meer om aanwijzingen over hoe je een zo'n combinatie-project van hard- en software behapbaar maakt. Aanwijzingen die voor docenten vanzelf spreken zoals systematisch gebruik van draadkleuren, het onafhankelijk testen van subonderdelen van zowel hardware als software en het grondig testen van de hardware voordat deze als debugging vehicle voor de software wordt gebruikt zijn zo maar wat zaken die studenten op het goede spoor kunnen zetten, zonder ze daarbij het plezier en het leereffect van zelf zoeken en vinden te ontnemen.

Wat betreft structuur kan een voorbeeld-stappenplan worden aangereikt, een beetje zoals een voorbeeld-indeling van een afstudeer- of stageverslag. Hierin worden een aantal verstandige stappen bij het maken van een dergelijke toepassing genoemd, met per stap het nut ervan en wat hints omtrent de praktische uitvoering. Een voorbeeld van zo'n stappenplan staat in onderstaande tabel.

Stap	**Waarom**	**Hoe**
Eenduidig boven tafel krijgen van de vereisten aan het projectresultaat	Helder vasleggen wat precies moet worden gemaakt	Bestuderen en samenvatten van bestaande specificatie of, indien deze niet voor handen is, achterhalen en vastleggen ervan
Kiezen van principe-oplossingen om het projectresultaat te verkrijgen	Zo'n principe-ontwerp maakt duidelijk welke taken er zoal zijn bij de realisatie ervan	Samen creatief nadenken en overleggen, zodat gebruik wordt gemaakt van alle ideeën binnen de groep
Inventariseren van kennis, vaardigheden en voorkeur Het bieden van enige sturende	Zo kan men er voor zorgen dat ieder datgene doet waar zij goed in is of wil worden	Ieder groepslid geeft aan waar zij goed in is of wat zij wil leren.

begeleiding is in overeenstemming met niveau 1 van tabel 2 in het OTI.		
Inventariseren van taken binnen het project	Zo kunnen deze taken worden verdeeld onder de groepsleden	Zoek taken waarvan het resultaat onafhankelijk kan worden getest
Verdelen van de taken	Doorlooptijd verkleinen	Elk groepslid geeft minimaal 2 voorkeurstaken aan. Uiteindelijk wordt in overleg besloten. De verdeling van taken over groepsleden hoeft niet 1 op 1 te zijn
Inventariseren van de benodigde hardware-onderdelen	Het kan zijn dat niet alles op voorraad is. Sommige dingen moeten zelf worden gemaakt uit basismaterialen	Opstellen van materialenlijst met bron waaruit deze kunnen worden betrokken en, indien van toepassing, kosten
Splitsen van het totale hardware systeem in onafhankelijk testbare deelsystemen	Het is dan snel duidelijk in welk deelsysteem “het niet doet”	Deelsystemen zo kiezen dat ze zo min mogelijk verweven zijn
Splitsen van de benodigde software in onderdelen	Zo kan aan die onderdelen onafhankelijk worden gewerkt en kunnen ze onafhankelijk worden getest	Maak de splitsing zo dat de onderdelen zo min mogelijk met elkaar verweven zijn
Vervaardiging en individueel testen van hardware-deelsystemen	Een systeem dat uit betrouwbare deelsystemen wordt opgebouwd is meestal zelf ook betrouwbaar	Gebruik van test-hulpmiddelen zoals een universeelmeter
Testen van de hardware als geheel	Zo verkrijgt men een stevige basis waarop in een volgende stap de software kan worden getest	Dit kan door rechtstreeks de I/O van de besturing(en) in te lezen en aan te sturen met behulp van een eenvoudig testprogrammaprint (type (ship) .__name__, ship.name, ship.length, ship.orientation)
Testen van delen van de software op de grondig geteste hardware (unittest)	Zo wordt snel duidelijk waar eventuele fouten zitten	De diverse software-onderdelen aansturen via speciale, eenvoudige test-code
Testen van de samengevoegde software op de samengevoegde hardware (integratietest)	Hiermee wordt een correcte samenwerking tussen alle delen getest	Stel een testplan op aan de hand van de vereisten, voer dit stap voor stap uit en leg de resultaten vast
Opstellen gebruiksdocumentatie	Een gebruiker moet weten hoe hij het systeem kan bedienen	Bekijk het systeem vanuit het standpunt van een gebruiker zonder technische kennis

Een dergelijk stappenplan zorgt dat de studenten niet helemaal “het bos in gaan”, want daar leren ze weinig van. Goede gewoonten, zoals een systematische aanpak bij opbouw en testen kunnen niet vroeg genoeg worden aangereikt. Het bieden van enige sturende begeleiding is in overeenstemming met niveau 1 van tabel 2 in het OTI.

Zo krijgen de studenten enige steun bij het aanpakken van het project en vermindert de frustratie, maar blijven de uitdaging en het plezier van de overwinning volop bestaan.

2. Studeren met nieuwsgierigheid

Sommige studenten zijn van nature nieuwsgierig. Anderen zijn vooral gefocust op het snel en succesvol afronden van de studie. Dat mag, maar zonder nieuwsgierigheid worden veel kansen op het zich eigen maken van kennis gemist. Er ontstaat dan een patroon van kortstondige inspanning om een voldoende te halen, waarbij de opgedane kennis nauwelijks blijft hangen. Immers, die kennis is dané bijzaak, het gaat primair om het behalen van een voldoende en dat kan vaak met oppervlakkig, vluchtig begrip. Wat moeten we precies weten? Wat zijn de minimumeisen aan het projectresulaat? Delete en door.

Deze minimalistische strategie leidt er toe dat een deel van de studenten veel minder leert dan zou kunnen. Er is letterlijk sprake van verspilling. Een belangrijk deel van de onderwijs-effort gaat ernaast. Dat is niet de bedoeling van de maatschappelijke investering in onderwijs. Al bij project 1, op dit moment het lift-project, zijn er mogelijkheden om de nieuwsgierigheid expliciet te stimuleren. Bij studenten die niet van nature “aangedreven worden” door nieuwsgierigheid voor techniek, maar meer door het carriere-perspectief, biedt vooral stimuleren vanuit de praktische toepassing mogelijkheden. Zulke studenten zijn immers al bezig met het moment dat ze bij een bedrijf aan de slag kunnen en vormen zich daarvan een beeld. Hiervan kan meer gebruik worden gemaakt.

Wat ben ik aan het maken? Alleen maar een aantal plankjes, met wat knopjes, displays, draadjes, sensoren, Arduino’s en een stappenmotor? Veel studenten maken zelf de stap naar “een lift” niet. Er zijn liften in ons gebouw. Hoe werken ze precies? Wat voor strategie volgen ze als op meerdere verdiepingen iemand op de knoppen drukt. Zo iets zou onderwerp van een eenvoudig experiment kunnen zijn, bijvoorbeeld met een goederenlift die op dat moment geen spitsuur heeft. Hoe zit het eigenlijk met energie-verbruik? Kost het niet heel veel energie om elke keer een zware liftkooi op te hijsen? Hier zijn twee didactische standpunten mogelijk:

1. Het doet er niet toe hoe een echte lift precies werkt. Maak nu maar wat je minimaal moet maken. Doe je ding, we vinken punten af op ons lijstje en als je houten hokje op en neer gaat en op de juiste plekken stopt, en de displays de juiste cijfers laten zien heb je een voldoende. Zo wordt onbedoeld de minimalistische strategie gestimuleerd en de berekende student geboren. Wat is er minimaal nodig voor een voldoende.

2. Leren gaat niet alleen over techniek maar ook over nieuwsgierigheid naar praktische toepassingen. Wat is een echte lift voor ding en hoe werkt hij precies? En dan gaat het om alle aspecten. Het feit dat er dan bijvoorbeeld een contragewicht op en neer gaat is geen bijzaak meer. Het maakt immers deel uit van die echte lift en die staat centraal!

Aanpak 2 is omdenken voor de gepassioneerde ras-technicus. Immers die is vooral geinteresseerd in de knopjes, lampjes, displays, draadjes, compuéters en software. Maar interesse voor het hele praktische fenomeen “lift” is wel een manier om juist de studenten met een meer praktische insteek aan te spreken. Dit vergt van de docenten dat ze bereid zijn, niet alleen in te gaan op de TI kant van een lift, maar ook op aspecten die in hun ogen “bijzaak” zijn.

Automatisering is maar één aspect van een lift. Een lift is een materieel ding. Een lift vervult een functie voor de gebruikers. Een lift wordt ergens gemaakt. Er wordt aan verdiend. Er werken mensen aan. Er is een productielijn. Er zijn toeleveranciers.Veiligheidsvoorschriften. Esthetiek. Kortom allerlei kanten die niet helemaal “techniek” en ook niet helemaal “skills” zijn. En zo komen we vanzelf op uitgangspunt 3.

4. Een contextrijke leeromgeving

Plezier, nieuwsgierigheid en contextrijkheid gaan hand in hand. Juist studenten bij wie de pure techniek niet de voornaamste brug naar interesse is, komt de context te hulp. Interesse in de context is interesse in “waar je het voor doet”. En als je weet waar je het voor doet, is kennis geen overbodige ballast meer maar een zinvolle lading. Dat maakt het een stuk makkelijker kennis te verwerven. Leren uit interesse gaat nu eenmaal veel soepeler dan leren “omdat het moet”. Studenten vellen een snel en automatisch een oordeel: “ballast” of “lading”. En wie ze ballast (in hun ogen) probeert op te dringen, raakt onvermijdelijk teleurgesteld. Niet dat alle studenten echt kunnen beoordelen wat ballast is en wat niet. Daarvoor weten velen nog te weinig van de beroepsprakijk. Deze beperking komt aan de orde bij uitgangspunt 5: Studeren met eigenaarschap en verantwoordlijkheid.

De beroepspraktijk is een deel van de context. Immers, producten spelen niet alleen een rol in het leven van consumenten, maar ook van producenten. Hoe meer context bij een project wordt geboden, des te meer zal het een brede groep studenten aanspreken.

Twee kanttekeningen:

Ten eerste zijn er natuurlijk grenzen. Context zonder een significante hoeveelheid harde, technische inhoud mag dan nuttig zijn, onder de vlag van Technische Informatica hoort het bijbrengen van zulke harde technische kennis en vaardigheden nu eenmaal tot de lading. Voor puur maatschappelijk of bedrijfskundig geïnteresseerden zijn er andere studies.

Ten tweede: Als de nadruk vrijwel geheel op de context komt te liggen, verliezen studenten met een sterk technische focus hun interesse. Er zijn natuurlijk studenten die alle context zowiezo maar onzin vinden. Het is niet de bedoeling het onderwijs geheel in te richten op deze eenzijdige gefocuste groep. Een veel grotere groep echter heeft een passie voor techniek en is wel in staat te zien dat een bepaalde mate van engagement met de context nodig is om een goede technicus te zijn, maar ventileert ongezouten zijn ongenoegen als de techniek ondersneeuwt. Hier hebben ze niet voor gekozen. Dat klopt. De pure technici horen ook aan boord thuis. Voor een deel valt dit op te lossen met

mogelijkheden tot verdieping. Deze verdieping hoort ook expliciet beoordeeld en gewaardeerd te worden.

Bij de lift kan het gaan om een slimme prioritisering van de manier waarop de etages doorlopen worden. Of om meerdere liften die op zinnige wijze samenwerken om mensen zonder omwegen en met zo min mogelijk wachttijd naar de juiste etage te krijgen. Of om een strakke, bijvoorbeeld object georiënteerde, programmeerstijl met een helder, goed gedocumenteerd ontwerp. Of om een elegant protocol, waarbij een OSI-achtige lagen-indeling is aangehouden. Of om een speciale faciliteit om voorrang te geven aan urgent gebruik. Belangrijk is dat deze mogelijkheden in de practicum-omschrijving zijn aangegeven en dat de docenten ze op waarde schatten en meenemen in de beoordeling.

Zoals besproken sluiten context en technische verdieping elkaar niet uit. Ze versterken elkaar eerder. Het meenemen van meer context vergroot de mogelijkheden tot verdieping. De voorgestelde uitbreidingen van het materiaal met zowel een bredere context van het fenomeen lift als verdieping aan de hand van bijvoorbeeld de gedane suggesties voor “extra’s” vormen een eenmalige investering in het project.

Daarnaast is aandacht van de docenten voor de context nodig, ook als dat niet hun eigen primaire interesse is. Van belang is daarbij dat niet te gauw wordt aangenomen dat een aspect van een echte lift “er voor het project niet toe doet”. Dat is de visie van een techicus pur-sang, sommige studenten hebben dezelfde focus, anderen varen wel bij meer context. In ons gebouw bevinden zich diverse liften. Die kunnen op z’n minst grondig bekeken worden, inclusief prioriteits-sleutels, contragewichten en deurbeveiliging. Daarnaast kan een (goederen)lift in overleg met gebouwenbeheer voor experimenten betreffende service-volgorde van etages worden gebruikt. De resultaten kunnen in het project-verslag worden opgenomen en al dan niet worden gebruikt als basis voor een eigen algoritme.

5. Studeren met eigenaarschap en verantwoordelijkheid

Dit is bij project 1 voor een deel van de studenten een beginnend “werk in uitvoering”. Op de middelbare school wordt je over het algemeen eenduidig vertelt wat je moet doen. De enige verantwoordelijkheid is dat je dat ook werkelijk min of meer doet. Van eigenaarschap is heel vaak geen sprake. Je moet gewoon naar school en daar moet je leren. Of een dergelijke aanpak verstandig is wordt hier in het midden gelaten. Met alle gevolgen die de leerplichtwet heeft gehad is dit mogelijk een negatief bijeffect. Leren is geen gunst maar een plicht. Kinderen in Afrika of India denken daar vaak heel anders over. Maar, zoals gezegd, het is geen manco van de leerlingen maar van het systeem. En iets beters bedenken is niet eenvoudig. Voor HBO opleidingen is het een gegeven dat veel van de studenten met een dergelijke mentaliteit binnenkomen, gecombineerd met een deficiet aan basiskennis zoals rekenen en lezen. Studenten hebben tegenwoordig ook vaardigheden, die studenten vroeger niet hadden. Blijft wel het genoemde deficiet. Dit is geen waarde-oordeel, maar een vaststelling die door breed onderzoek en meten wordt ondersteund.

Bij project 1 kan een begin worden gemaakt me het aankweken van eigenaarschap en

verantwoordelijkheid, maar met mate. Dit is iets dat geleerd moet worden. Een beetje vallen en opstaan hoort daarbij. Vallen en niet meer opstaan is natuurlijk niet de bedoeling. En dat is wel wat er uiteindelijk wat er met een flink deel van de studenten gebeurt.

Centraal staat het begrip “positieve bekrachtiging”. Geef studenten de verantwoordelijkheid die ze aankunnen, maar niet (veel) meer. Op die manier komen ze in een positieve spiraal: Ik kan dit zelf, ik kan mijzelf managen. De hoofdrol is hier weggelegd voor SLC (Studie Loopbaan Coaching). Studenten worden zoveel mogelijk losgelaten maar zijn wel in beeld.

Echt eigenaarschap is voor een deel van de studenten nog te veel gevraagd. Naast het aanleren ervan mogen er daarom in project 1 ook externe prikkels zijn. Een voorbeeld daarvan zijn tussendoelen met bijbehorende deadlines. Een ander voorbeeld is, dat er eisen worden gesteld aan de taakverdeling, om te zorgen dat er geen “meelopende wielen” zijn. Zo mag geeist worden dat alle groepsleden tenminste passieve kennis hebben van wat de anderen hebben gemaakt. Dus, specifiek, ook iemand die de software niet zelf geschreven heeft, dient deze regel voor regel te kunnen uitleggen. En ook iemand die de hardware niet zelf bedraad heeft, dient te kunnen uitleggen wat waarop aangesloten is en waarom. Zelfde geldt voor de volgorde-strategie bij het aandoen van etages en het communicatieprotocol.

Deze manier van beoordelen hoeft niet extreem veel tijd te kosten, maar 15 a 20 minuten per groep is echt het minimum. Daarbij moeten ook individuele vragen gesteld worden, waarbij de anderen niet “voorzeggen”. Dit is niet alleen controle maar ook zorgen dat studenten zich gezien voelen.

Het bank-project

Wat is het

Aan het einde van het eerste studiejaar doen TI studenten het bank-project. Ze weten dan al wat meer van hardware en hebben dan behalve de programmeertaal C ook enig Java gehad. Ze hebben al wat ervaring met hoe je een project in principe aanpakt, samenwerkend met andere studenten.

Het bank-project gaat over realisatie van een aantal banken met pinautomaten. Elke pin-automaat is via een data-verbinding gekoppeld aan een bank en alle banken zijn aan elkaar gekoppeld in een LAN. De pinautomaat omvat zowel hard- als software aspecten. Het gelduitgifte-systeem is een plek waar de studenten veel creatieve, praktische oplossingen kwijt kunnen. Werk je voor het uitgeven van biljetten met rolletjes? Of met zuigers? Of zijn er nog originele en effectieve alternatieven? Of maak je stiekem je biljetten onopvallend wat dikker, zodat het meer kaartjes dan papiertjes zijn en makkelijker te manipuleren?

De kast van de uitgifte-automaat is een geliefd doelwit voor lasercutters, 3D printers, lijmpistolen en ander gerei. Daarnaast worden de meeste automaten met zichtbaar plezier voorzien van grappige namen en logo’s.

Ieder groepje studenten maakt een bank. Omdat banken met elkaar moeten communiceren is overleg tussen de groepjes over het communicatie-protocol nodig. Vaak neemt een beperkt aantal studenten hierin het voortouw. Dat is conform de realiteit. Ook in het “echt” worden dergelijke zaken niet beslist in een brede maatschappelijke disussie.

Een ander aspect van het bank-project is security. Hoe check je de indentiteit van een klant. Hoe zorg je dat de geld-automaten niet meer flappen tappen dan het saldo van de klant rechtvaardigt. En hoe zorg je dat er geen geld wordt weggesluisd door simpelweg het interbankaire dataverkeer te hacken?

De afsluiting van het project is een happening die gekenmerkt wordt door een combinatie van serieuze spanning en plezier. Studenten mogen proberen elkaars bank te hacken en de robuustheid van de geldautomaten worden op diverse wijzen op de proef gesteld. Techniek- en skillsdocenten lopen rond om het resultaat te beoordelen. Een groot deel van de verdieping is gevuld met leven en gedoe, draadjes, schermpjes. De TI-sfeer op z’n best.

Het project door de bril van de vijf uitgangspunten

1. Studeren met plezier

Vooral op de “slot-happening” is het plezier bij de meesten duidelijk aanwezig. Een belangrijk deel van dat plezier is sociaal plezier. Studenten kijken naar de resultaten van andere groepen en leveren commentaar, vaak met een knipoog. Het contact tussen de docenten en studenten heeft op dat moment

een serieus doel, beoordeling, uiteindelijke na een eventuele verbeterslag uitmondend in het al dan niet behalen van studiepunten.

Maar de sfeer is behalve serieus ook informeel. De relatie tussen student en docent is een belangrijk element van deze gebeurtenis. Die kan zich uiten in een opmerking, waaruit blijkt dat docent en student elkaar langer kennen dan vandaag. Of gewoon in een blik of grapje. Of in een serieus coachend verhaal om de student alsnog op het goede spoor te zetten voor een tweede kans. Of in erkenning van het talent of het harde werk dat blijkt uit het resultaat.

gezien voelen
Alleen al de afsluiting van het project draagt op deze manier in belangrijke mate bij aan het studieplezier. Natuurlijk telt voor velen vooral het resultaat. Dat is ook okee. Tenslotte is er sprake van een opleiding met een inhoudelijke doelstelling. Echter bij die doelstelling hoort ook groei van de student als mens. Dat samenwerken behalve resultaatgericht ook gewoon domweg leuk kan zijn en kan leiden tot je verbonden voelen met je eigen groepje en met andere studenten, dat er een prettig soort competetie kan ontstaan en dat je je in een samenwerkings-situate gezien en “at home” kunt voelen is een belangrijke leerervaring en vergroot de kans op studiesucces.

Aanwezig zijn, meegenomen worden in de stroom, informeel contact met inbreng van pril, maar groeiend eigen vakmanschap, het zijn allemaal zaken die net zo belangrijk zijn als de techniek omdat ze een opstap zijn naar plezier in je werk, iets dat nauwelijks op waarde valt te schatten.

Voor de nuttigheidsdenkers: Wie met plezier werkt, levert over het algemeen ook goede resultaten af. Motivatie en inzet zijn hier de sleutelwoorden. Die motivatie is intrinsiek, komt voort uit de activiteit zelf. Een betere motivatie is er niet.

Het bank-project draagt op de beschreven manier bij aan het verwezelijking van het uitgangspunt 1: Studeren met plezier. Een klein punt zijn misschien de wat minder zichtbare aspecten van de project-resultaten. Dat de bedrading netjes is, dat het protocol logisch in elkaar zit en dat de geldautomaat met behulp van 3D printing er behoorlijk professioneel uitziet en de schermdialogen eenduidig zijn, wordt ook bij oppervlakkige beschouwing duidelijk. Dat één en ander doet wat het moet doen blijkt ook bij de demo.

Het verschil tussen goed gestructureerde, helder geschreven software en een kluwe veredelde spaghetti is minder zichtbaar. Toch is dit aspect iets waarin de betreffende studenten kennis, energie en vaardigheden hebben geïnvesteerd. Het draagt bij aan hun studieplezier als ook deze inspanning wordt gezien en beloond, in woorden en becijfering. De aanbeveling is om, voor zover dat al niet gedaan wordt, hier expliciet aandacht aan te besteden bij alle groepen. Dit kan door in te zoomen op dat deel van de code, waarvan de docent weet dat het er qua structuur opaan komt. Begin met de grof, bijvoorbeeld bij de indeling in klassen, modules of functies en zoom dan in op één onderdeel, bijvoorbeeld één klasse en, verder inzoomend, één functie. Geef constructieve feedback op alle aspecten die de docent opvallen, zowel positief als negatief. De bevindingen hierbij maken deel uit van het eindoordeel.

Op deze manier oog hebben voor wat zich onder de oppervlakte van het projectresultaat bevindt, maakt

dat de student zich ook in de inspanning voor dit belangrijke deel van het projectresultaat gezien weet. “Gezien worden” draagt bij aan plezier in het werk. Het is daarnaast een stimulans om ook aandacht te besteden voor de “inwendige” kwaliteit van wat wordt gemaakt. Hiervoor uiteindelijk zelf verantwoordelijkheid nemen maakt van iemand een goede technicus, die aan de maatschappij betrouwbare producten levert, zonder dat daarvoor een controle-circus nodig is. Dit draagt bij aan een “high-trust” society, waarin niet enorm veel menskracht en andere middelen hoeven te worden verspilt aan voortdurende controle.

https://en.wikipedia.org/wiki/High_trust_and_low_trust_societies

De gewoonte, ook aandacht te besteden aan wat niet direct zichtbaar is, leidt naar zelfrespect, en daarmee ook weer naar plezier, niet alleen in de studie, maar ook later in het dagelijks werk.

2. Studeren met nieuwsgierigheid

Je kunt niet nieuwsgierig zijn naar iets waarvan je niet weet dat het bestaat. Nieuwsgierigheid wordt gewekt door een tipje van de sluier op te lichten, niet door alles tot in detail expliciet te laten zien.

Voorbeeld: Het ontwerp van een GUI voor de geldautomaat. Studenten benaderen het GUI probleem intuïtief en op basis van eigen ervaring met bijvoorbeeld games. Of ze kopiëren vrijwel één op één de GUI van een bestaande geldautomaat. In het eerste geval is de oplossing meestal suboptimaal. Immers een geldautomaat dient een andere doel en wordt onder andere omstandigheden gebruikt dan een game. In het tweede geval is er geen sprake van enige prikkel tot innovatie. Kopiëren wat anderen hebben bedacht werkt soms goed genoeg maar zet niet aan tot meer dan oppervlakkige verbetering. En juist van oog voor innovatie-mogelijkheden moeten we het in Nederland met onze dure arbeid hebben.

Het is bij de meeste studenten niet bekend dat er algemene, goed doordachte, experimenteel onderbouwde richtlijnen bestaan voor wat “een goede GUI” is. Zo mag er bijvoorbeeld nooit meer dan één cursor tegelijk om aandacht vragen. De cursor moet eenduidig aangeven waar de gebruiker “verder moet”.

Onbekendheid met deze proefondervindelijk vastgestelde richtlijn leidt bijvoorbeeld soms tot verwarring bij betaalautomaten in parkeergarages, waar midden in een dialoog op een groter kleurenscherm een subdialoog op een specifiek scherm van de creditcard-lezer dient te worden gevoerd. De gebruiker wordt uit de vanzelfsprekende flow van de huidige dialoog gehaald en vraagt zich even af waar hij nu “verder” moet. Uiteindelijk wennen mensen aan zoiets, maar ideaal is het niet. Wie heeft nooit eens iemand zoekend en peinzend bij een parkeerautomaat zien staan.

We hoeven die GUI-richtlijnen niet voor te kauwen. Echter als de studenten niet weten dat ze bestaan kunnen ze ook niet nieuwsgierig zijn naar wat deze richtlijnen inhouden. Een kleine hint in het cursusmateriaal is voldoende.

Het zelfde geldt voor de veiligheidseisen. Als je niet weet dat er PCI-eisen aan betaalautomaten bestaan, kun je ook niet nieuwsgierig zijn naar wat deze inhouden.

En hoe zit ‘t trouwens met toegankelijkheid voor rolstoelers en visueel gehandicapten. Voor al deze zaken bestaan er richtlijnen. De kans dat de student hier spontaan naar op zoek gaat is vrijwel nul. Echter een kleine hint is ook hier weer voldoende om de nieuwsgierigheid te wekken.

Tip voor dit project dus: Tipje van de sluier!

3. Een contextrijke leeromgeving

De wereld van de financiële instellingen is een andere wereld dan die van de gemiddelde TI-student, een wereld met z’n eigen terminologie, logica en mores. Het is een wereld die tijdens de borrel en in de pers meestal negatief aan de orde komt. Tenslotte, wat gebeurt er precies in die grote gebouwen met veel marmer en mensen in pak? Wie zich er ook maar enigszins in verdiept, weet dat onze maatschappij niet zou kunnen functioneren zonder deze instellingen. Met al hun gebreken dienen ze wel degelijk een doel. Het alternatief is ruilhandel zonder de mogelijkheid waarde op te slaan of buiten de eigen groep uit te wisselen. Dat alternatief is minder idyllisch dan het lijkt. Mooi dat je een kist sinaasappels kunt ruilen tegen een kist appels of bananen. Maar maak je ooit een fruithap voor je kind als alle drie deze vruchten in een ander land en ander seizoen worden aangeboden. Geld als opslag en transportmiddel van waarde is ontstaan uit een behoefte. Voor veel afgeleide financiële producten geldt een vergelijkbaar verhaal, dat weinig bekend is buiten een kring van insiders. Hier ligt bij het bankproject een kans.

Je best doen voor een projectresultaat dat betrekking heeft op een onbekende of zelfs afkeurenswaardige wereld is minder makkelijk dan je best doen voor iets met een “goed” doel. Projecten op het terrein van bijvoorbeeld hulpmiddelen voor gehandicapten zijn een voorbeeld van het tegenovergestelde. Als iemand met een beperking geholpen kan worden met iets wat jij maakt, dan blijkt dat motiverend te werken, intrinsiek, maar ook qua ontmoette waardering door anderen. Tegen de morele aantrekkingskracht van projecten in de zorgsector kan weinig op, en dat is okee. Maar er zijn meer nuttige instellingen dan ziekenhuizen en revalidatie-centra alleen. Banken horen daarbij. Daarbij kan het gaan over micro-kredieten of coöperatieve instellingen of over de meer traditionele banken die hier deel uitmaken van het financiële raderwerk.

Praktisch: Op dit moment gaan mensen in het bankproject aan de slag vanuit een zeer beperkte visie op hoe het nu allemaal echt werkt met interbancair monetair verkeer, beveiliging van verbindingen naar pinautomaten en fysieke beveiliging van die automaten zelf. En trouwens, hoe werkt zo’n gelduitgifte-systeem nu echt. Want het uitschuiven van twee aan elkaar geplakte honderdjes is een fenomeen waarop je helaas lang moet wachten.

Om dit project context-rijker te maken zijn kan bijvoorbeeld een spreker uit de banken-wereld worden uitgenodigd. Dit kan iemand zijn met financiële of technische kennis of allebei.

Voorbeelden:

- Een werknemer van een pin-automaten fabriek (die ongetwijfeld heel veel niet mag vertellen, maar hopelijk ook sommige dingen wel).

- Een financiëel medewerker van een bank die kan vertellen wat er precies gebeurt als iemand in Spanje geldt opneemt van een rekening in Nederland.
- Iemand die, binnen de grenzen van z'n non-disclosure agreement, kan vertellen hoe betalingsverkeer over WAN's of Internet nu *echt* beveiligd wordt.
- Een financiëel expert die een wat genuanceerder licht kan laten schijnen op het verschijnsel bank, tussen de uitersten "dievenbende" en "de glamour van het grote geld" in. De bank als maatschappelijke dienstverlener.

Een excursie is een andere mogelijkheid. Het is dan wel belangrijk dat ook echt een inkijkje wordt gegund dat verder gaat dan een gepolijst verhaal door een voorlichter.

In alle gevallen is sprake van verschaffen van een context die, behalve tot een beter projectresultaat, ook kan leiden tot een bredere visie van de student. Tijdens het afstuderen zijn er regelmatig studenten die er blijk van geven, zich totaal niet verdiept te hebben in de preciese plek van hun afstudeerwerk binnen de bedrijfsactiviteiten en doelstellingen van de opdrachtgever. De kans dat dat afstudeerwerk dan ook echt kan worden gebruikt in de praktijk van alledag verminderd daarmee. Terwijl het nu juist zo motiverend is als wat je maakt ook echt gebruikt wordt.

Deze technische tunnelvisie is beperkend voor de ontplooiing van de student. Immers onbekend maakt onbemind. Met techniek als startpunt kan er bij een project een tot dan toe onbekende wereld voor iemand open gaan. Als we de student willen helpen bij het realiseren van z'n potentieel, dan hoort de context waar mogelijk in beeld gebracht te worden. Natuurlijk niet ten koste van fundamentele technisch-inhoudelijke kennis en vaardigheden. Maar het één sluit het ander niet uit. Het bankproject is wat dat betreft één van de kansen die zich voordoen om de student te leren op te kijken van z'n printplaat of monitor en z'n voordeel te doen met wat hij om zich heen ziet.

4. Studeren door te leren van andere studenten en docenten

Wie de studenten met het bankproject bezig ziet, ziet dat ze van elkaar leren. Dat kan variëren van samen nadenken over welke afspraken nodig zijn voor het communicatie-protocol tot na-apen tot het na-apen van oplossing voor een (enigszins) betrouwbare biljet-uitgifte. Ook na-apen is leren, zij het ook niet de meest "actieve" vorm.

Studenten geven elkaar ook uitleg, bijvoorbeeld over welke aansluitpinnen van een bepaald onderdeeltje waarvoor dienen. Sommige uitleg gaat dieper, zoals een student met wat meer programmeerervaring, die aan een ander uitlegt hoe het precies zit met het gebruik van arrays in C en wat je in Java het beste al dan niet bij elkaar kan stoppen in één klasse.

Ook leren van docenten gebeurt regelmatig. Bij contacten met SLC'ers gaat het bijvoorbeeld om de hoe de studenten omgaat met problemen die hij tegenkomt. Probeert hij dingen alleen uit te zoeken en op wat voor momenten is dat handig, op wat voor momenten niet? Hoe gaat het in de groep? Ergert hij zich aan de opstelling van sommige groepsleden en bespreekt hij dat dan ook met die groepsleden. Escaleert hij indien nodig dergelijke problemen naar een docent, of ergert hij zich alleen, houdt z'n mond en trekt in z'n eentje of met een maatje de kar. Gaan dergelijke zaken deel uit maken van z'n persoonlijke evaluatie? Is dat "klikken" of een mogelijkheid tot groei voor alle groepsleden? Een SLC

docent draagt daarin gezichtspunten aan die nieuw kunnen zijn voor een bepaalde student.

Ook techniek-docenten komen af en toe langslopen en de studenten kunnen met vragen bij ze terecht. De hulp varieert van hints over een oplossingsrichting tot even meekijken bij een concreet technisch probleem. Al met al leren studenten bij het bank-project inderdaad nu al veel van andere studenten en van docenten.

Project-management

Het specifieke karakter van dit project maakt het mogelijk bepaalde leer-elementen nog wat meer expliciet aanwezig te laten zijn. Vanwege de koppeling van de banken van de verschillende groepen aan elkaar is samenwerking tussen de groepen studenten onderling mogelijk. Dit is een grootschaliger samenwerking dan in het liftproject. Daar volstaat informeel contact in principe.

Bij de schaal van samenwerking in bankproject wordt een wat meer formele benadering van project-management, volgorde-afhankelijkheden en manieren om te zorgen dat de producten van de diverse groepen uiteindelijk in elkaar passen.

Project-management is bepaald geen nieuw probleem. Studenten komen echter over het algemeen niet zelf op het idee eens te kijken of er anderen zijn die goede ideëen hierover op papier of YouTube video hebben gezet. Waarschijnlijk zegt de meesten de kreet project-management niets. Hier ligt een kans. De bijdrage van de docent hoeft niet verder te gaan dan uitganspunt 2: Studeren met nieuwsgierigheid.

Van de student kan worden gevraagd student gevraagd worden iets te zeggen over projecten in het algemeen. Daardoor wordt deze gestimuleerd er iets over te lezen. Bij sommigen zal dit interesse wekken. Immers een deel van onze technici worden uiteindelijk na een paar jaar werkervaring als projectmanager of hebben deze incidenteel deze rol als deel van hun werk.

Het leereffect wordt vergroot indien de studenten wordt gevraagd hierover heel kort iets op papier te zetten, *in eigen woorden, niet als standaard-formulering,* zodat de betreffende informatie ook werkelijk “geprocessed” wordt:

Onderstaand wat mogelijke vragen om de student te helpen bij de kern te komen. Standaard antwoorden zijn erbij vermeld als hulp voor de docenten en dus *niet voor de studenten*, van hen wordt gevraagd met een eigen antwoord te komen.

Vraag	**Standaard-antwoord als hint aan docent**
Wat is nou eigenlijk een project?	Een doelgericht stuk werk met als voorwaarden een goed gedefineerd begin, eind, project-resultaat en kostenplaatje.
Waarom is project-management nodig?	Omdat er sturing nodig is om te zorgen dat inderdaad aan de voorwaarden voldaan wordt.
Wat is een project-plan	Een document waarin de genoemde voorwaarden expliciet worden vermeld en bij voorkeur ook

	details zoals een planning die klopt met de genoemde tijdsduur.
Wat is een project-team	Een groep mensen die voor de gelegenheid samenwerkt om een bepaald project te doen.

Er zijn natuurlijk veel meer vragen mogelijk, maar het wekken van een beginnende interesse is voldoende. Hoewel er zeker medestudenten zijn die hier al gevoel voor hebben, gaat het wat betreft uitgangspunt 4 om iets dat vooral de docent inbrengt, op basis van zijn inzicht en liefst ook ervaring in de beroepspraktijk.

Samenwerking

Samenwerken in een team, of het nu een project-team of een voetbal-team is, heeft z'n eigen dynamiek. Ieder levert een bijdrage vanuit eigen rol, gebaseerd persoonlijkheid en inhoudelijk talent. Aanvallers, keepers, middenvelders, de één is niet belangrijker dan de ander, ook al "scoor" je vooral met scoren.

Ook in project-teams is er sprake van een rolverdeling. Hiervoor bestaan verschillende theoretische modellen. Een bekend voorbeeld zijn de rollen van Belbin.

https://werkenmetteamrollen.nl/uitleg-teamrollen/

Het gaat hier niet om dit theoretisch model op zich. De docent kan het noemen en geïnteresseerde studenten kunnen er iets over lezen. Het gaat er echter vooral om, de studenten de ogen te openen voor het feit dat niet iedereen dezelfde aanleg en voorkeuren heeft wat betref z'n rol in een project-team. Als je van jezelf weet waar je goed in bent, en ook kunt zien waar anderen goed in zijn, loopt de samenwerking soepeler. Niet iedereen is een voorzitter, specialist of groepswerker. Dat hoeft ook niet. Een team functioneert als beste als de rollen zo verdeeld zijn dat ze het beste bij de teamleden passen. Daarbij is een voorzitter niet belangrijker dan een specialist of bedrijfsman.

Het doorbreken van het inzicht dat ieder z'n eigen soort bijdrage levert, kan gestimuleerd worden door bijvoorbeeld de rollen van Belbin kort te bespreken. Aan de studenten kan dan gevraagd worden, in hun persoonlijke evaluatie aan te geven welke rol het dichtst kwam bij die van hen in het bank-project en dit toe te lichten. Hierover nadenken helpt later ook bij de beroepskeuze.

Ook de rollen van anderen binnen het team kunnen worden benoemd. Dit kweekt oog voor mensen en daarmee mensenkennis. Het zorgt dat conflicten beter worden begrepen en gehanteerd. Immers als je snapt *waarom* iemand iets doet, kun je hem tegemoet treden met meer begrip en minder boosheid. Dat is een belangrijke stap naar het oplossen van zulke conflicten.

Zo leren studenten inderdaad van andere studenten en krijgt uitgangspunt 4 meer inhoud. Ze leren elkaars verschillen te zien zonder daar al te snel een oordeel aan te verbinden. Wie dat heeft geleerd is een beter teamlid, ook later op het werk.

5. Studeren met eigenaarschap en verantwoordelijkheid

Het bankproject is minder eenduidig gedefinieerd dan het liftproject. Bij de lift zijn de eisen zodanig dat de oplossingsrichting op z'n minst voor de hand ligt. Vrijwel alle studenten gebruiken een stappenmotor om de lift in de buurt van het doel te brengen en daarna een fotodiode om te kijken of hij precies bij de verdieping is. Enige vrijheid is er in de semantische laag (wat de messages betekenen, niet hoe ze technisch in elkaar zitten) van het protocol waarmee de Arduino's communiceren. Ook de taakverdeling tussen centrale unit en etage-processors kan iets variëren. Wat betreft de mechanica maken de meeste studenten er een zwoegende hijskraan van in plaats van een energie-zuinige balans met een tegenwicht zoals een echte lift.

De bank biedt veel meer opties. Om te beginnen is er het mechanische gelduitgifte-systeem. Hier is ruimte voor individuele, creatieve oplossingen die het wel moeten doen, anders leidt de groepsprestatie schipbreuk: eigenaarschap en verantwoordelijkheid voor de studenten die dit deel voor hun rekening nemen. Ook de GUI van de geldautomaat biedt veel meer vrijheid tot creatieve oplossingen. Ook hier gaat het weer om keuzen met consequenties. Ook de student die zich eigenaar maakt van dit deel van het probleem heeft daarmee verantwoordelijkheid voor het totaalresultaat. Een onlogisch of onpraktisch userinterface heeft negatieve consequenties voor de beoordeling en daarmee voor de hele groep.

Bij het bankproject worden de studenten ook al meer losgelaten dan bij de lift. De begeleiding is al iets meer on-demand dan bij de "groentjes" in het liftproject. Er hoeft geen middelbare scholier meer geupgrade te worden tot een student. De meesten hebben inmiddels wel in de gaten dat er bij HR-TI eigen initiatief van je verwacht wordt en velen hebben wat dat betreft voortgang geboekt. Daarmee is er ruimte voor meer eigen verantwoordelijkheid.

Om eigenaarschap en verantwoordelijkheid bij alle projectteam-leden te bevorderen is het belangrijk dat de leden een heldere, expliciete taakverdeling overeenkomen waarbij ieder lid inderdaad eigenaar is van een onderdeel van het projectresultaat en daarmee verantwoordelijk voor het voldoen aan de requirements betreffende dat onderdeel. Deze expliciete taakverdeling speelt een rol bij de beoordeling. Iedere student hoort te kunnen aangeven wat zijn aandeel is in het geheel en bij de beoordeling ook desgevraagd in detail te kunnen vertellen hoe hij dit aandeel precies heeft ingevuld. Dit is zeker niet teveel gevraagd, wie een actief aandeel heeft geleverd kan daar over vertellen, hoeft niet in mooie woorden.

Het keuze-project

Wat is het

Het keuzeproject, op dit moment project 5/6 bij technische informatica, is een project in het tweede studiejaar, waarvan de studenen zelf het onderwerp kiezen. Ook bij dit project wordt in groepen gewerkt. Onderwerpen variëren van rolstoelen en handprothesen tot dakgoot-reinigers en laser-projectors. Elk project heeft een opdrachtgever. Soms is deze opdrachtgever extern, soms is het een docent binnen de hogeschool.

De projecten worden afgerond met een feestelijke eindmarkt, waarvoor alle studenten en docenten als bezoeker worden uitgenodigd. De bezoekers kunnen een aantal punten verdelen over de projecten. Voor de winnende projecten zijn er prijzen, bijvoorbeeld een taart.

Een uitdaging bij een deel van de keuzeprojecten is de verkrijgbaarheid van de hardware. Met het huidige chiptekort is deze uitdaging nog groter geworden. Hardware waar de studenten vertrouwd mee zijn is soms niet meer te krijgen en alternatieven moeten worden onderzocht.

Daarnaast valt de leveringstermijn regelmatig tegen. Er wordt dan een beroep gedaan op de creativiteit van de studenten. Studenten reageren hier uiteenlopend op. Sommige komen inderdaad met creatieve oplossingen. Een minderheid blijft hierin passief. Soms mislukt een project hierdoor, maar dat zijn uitzonderingen.

De samenwerking in de groepen is meestal goed, de studenten die het zover hebben weten te brengen, blijken dit nu echt wel behoorlijk geleerd te hebben. Ze kennen elkaar ook beter en de groepsindeling is vrij, waardoor er binnen de teams vaak een goede “klik” is.

Het project door de bril van de vijf uitgangspunten

1. Studeren met plezier

Het plezier van dit project begint al bij de vrijheid in onderwerpskeuze. Studenten kiezen onderwerpen waar ze vaak echt affiniteit mee hebben. Dit kan bijvoorbeeld een prothese zijn die gebruikt kan worden door een van de groepsleden of door een familielid. Of gewoon iets leuks, zoals een bolvormige 3D display, gemaakt met een boogvormige roterende kleuren-ledstrip. Of iets dat aansluit bij een persoonlijke passie, zoals een electronische beeldversterker voor een telescoop.

Sommige groepen zijn echt enthousiast over wat ze aan het doen zijn en halen alles uit de kast. De projectresultaten zijn vaak meer dan alleen functioneel, ook aan het uiterlijk wordt zorg besteed. Pittoreske maar toch wat corny ge-laser-cutte kastjes maken plaats voor ge-3D-printe kunststofbehuizingen met vloeiende lijnen. Een enkel projectresultaat is bijna rijp voor productie, maar dat zijn uitzonderingen.

De projectresultaten laten zien wat de studenten allemaal al kunnen, en dat is niet gering. Geconfronteerd met een probleemstelling weten de meesten te komen tot een min of meer functionerende technische oplossing. Al met al is er rond het eindresultaat sprake van plezier voor

studenten en docenten, een mooie mijlpaal.

De weg naar dat eindresultaat is er eentje met ups en downs. Zoals bij “echte” projecten is slagen of falen vaak een kwestie van heel veel inzet, volharding en, waar nodig, improvisatie. Een belangrijke succesfactor is de specificatie van eisen aan het projectresultaat. Het is heel eenvoudig projecten te definiëren die te hoog gegrepen zijn of waarbij het niet duidelijk is wat de opdrachtgever nu precies verwacht.

Het feit dat het om een keuzeproject gaat, vergroot het risico dat men meer afbijt dan men kan kauwen. Dat is op zich een hele nuttige ervaring, waarbij het kwartje meestal niet in één keer valt. Uit de praktijk is de factor 3 of, voor technici, π bekend. Verwijder alle toeters en bellen uit de eisen aan het eindresultaat, denk van te voren heel goed na over het ontwerp, doe je uiterste best en met heel veel geluk doe je er 3 keer zo lang over dan je gepland had.

“Het slagen of falen van een project wordt bepaald in de requirements-fase” is een bekend gezegde uit het bedrijfsleven. De missers halen soms de krant. De Fyra, de Vliegende Hollander bij de Efteling, IT-projecten bij de overheid, de reeks is eindeloos. Van studenten kan niet worden verwacht dat ze deze valkuil helemaal in hun eentje vermijden. Een beetje spanning is meestal niet erg, kan zelfs bijdragen aan het plezier. Maar keihard werken aan een complete mislukking is geen nuttige leerervaring.

Wat hieraan te doen valt is simpel: Studenten mogen hier een beetje tegen zichzelf in bescherming worden genomen. Bij de goedkeuring van het projectvoorstel is het belangrijk dat hierop wordt gelet. Sommige projecten kunnen beter in tweeën worden gehakt. Of in nog meer moten. Dit gebeurt soms achteraf: Zo ver zijn we gekomen, de rest is voor het volgende team. Indien dit echter door inbreng van de ervaring van een docent, al van te voren kan gebeuren heeft dat een belangrijk voordeel. Geleerd wordt dat projecten niet persee onbeheersbaar zijn, mits men een stukje ervaringswijsheid uit het bedrijfsleven inzet: de factor π.

Nogmaals, dit is iets waarvan steeds opnieuw blijkt dat mensen het *niet* makkelijk door ervaring alleen leren. Daarvoor verdwijnen er te veel dure “real world” projecten in de shredder. Het is goed als docenten dit hardnekkige fenomeen van budget-overschrijding bespreken en studenen handvatten geven om het te vermijden. Dit is nu typisch een geval waarbij de docent zijn eigen ervaring of inzicht inbrengt. Een docent is niet alleen een procesbegeleider, hij draagt ook vakkennis over, technisch en procesmatig.

Meestal gaat dit aspect bij het keuze-project goed. Een enkele keer was in plaats van een mislukking door begrijpelijke onderschatting van de hoevelheid werk die iets kost een positieve leerervaring mogelijk geweest. Door eigen ervaring dat je het me een factor 3 marge tot je verbazing maar net red is ook leren, maar dan met positieve bekrachtiging.

2. Studeren met nieuwsgierigheid

Een reden voor een bepaalde projectkeuze *kan* nieuwsgierigheid zijn. Gelukkig zijn veel studenten opzoek naar leerervaringen. Ze zoeken dingen waar ze nog niet vertrouwd mee zijn en gaan zulke

uitdagingen bewust aan. En natuurlijk zijn er ook heeél wat studenten die de kortste weg naar een voldoende zoeken. Een manier om te zorgen dat zij niet al te veel in hun comfort zone blijven, is om de keuze te bieden uit een vooraf samengestelde lijst van kwalitatief goede projecten. Daarbij zou zelfs bewust rekening kunnen worden gehouden me de studenten in kwestie.

Iemand die een al programmeertijger is, leert misschien weinig van een project waarbij software de boventoon voert. Er zijn nogal wat studenten met web-development ervaring, die graag iets maken waarbij een browser-interface een belangrijke rol speelt. Iemand die al sinds de middelbare school websites maakt, leert daar waarschijnlijk weinig nieuws meer over.

Soms heeft nieuwsgierigheid een zetje nodig. Een docent die de studenten goed kent, kan ze een nieuwe wereld binnenleiden. Juist zo'n nieuwe wereld kan nieuwsgierigheid wekken, als hij met een duwtje in de rug betreden wordt. Het is goed als docenten gebruik maken van wat ze van de studenten weten om ze af en toe op een nieuw spoor te zetten. Het keuzeproject is daartoe een mooie gelegenheid.

Keuze is niet persee volledig vrije keuze van de student voor wat hij al weet. Keuze kan ook betekenen: er zijn meerdere mogelijkheden en in overleg met de student wordt iets gekozen waarvan hij het meeste leert zonder overvraagd te worden. Want dat is de andere kant van de zaak. Zoals eerder gesteld bij uitgangspunt 1, studeren met plezier, is het ook belangrijk studenten niet iets te vragen wat ver boven hun macht ligt. Iemand die het verschil tussen stroom en spanning niet weet, maar het plan heeft opgevat een computergestuurde mengtafel te maken, gaat waarschijnlijk een negatieve ervaring tegemoet. Enthousiasme alleen is niet genoeg. Ook hier is het handig als de docenten weten wat voor vlees ze in de kuip hebben en daar rekening mee houden.

In het algemeen dus: Niet "one size fits all" maar passend bij kennis en vaardigheden die in een groep aanwezig zijn. Zo wordt de nieuwsgierigheid optimaal geprikkeld. Nieuwsgierigheid naar nieuwe kennis, maar vooral naar eigen kunnen.

3. Een contextrijke leeromgeving

Er zijn studenten die bewust kiezen voor een project binnen een bepaalde context. De medische context is daar een voorbeeld van. Bij kiezen vanuit de context komt de contextrijkheid vanzelf mee.

Veel studenten kiezen een echter project op technisch-inhoudelijke gronden. Ze vinden een bepaalde technologie interessant, spraakmakend en/of weten er al het één en ander vanaf. Als het een project met een externe opdrachtgever betreft, dan krijgen ze de context er als het ware bij, gewild of niet. Een opdrachtgever die hen inderdaad in die context betrekt, en niet alleen een steriele "prefab" opdracht geeft is wenselijk. Gelukkig laten bedrijven meestal graag zien waar ze mee bezig zijn, uiteindelijk staan ze immers vaak met een schepnet aan de poort.

Projecten met een interne opdrachtgever, waarbij techniek leidend was bij de keuze, lopen de grootste kans weinig contextrijk te zijn. Dit kan worden ondervangen in de opdrachtomschrijving, door deze in te bedden in een realistische casus. Iets wat ook helpt is als de docent die als opdrachtgever functioneert, in z'n rol kruipt. Dit houdt in dat hij z'n inhoudelijke deskundigheid wat opzij zet en meer denkt en spreekt vanuit de behoefte binnen het "bedrijf" aan het projectresultaat. We hebben een besturing voor onze lasrobot nodig, gebaseerd op 3D camerabeelden. Dit en dat en dat moet hij kunnen.

En zoveel mag het kosten. En dit zijn onze veiligheidsvoorschriften. We willen graag dat de camera in Europa gefabriceerd wordt. Het productieproces mag bij installatie niet langer dan 4 uur stilgelegd worden. Etc. Praktische eisen en wensen die te maken hebben met de dagelijkse gang van zaken in een gefingeerd bedrijf. Dus geen gedetailleerde adviezen betreffende de technisch-inhoudelijke manier van oplossen. Een opdrachtgever wil meestal een systeem dat werkt. De technologie is secundair.

Om dit goed te laten werken is het handig als docent-opdrachtgever en docent-begeleider gescheiden zijn. Dat kan door expliciet steeds van rol te wisselen, eventueel letterlijk met een bepaalde pet op. Makkelijker is het als het twee verschillende docenten zijn. Dit is jullie opdrachtgever en bij die en die andere docent(en) kun je terecht voor begeleiding betreffende de techniek, planning en groepsproces. In het eindoordeel hoort de tevredenheid van de opdrachtgever een grote rol te spelen.

Sommige docenten spelen van nature een overtuigende opdrachtgevers-rol. Voor anderen kost het misschien meer moeite. Het is echter in overeenstemming met de keuze die is gemaakt in het opleidingsprofiel (OTI). We willen TI'ers afleveren met oog voor de context. Dat is een doelstelling met een duidelijke maatschappelijke kant. In die zin is de hogeschool dienstbaar aan de maatschappij die haar van middelen voorziet.

4. Studeren door te leren van andere studenten en docenten

Bij voorgaande projecten waren alle studenten in principe met een variant van hetzelfde bezig. Daarom was leren van andere studenten relatief simpel: Hoe hebben jullie dit probleem opgelost? Kan ik iets dergelijks inpassen in de manier waarop onze groep bezig is. Het duidelijkst was dat bij het lift-project. De gekozen mechanische oplossingen, type en plaatsing van sensoren en zelfs algoritmen en protocollen lijken vaak als twee druppels water op elkaar.

Bij het bank-project is er al veel meer variatie, zowel bij hardware als software. Welliswaar zie je bijvoorbeeld bij de geldautomaten slechts een beperkt aantal varianten op het gelduitgave-mechanisme, maar iedere groep geeft er toch wel z'n eigen draai aan.

Bij het keuze-project zijn lijken de projecten over het algemeen totaal niet op elkaar. Dit houdt in dat de individuele groepen meer op zichzelf aangewezen zijn dan voorheen. Toch is het van elkaar leren niet beperkt tot binnen de eigen projectgroep. Het vraagt alleen minder kopiëren en meer overdracht van de kennis achter de weetjes. Dus niet "Hoe zit jullie communicatieprotocol in elkaar?", maar "Hoe maak je eigenlijk een eenduidig en toch flexibel communicatieprotocol?".

Het uiteenlopen van de projecten maakt dus generalisatie en abstractie van de overgedragen kennis noodzakelijk. Er worden geen "drop-in" oplossingen en trucjes meer geleerd, maar oplossingsstrategiën. En strategiën zijn superieur aan "drop-in" oplossingen. Immers een strategie is, met de juiste vertaling naar een concrete oplossing, toepasbaar in uiteenlopende situaties. Een "drop-in" oplossing past alleen in precies die ene open plek in die ene puzzel. Zo tilt het keuze-project ook het leren van elkaar naar een hoger niveau.

Daarnaast begint in het leren van elkaar kruisbestuiving een rol te spelen. Kruisbestuiving komt in de techniek veelvuldig voor. Wie een kolossale meebewegende aanmeersteiger voor boorplatforms van Ampelmann vergelijkt met het bescheiden PIXL meetinstrument aanboord van het Perseverance karretje dat op Mars rondrijdt, ziet een verbluffende overeenkomst. Juist het ter beschikking hebben van een arsenaal aan oplossingen uit “aanpalende” toepassingsgebieden kan leiden tot die ene innovatieve schakel in het huidige project. Geïnspireerd door een oplossing voor een heel ander probleem.

Ook bij het keuzeproject is het de moeite als de groepen van elkaar weten wat ze aan het doen zijn. Een tijd geleden was daarvoor de term “knowledge management” in zwang. De hype eromheen is verdwenen, maar het delen van kennis om elkaars gereedschapskist aan te vullen is nog steeds enorm waardevol. De waarde van innovatieve bedrijven zit ‘m in de kennis in de hoofden van de medewerkers. En in hun vaardigheid om die toe te passen in een compleet andere, soms onverwachte context.

Het positionerings-systeem van de automatische stacking container-kranen van ECT op de Maasvlakte maakt gebruik van meetlineaals met hall-elementen, bedoeld voor binnen opgesteld kleine draaibanken. Deze lineaals hangen op z’n kop onder de containerkranen en reageren op magneetjes die op de rails van de kraan zijn geplakt. Het systeem functioneert al jaren in weer en wind. Een voorbeeld van een off-label toepassing bij uitstek, creatief, innovatief en effectief.

Het keuzeproject biedt een mooie mogelijkheid om de studenten te laten kennismaken met het systematisch delen van kennis, niet met een onmiddelijk doel, maar als strategie. Iedere groep geeft aan de andere groepen een inhoudelijke, diepgaande presentatie met als onderwerp: Wat was het lastigste technische probleem dat we bij dit project tegenkwamen en hoe hebben we dit opgelost. Daarbij gaat het niet om een vaag, gepolijst praatje, maar om details, het naadje van de kous, vragen en een gesprek van technicus tot technicus. Vroeg of laat werpt de op deze manier verspreide praktische kennis z’n nut af. En laat is prima. Ze leren immers niet voor school.

5. Studeren met eigenaarschap en verantwoordelijkheid

Het zelf kiezen van een project heeft veel met eigenaarschap en verantwoordelijkheid te maken. De student eigent zich er een probleemstelling mee toe en neemt verantwoordelijkheid voor de oplossing. Bij voorgaande projecten was er geen keuzemogelijkheid, afgezien van deelsystemen binnen een project. Hier kiezen de studenten zelf. Ze gaan binnen grenzen vrijwillig commitment aan. Dat geeft een wezelijk ander perspectief. Nog verder verwijderd van het “moeten” op de middelbare school. En dichter bij het “willen” in een beroep waarin je hopelijk plezier kunt hebben.

Eigenaarschap en verantwoordelijkheid komen binnen het keuzeproject dan ook meestal meer vanzelf, er is dan sprake van intrinsieke motivatie, de mooiste motivatie die er is. Toch kan ook binnen een project waar je aanvankelijk zelf voor koos een moment aanbreken dat de lol er een beetje af is. Het is allemaal moeilijker dan je dacht. De hardware uit China blijkt volledig ongedocumenteerd te zijn.

Drivers doen het niet, dure onderdelen gaan kapot. Een mooie principe-oplossing blijkt in de praktijk om een banale reden niet te werken. Daa

Het positionerings-systeem van de automatische stacking-kranen van ECT op de Maasvlakte maakt gebruik van meetlineaals met hall-elementen, bedoeld voor binnen opgesteld kleine draaibanken. Deze lineaals hangen op z'n kop onder de containerkranen en reageren op magneetjes die op de rails van de kraan zijn geplakt. Het systeem functioneert al jaren in weer en wind. Een voorbeeld van een off-label toepassing bij uitstek, creatief, innovatief en effectief.r sta je dan.

Eigenaarschap en verantwoordelijkheid, vrijwillig aangegaan krijgen dan het karakter van aangegaan commitment, samen met je groep. Doorzettingsvermogen en frustratie-toleratie zijn onontbeerlijkom eigenaarschap en verantwoordelijkheid waar te maken. Tegenslagen kunnen leiden tot een passieve houding met aan de docent de boodschap: We komen er niet uit. U moet het oplossen. U bent docent. Echter de docent is niet de probleem-eigenaar. En de opdrachtgever heeft de opdracht, en daarmee het probleem, bij de projectgroep neergelegd. Het kan zijn dat de projectgroep er niet uitkomt. Open overleg met de opdrachtgever is dan de aangewezen weg.

Techniek is unforgiving. Het is geen schande als iets niet lukt, maar je kunt je er niet uitkletsen. Zijn er andere manieren dan wat oorspronkelijk werd gespecificeerd om het onderliggende probleem van de opdrachtgever, de vraag achter de vraag, op te lossen? Andere technische wegen? Of niet-technische misschien? Het gaat dan weer om context. Techniek was niet het doel maar het middel.

Bij de beoordeling is het belangrijk dat dergelijke alternatieve wegen om binnen de context een concreet probleem op te lossen worden gewaardeerd. Als de originele oplossingsstrategie na serieuze pogingen en variaties echt niet blijkt te werken, getuigt het van professionaliteit en oog voor het belang van de bedrijfsmatige of maatschappelijke context om een andere route te kiezen. Niet alles is nu eenmaal te voorzien. Beter ten halve gekeerd dan ten hele gedwaald.

Vak: Imperatief Programmeren

Wat is het

Veel studenten hebben wel eens iets met programmeren gedaan. Regelmatig wordt de combinatie HTML/CSS genoemd, een combinatie die tamelijk ver verwijderd is van wat gewoonlijk onder programmeren wordt verstaan. JavaScript wordt ook genoemd. Dat komt al wat dichter bij maar vermoedelijk gaat het in de meeste gevallen om een paar regels code om een webpagina tot leven te wekken.

Ongeveer een derde van de studenten heeft wel eens iets gedaan met C op een Arduino, soms als hobby, soms in het kader van een het vak Informatica op de middelbare school.

Een kleine minderheid heeft echt wat ervaring. Vaak wordt hierbij C++ genoemd, waarschijnlijk omdat het op de Arduino draait. Een enkeling kan met deze taal al heel veel doen, inclusief gebruik van zaken als dynamisch geheugenbeheer en polymorphism.

Ondanks dit alles is het vak Imperatief Programmeren voor een aanzienlijk deel van de studenten de eerste echte kennismaking met het maken van computerprogramma's die uit meer bestaan dan een paar regels code.

Het vak Imperatief Programmeren behelst een eerste kennismaking met de programmeertaal C. Voordelen van C als eerste taal zijn dat het dicht bij de machine staat en daardoor inzicht biedt in wat er "precies gebeurt" als je een computerprogramma draait. Daarnaast is het een gecompileerde taal, de compiler tikt je op de vingers als je iets raars doet met datatypen of parameterlijsten. Echter in dat opzicht is C "less than ideal". Er zijn weinig talen die zoveel dubieuze vrijheid bieden aan de programmeur en waarvan de foutmeldinge zo cryptisch zijn. Dat heeft te maken met de syntax van de taal, die volledig gericht is op "snappen door de computer" en niet op "snappen door de mens".

Daarnaast bevat de taal een aantal zaken die voor veel studenten te hoog gegrepen blijken te zijn bij een eerste kennismaking met programmeren. Ondanks aandacht voor address- en dereference-operators (* en &) blijkt dit concept bij vrijwel alle studenten compleet van de radar te zijn verdwenen op het moment dat in het tweede studiejaar C++ aan de orde komt. Mochten ze het aanvankelijk al echt begrepen hebben dan heeft het zich blijkbaar niet als relevante kennis gehecht aan wat ze al wisten.

Het vak wordt afgerond met een wat grotere opdracht, waarbij een ASCII art plaatje (Lena) op het scherm moet worden getoverd. Dit is voor de meesten een hele toer.

Op dit moment is wordt binnen TI opnieuw gekeken of de keuze en volgorde van aanleren van programmeertalen voor verbetering vatbaar is. Reden te meer om ook dit vak eens tegen het ligt te houden aan de hand van de vijf uitgangspunten.

Het vak door de bril van de vijf uitgangspunten

1. Studeren met plezier

Een deel van de studenten heeft al iets met programmeren, en dit vak valt bij hen in vruchtbare aarde. Maar bij velen lijkt van plezier geen sprake. Ze vinden het vak moeilijk. Fundamentele concepten landen moeizaam en worden soms vaak door elkaar gegooid. Zo blijkt het verschil tussen een datatype en een variabele voor velen ook na afloop nog steeds lastig.

De echte problemen beginnen als ze er constructief mee aan de slag moeten in een wat groter programma. Velen hebben geen idee hoe ze dit moeten aanpakken. Ze voelen zich in het bos achtergelaten zonder kompas. Hoe begin je eigenlijk. En hoe ga je dan verder.

Een heel duidelijk fenomeen is, dat studenten heel veel bezig zijn met details in de code en maar totaal geen grote lijn in hun hoofd hebben. Ze worstelen met de syntax op statement-niveau zonder toe te komen aan de vraag hoe het programma in principe zou moeten werken.

Hierin kan op eenvoudige wijze sterke verbetering worden gebracht. Het recept is simpel. Aan het begin van de broncode komen drie blokken commentaar, namelijk *Requirements*, *Testspec* en *Design*, elk van een regel of twintig.

Bij *Requirements* beschrijft de student in eigen woorden *wat* het programma precies moet doen, zonder zich druk te maken over *hoe* het programma dit doet. Deze beschrijving is zo gedetailleerd en concreet mogelijk. Door dit te doen maakt de student zich de opdracht actief eigen. Alleen maar doorlezen werkt voor velen niet. Gebrekkige leesvaardigheid is daarbij een factor. Lezen en begrijpen tegelijk blijkt moeilijk. De opdracht wordt meer vluchtig gescanned dan grondig gelezen. Daardoor dringt een student er niet werkelijk in door maar blijft aan de buitenkant hangen. Zelf opschrijven blijkt enorm te helpen.

Bij *Testspec* beschrijft de student hoe hij uiteindelijk gaat testen of de applicatie werkelijk aan de requirements voldoet. Deze beschrijving is operationeel, dat wil zeggen hij beschrijft exact welke handelingen worden verricht tijdens het testen en wat daarvan de resultaten zijn. Ook is hij kwantitatief, vage waarde-oordelen zoals *vaak, betrouwbaar, robuust, flexibel* hebben daarin geen plaats. Het moet objectief en precies duidelijk zijn wanneer wel of niet aan de testcriteria is voldaan. Dus een bepaalde handeling moet bijvoorbeeld achtereenvolgens 3x met succes kunnen worden gedaan met 3 van tevoren gespecificeerde sets invoer-gegevens. Of 10x. In de praktijk worden dergelijke getallen met een opdrachtgever overeengekomen. Zij bepalen hoe betrouwbaar de test is. Bij dit eerstejaars vak is de situatie relatief simpel en speelt het kwantatieve niet zo'n rol. Echter de operationaliteit van de test wel. Iedereen moet hem aan de hand van de *Testspec* kunnen uitvoeren. En objectief kunnen vaststellen of de test geslaagd is.

Het tekstblok met de kop *Design*, tenslotte, is bij deze eerste kennismaking met programmeren het belangrijkste. Het gaat daarbij niet om diagrammen. Student, beschrijf nu eens hoe je denkt dat je programma moet gaan werken zonder je daarbij te verliezen in de details van een nog relatief onbekende programmeertaal. Doe maar net alsof je aan je tante Betje uitlegt hoe je dit gaat aanpakken. Stap voor stap.

Bij de huidige eindopdracht (Lena in ASCII-art):

> N.B. Bij het *Design* is er vanuit gegaan dat de datafile iets handiger in elkaar zit dan op dit moment.
>
> N.B.2 Hoewel de tekst hier in het nederlands is gegeven, verdient engels voor *Requirements*, *Testspec* en D*esign* sterk de voorkeur als de student hiertoe in staat is. Taalfouten zijn daarbij van ondergeschikt belang. Ook de programmering gebeurt bij voorkeur in het engels. Hiermee kan niet vroeg genoeg geoefend worden. Bij bedrijven wordt bijna altijd in het engels geprogrammeerd, en vragen aan de programming community, bijvoorbeeld op Stack Overflow, met nederlandstalige voorbeeldcode zijn betrekkelijk kansloos. Het deel van de wereldbevolking dat nederlands spreekt is nu eenmaal zeer klein.

Design

De gegevens van het Lena-plaatje staan in een bestand. De getallen in dit bestand staan in tekstvorm en stellen helderheden voor. Ze lopen van 0 t/m 255. De eerste twee getallen in het bestand zijn het aantal rijen en kolommen van het beeld.

Allereerst wordt een tweedimensionaal (short unsigned) integer array gedefinieerd met deze afmetingen. In twee geneste for-lussen worden de gegevens uit het Lena-bestand in dit array ingelezen.

Vervolgens wordt een helderheidsschaal gedefinieerd. De getallen van 0 t/m 255 worden daarbij gekoppeld aan bijvoorbeeld 8 letters met opklimmende helderheid bij witte letters op een zwart scherm, bijvoorbeeld ".,-=ILNM". Als je een bepaald helderheids-getal hebt en de letters zijn genummerd van 0 t/m 7 (dit zijn de indices van de letters in de string ".,-=ILNM") dan kun je de index van de bijbehorende letter als volgt vinden:

index = helderheid / 32

Vervolgens wordt in twee geneste for-lussen het genoemde tweedimensionale array regel voor regel, en daarbinnen de helderheden getal voor getal doorlopen. Elk helderheid wordt aan de hand van bovenstaande formule omgezet in een index. De letter met die index wordt op het scherm gezet. Elke keer als de binnenste lus geheel doorlopen is, is er een regel klaar. Op dat moment wordt steeds een newline ('\n') naar het scherm geschreven, waardoor op het scherm naar de volgende regel wordt gesprongen.

(En zo verder)

Bij deze werkwijze krijgt de student gelegenheid na te denken over een oplossinsstrategie, zonder daarbij enorm gehinderd te worden door de ballast van een programmeertaal. Als het gelukt is om op deze manier te beschrijven *wat* er precies moet gebeuren, worden alle tussenstappen in de vorm van commentaarblokjes in het programma gekopieerd (originele commentaarblokken ook laen staan) en eventueel wat gedetailleerder beschreven. Pas daarna begint het eigenlijke coderen, elk codefragment zoveel mogelijk direct onder het betreffende commentaarblokje.

Deze werkwijze, afgeleid van Donald Knuth's "Literate Programming" blijkt beginnende programmeurs enorm te helpen. Als de commentaarblokjes er eenmaal staan, ligt de grote lijn vast en kunnen de studenten zich focussen op de details en de uitdaging van de programmeertaal, zonder het

risico te lopen die grote lijn uit het oog te verliezen. Deze “comments first” manier is ook voor zeer ervaren programmeurs een hele prettige werkwijze. Immers de code is daarmee gelijk grondig gedocumenteerd, niet in de vorm van triviaal, obligaat afgeraffeld commentaar, maar op een doordachte, functionele manier waar niet alleen toekomstige collega’s plezier van hebben maar, ook de originele programmeur zelf, aangezien het vantevoren gebeurt. Coderen met beheersing en overzicht, en daardoor met plezier, is bij Techische Informatica een belangrijk onderdeel van studeren met plezier.

2. Studeren met nieuwsgierigheid

Studenten al eerder hebben geprogrammeerd zijn nieuwsgierig naar wat ze bij TI wat dit betreft allemaal voor nieuws gaan leren.

Studenten die nog nooit hebben geprogrammeerd zijn nieuwsgierig naar wat programmeren nu eigenlijk inhoudt.

Het verschil in startniveau is groot. Bij de huidige opbouw van het vak komen beide groepen te kort. Studenten die al eerder hebben geprogrammeerd zijn vaak teleurgesteld in dit vak. Ze hoopten dat nu “het echte werk” zou gaan beginnen. Het blijkt dat ze bij het maken van de opgaven onder een juk door moeten waar ze met gemak wijdbeens overheen zouden kunnen springen. En dat bij één van de eerste vakken die ze op de Hogeschool krijgen.

Studenten voor wie programmeren nieuw lopen tegen een andere teleurstelling aan. Programmeren is moeilijker dan ze dachten. Je moet heel veel van de taal snappen om eenvoudigweg een gegeven van het toetsenbord te kunnen inlezen. Hoezo precies *scanf (“%f”, &x)* ? Waar zijn % en & precies goed voor? En waaarom die & niet bij *printf (“%f”, x)* ? Geen eenvoudige vragen als je net leert programmeren. *x = input ()* oogt een stuk minder intimiderend, net als *print (x)*. Natuurlijk zal een TI’er uiteindelijk een keer C(++) moeten leren. De vraag is of C z’n eerste kennismaking met dit nog onbekende vak moet zijn.

Studeren met nieuwsgierigheid komt neer op leren door nieuwsgierigheid. Dat is zoals een kind leert, door dingen gewoon uit te proberen en te ontdekken wat er gebeurt. De programmeertaal Python leent zich hier veel beter voor dan C. Plezier is niet het enige aspect dat moet worden meegenomen bij de keuze van een eerste programmeertaal. Echter wel een belangrijk aspect. Je kunt iemand maar één keer goed tegen maken.

Het niveauverschil is een probleem op zich, ongeacht de taalkeuze. Wat voor taal je ook kiest, een bepaalde opgave zal voor sommigen te makkelijk zijn en voor anderen te moeilijk. Hiervoor is gelukkig een trefzekere remedie: gedifferentieerd lesmateriaal. Ja, het kost extra werk om dat te maken, maar ook weer niet zoveel werk.

Als alleen al de opgaven gedifferentieerd zouden zijn, bijvoorbeeld in drie niveaus, zouden ook de wat meer ervaren programmeurs uitgedaagd worden. Het bovenste niveau mag pittig zijn. Juist ervaren programmeurs hebben hun kennis waarschijnlijk zelf opgedaan omdat ze geboeid waren door het fenomeen programmeren. Ze zijn van nature nieuwsgierig naar nieuwe mogelijkheden en kunnen dingen over het algemeen goed zelf uitzoeken met behulp van Internet. Aan hen tegemoet komen is

niet zo’n kunst. Behalve de opgaven hoeft het materiaal niet persee te worden aangepast, ze vinden hun Behalve de opgaven hoeft het materiaal niet persee te worden aangepast, ze vinden hun weg wel.weg wel. Een uitdagende opgave op niveau 3 zet ze aan het zoeken, de rest gaan vanzelf, uit nieuwsgierigheid.

Het laagste niveau is ook interessant. Horen pointers en adres-operatoren werkelijk in een allereerste kennismaking met programmeren thuis? Of kan dat ook later aan de orde komen, bijvoorbeeld bij C++, gesteld dat C de eerste taal blijft die ze leren?

Er is bij deze eerste kennismaking met programmeren duidelijk ruimte voor verbetering. Die wordt op dit moment onderzocht. Studeren met nieuwsgierigheid is daarbij één van de brillen waardoor zou moeten worden gekeken, als we het profiel-document (OTI) serieus nemen.

3. Een contextrijke leeromgeving

In het eerst jaar van HR-TI is voor het vak Imperatief Programmeren een duidelijke context aanwezig: Gebruik van C op de Arduino in de beschreven projecten. De C kennis die wordt opgdaan in dit vak is noodzakelijk om de projecten te kunnen doen. Daarmee is ook verzekerd dat het deel van de studenten dat bijvoorbeeld bij het lift-project de software schrijft, deze taal actief gebruikt. En actief gebruik van een programmeertaal is de enige manier om hem werkelijk te leren.

Programmeren leer je door het te doen, net als fietsen. De theorie is zeker belangrijk, maar die krijgt voor de meeste studenten pas betekenis in de context van praktische toepassing. Tot dat moment blijft programmeren een abstract spel. Pas als je ontdekt wat je er mee kunt doen wordt programmeren voor de meesten echt leuk. Veel van onze studenten hebben een fascinatie voor het *maken* van dingen, we profileren ons niet voor niets als een maak-opleiding en dat zijn we ook.

Ondanks het feit dat belangstelling voor de context, “wat je er in de praktijk mee kunt doen” , studenten door het vak heen kan trekken, is tevens een systematische theoretische opbouw nodig. Immers we willen solide makers afleveren, geen hap snap knutselaars. Dat de theorie het beste opgehangen kan worden aan de praktijk sluit gelukkig systematische behandeling niet uit.

Een populaire manier om een roman of filmscenario te schrijven is de volgende. Begin een aantal verhaallijnen rondt verschillende personen en laat uiteindelijk deze steeds meer samenkomen tot uiteindlijk aan het slot alle puzzelstukjes in elkaar passen. De oefenopaven bij het aanleren van een programmeertaal kunnen op een dergelijke manier in elkaar passen. Dit kan als volgt worden gerealiseerd:

1. Inventariseer de leerdoelen. Werk deze vervolgens zo concreet mogelijk uit. Welke taalconstructies moeten de studenten aan het einde beheersen en tot op welk niveau? Hoeveel creativiteit moeten ze al aan de dag kunnen leggen in het ontdekken van wat ze in een gegeven situatie nodig hebben?

2. Verzin vervolgens een aansprekende eindopdracht waarin al deze constructies aan de orde komen. Het is de moeite waard hier goed over na te denken. Belangrijk is dat de eindopdracht geen abstract programmeer-”kunstje” is maar verbinding heeft met een voor de studenten zo

aantrekkelijk mogelijke context. Gaming ligt voor de hand, besturing van hardware is echter ook iets wat voor velen behoorlijk nieuw en boeiend is. Er zijn ook prima combinaties tussen deze twee te maken.

3. Realiseer die van eindopdracht vantevoren minimalistisch als modeluitwerking voor de docenten. Dit is belangrijk omdat het de docenten die zich enigzins in beginners kunnen inleven, vantevoren een beeld geeft van de moeilijkheidsgraad en benodigde effort. Zoek het niet te veel in de toeters en de bellen, dat is voor programmeurs zowiezo een slechte gewoonte. Een opdracht met een harde, intrinsiek boeiende “kern” is beter. Graphics zijn in dit stadium nog te hoog gegrepen. Gebruik van sensoren en bewegende hardware is wel haalbaar en kan net zo boeiend zijn. De hardware mag worden “voorgekauwd”, het is immers geen project maar een programmeervak.
4. Pluk het resulterende programma uiteen in onderdelen, één onderdeel per les. Omdat het aantal lessen beperkt is, kunnen in één onderdeel soms meerdere leerdoelen aan de orde komen. Wees hierin echter niet te ambitieus. Als de lessen hierdoor “overvol” raken is het vak blijkbaar te groot voor het aantal beschikbare lessen. You can only do so much. Alles wat meer is dan dat wordt gewoon niet geabsorbeerd of, blijkens ervaringen, snel weer vergeten.
5. Demonstreer vantevoren de modeluitwerking van de eindopdracht aan de studenten, gewoon als teaser, zonder op de code in te gaan of deze zelfs maar te tonen. Om dit goed te laten werken is het inderdaad nodig dat het resultaat voor de studenten zo aansprekend mogelijk is. Iets dat op de Arduino draait en hardware aanstuurt is wel haalbaar. De eindopdracht mag redelijk moeilijk zijn, omdat ze hem niet helemaal zelfstandig gaan realiseren. Ze worden stap voor stap naar het eindresultaat geleid, dat daardoor veelal meer is dan waar ze zichzelf oorspronkelijk toe in staat achtten. Dit geeft natuurlijk een kick! Heb ik dat gemaakt? Het stap voor stap proces is een leerpunt op zich. Een samenstel van betrekkelijk simpele onderdelen kan uiteindelijk iets boeiends of bruikbaars opleveren. En een bij een demo complex lijkende applicatie is blijkbaar terug te brengen tot behapbare brokken.
6. Geef duidelijke, bindende richtlijnen voor de code van de deelopdrachten. Bijvoorbeeld: “gebruik hierbij een 2D integer array” of “maak hiervoor een aparte functie die deze waarden aan de caller teruggeeft door middel van pointer parameters” of “stop die en die zaken in aparte modules met bijbehorende headers”. Zo worden studenten ertoe geleid ook werkelijk de diverse taalelementen te gebruiken die ze volgens de leerdoelen moeten beheersen. Een werkende applicatie is immers een middel, geen doel. Gebruik de contacturen zoveel mogelijk voor individuele begeleiding gericht op gebruik van het juiste taalmiddel voor een bepaald doel.
7. Laat de studenten in de lessen en bij het huiswerk stap voor stap de programma-onderdelen realiseren. Sommige onderdelen worden misschien al geïntegreerd, andere blijven nog los. Sommigen omvatten misschien één of meer modules. Met module wordt bedoeld: een interface (declaraties, in C voornamelijk functie signatures) in een .h file en de bijbehorende implementatie (definities, in C voornamelijk functie bodies) in een .c file. Het feit dat je zo’n module dan bij de eindopdracht ongewijzigd kunt hergebruiken is een heel nuttige leerervaring. Wie alle declaraties bij elkaar in één header stopt, vergeet niet snel meer waarom dat onhandig

is. Stel het begrip unit-testen aan de orde en laat ze dit (op eenvoudige wijze) ook echt doen en de test-code bewaren voor latere regGraphics zijn in dit stadium nog te hoog gegrepen. resssie-tests.

8. Laat ze in de laatste les(sen) werken aan de eindopdracht en biedt ook hier individuele begeleiding. Zelfstandig ontdekken is zeker nuttig, maar overdracht van vakmanschap van de docent kan het leerproces zeer versnellen en verdiepen. Stel het begrip integratie-testen aan de orde en laat indien er problemen met de eerder gemaakte onderdelen opduiken zien dat de unit-tests nu als regressie-tests kunnen worden gebruikt. Dit hoeft allemaal niet zeer uitgebreid, maar er is geen enkele reden om ze niet gelijk het hele plaatje te laten zien. Indien deze systematische, gedisciplineerde werkwijze steeds weer opduikt, maken de studenten zich deze op den duur vanzelf eigen.

Samenvattend: Bovenstaand wordt een beproefde methode geschetst om de theorie systematisch aan de orde te laten komen. Daartoe worden een boeiende eindopdracht bedacht waarin alle elementen van de theorie practisch worden gebruikt. Deze eindopdracht wordt daarna uiteengerafeld in deelopdrachten. In de les waarin een bepaalde deelopdracht wordt verstrekt, wordt ook de bijbehorende theorie behandeld. Er wordt dus niets ingeleverd op inhoud, alleen de voorbeelden leiden tot een samenhangend geheel. Ook de volgorde van behandelen van de theoretische onderwerpen, zoals taalconstructies of meer abstracte zaken zoals modularisering, kan grotendeels vrij worden gekozen indien er niet te veel afhankelijkheden tussen de “puzzelstukken” zijn waaruit de eindopracht wordt opgebouwd. Met wat “clever thinking” zijn zulke afhankelijkheiden meestal grotendeels te vermijden. Daarmee is het eindresultaat tevens een goed voorbeeld van een programmeerstragie die streeft naar het ontkoppelen van onderdelen. (Google: Coupling and Cohesion)

4. Studeren door te leren van andere studenten en docenten

Op dit moment wordt bij Imperatief Programmeren al gebruik gemaakt van les-video’s. Daardoor is er in de lessen meer tijd voor interactie tussen individuele studenten en de docent. Er ligt een periode achter ons waarin door de Corona pandemie veel lessen on-line plaatsvonden. Dat is fnuikend voor elke vorm van interactie. Soms kwam lesgeven neer op het aanpraten tegen een scherm met zwijgende logo’s. Soms bleken mensen bereid hun camera aan te zetten, maar dat is nog iets heel anders dan rondlopen, over de schouder meekijken en contact maken.

Bij een fysieke les is meestal onmiddelijk duidelijk wie liever niet gezien wordt omdat hij onzeker is of gestrand. Alleen al de plek waar een bepaalde groep studenten keer op keer in het lokaal gaat zitten geeft een ervaren docent veel informatie. Ook interactie tussen groepjes studenten is direct waarneembaar. Fysiek aanwezigheid is juist bij programmeren enorm belangrijk. Mensen die dingen niet snappen zijn geneigd zich op subtiele wijze terug te trekken, om niet te laten merken dat ze vast zitten. Immers, dan ben je kwetsbaar voor veroordeling door medestudenten en door de docent. En kwetsbaarheid staat onder een deel van de jongeren gelijk met zwakte. Voor het is “je kaarten tegen de borst houden” de natuurlijk reactie.

Punt één bij het faciliteren van leren van andere studenten en docenten is daarom het creëren van een veilige sfeer. Dit begint bij vriendelijkheid, benaderbaarheid en betrokkenheid van de docent bij

studenten op alle niveaus. Sarcastisch is hij nooit, ironisch slechts met mate. Hij zegt niet dat dingen makkelijk zijn. Want als dingen makkelijk zijn en ik snap ze toch niet, ben ik dus dom. Eerder laat zijn eigen moeilijkheden op de weg naar het zich eigen maken van de lesstof zien. Immers ook een geboren docent is niet als docent geboren.

Punt twee betreft de omgangsvormen tussen studenten onderling. Domme vragen bestaan niet, alleené nieuwsgierig studenten die iets nog niet weten wat jij al wel weet. Aan jou de taak om de ander erbij te halen. Dat is de norm, en die mag expliciet en duidelijk worden neergezet. Als docent ben je een rolmodel. Als jij zegt dat dit de norm is, dan heeft dat betekenis, ook al wordt er niet onmiddelijk positief op gereageerd.

Naast het stellen van deze algemene norm kunnen er werkvormen worden ingezet waarbij het “leren van elkaar” wordt bevorderd. Deze werkvormen variëren van werken in duo’s (pair programming) of groepjes tot sysmatisch gepland reviewen van elkaars code. Een ander manier is desk checking: Je code aan een andere student uitleggen. Al uitleggend ontdek je fouten in je eigen redenering. Als aanvulling kan de ander ook fouten in jouw redening ontdekken. En hij kan van jouw aanpak leren. Allemaal werkvormen om het leren van elkaar te bevorderen. Zet ze in.

5. Studeren met eigenaarschap en verantwoordelijkheid

Het gebruik van lesvideos bevordert eigenaarschap en verantwoordelijkheid. De student bepaald wanneer en hoe lang hij kijkt, of hij ook actief oefent, bepaalde delen nog eens opnieuw bekijkt, of hij voorafgaand aan een les kijkt en zo vragen kan stellen over de stof of pas achteraf, wat een stuk minder effectief is. Allemaal zaken die eigenaarschap en verantwoordelijkheid in de hand werken.

Het kan echter geen kwaad enige discipline wat dit betreft te stimuleren, gewoon door uit te leggen dat je meer aan een les hebt als je vantevoren de video hebt bekeken. Natuurlijk kunnen ze dat ook zelf bedenken, maar uitstelgedrag is bijna niemand vreemd.

Het gaat hierbij weer om contact en individuele aandacht, gezien worden door de docent. Het kan hem iets schelen of ik dit vak haal. Dit is niet strijdig met eigenaarschap. Ook in een beroepssituatie willen mensen graag gezien worden. Ze hoeven niet constant geprezen te worden, maar weinig dingen zijn zo demotiverend als gebrek aan waardering. Uiteindelijk wordt die waardering steeds meer geïnternaliseerd, maar we blijven groepsdieren, mede gericht op de ander en onze plaats in de groep.

Eigenaar ben je trouwens niet alleen van een probleem, maar ook van de kwaliteit een door jou aangedragen oplossing. Altijd zijn er studenten die met heel mooie, uitgewerkte oplossingen komen, die veel verder gaan dan de opdracht. Ze hebben plezier in eigen kunnen. Een aspect daarbij is dat ze dit graag aan anderen laten zien. Maar ook kunnen ze gewoon intrinsiek plezier hebben in eigen vaardigheden. Dit is eigenaarschap op z’n leukst: plezier hebben in wat je kunt. En je verantwoordelijkheid altijd waarmaken, ook als niemand de kwaliteit van dat specifieke stukje codee ziet. Zo ben je betrouwbaar voor jezelf en anderen.

Vak: Computersystemen en Logica

Wat is het

Het vak Computersystemen en Logica is bevat een lading die de afgelopen jaren onder verschillende vlaggen is aangeboden. Logica was aanvankelijk een apart vak, dat qua inhoud op universitair niveau stond. Verscheidene systemen voor formele logica en notatie daarvan kwamen aan de orde. Hiermee vormde het vak een zeer solide basis voor tal van gedachten-gereedschappen in de IT, zoals correctheidsbewijzen en denotational semantics. De meeste studenten vonden het een moeilijk vak. Een enkele wiskundig getalenteerde student scoorde een tien.

De laatste jaren waren er binnen de opleiding vrijwel geen practische toepassingen binnen andere vakken van de genoemde formele benadering van logica. Het schrijven van een logical expression in disjunctieve of conjunctieve normaalvorm is bij real world software zeker af en toe verhelderend en nuttig, maar op de opleiding zijn zulke expressions meestal zo simpel dat de student het met z'n intuïtie af kan. Of denkt te kunnen, want zaken zoals het veelvuldig gebruikte *if (isSorted (nameList) == false)* in plaats van *if (not isSorted (nameList))* getuigen van een op z'n gunstigst wankel begrip. Ook het theorema van de Morgan komt in de praktijk regelmatig van pas. Maar veel verder gaat de toepassing van logica binnen het huidige TI curriculum niet.

Vanweg dit gebrek aan vervolg op het originele, uitgebreide vak Logica is daar een aantal jaren geleden flink in gesnoeid. Logica vormt de ultiem basis van de wiskunde en die wiskunde als geheel wordt de afgelopen jaren ook steeds meer selectief, gericht op de TI context behandeld. Numerieke methoden, uitvoerbaar op een computer, zijn daarbij vaak belangrijker dan analytische, uitvoerbaar met pen en papier. De verchuivende inhoud van de vakken wordt in dit document als gegeven beschouwd en de focus ligt op hoe die inhoud kan worden overgebracht op een manier die recht doet aan de vijf uitgangspunten.

Dat het vak over zowel over computersystemen als logica gaat heeft enerzijds historische wortels. Het is het resultaat van een samenvoeging van twee bestaande vakken over die onderwerpen. Anderzijds is heeft die samenvoeging met een goed reden plaatsgevonden. Elementair begrip van het gedrag van combinaties van logische bouwstenen is noodzakelijk om de elementen van computersystemen, zoals de Arithmetic and Logic Unit (ALU) te kunnen begrijpen. Die zijn immers uit zulke bouwstenen opgebouwd.

Met deze inhoud neemt Computersystemen en Logica in de opleiding een belangrijke plaats in. Daar waar INF'ers de hardware als gegeven beschouwen en er mee werken door de abstraherende bril van een hogere programmeertaal, worden TI'ers in hun beroepspraktijk geacht "onder de motorkap" te kunnen kijken. Ze moeten weten hoe de zaak hardwarematig echt in elkaar zit omdat ze direct met de betreffende chips te maken hebben. Een TI'er weet precies welke chip er op z'n single board computer zit, hoeveel stroom hij verbruikt en wat hij wel of niet kan. Een INF'er heeft daar meestal geen boodschap aan. Dieper dan het feit dat het om een Von Neumann machine (gescheiden programma en data) of bijvoorbeeld een Massive Parallel Dataflow machine (bijv. GPGPU) betreft, hoeft zijn kennis

niet te gaan. Zelfs het onderscheid tussen die twee wordt in een taal als C++, Python of Fortran teruggebracht tot de keuze van een backend voor een bepaalde numerieke library.

Bij de opdrachten binnen het vak wordt de taal C gebruikt. Dit zou niet persee zo hoeven zijn, maar past bij het feit dat C ook in het eerste jaar wordt aangeleerd. Ooit gebeurde dat zelfs uitsluitend binnen het vak Computersystemen, maar dat was een te grote brok voor de eerstejaars voor wie beiden nieuw waren. Daarom zijn dit twee aparte vakken geworden.

Het vak door de bril van de vijf uitgangspunten

1. Studeren met plezier

TI studenten behoren veelal tot de mensen voor wie een apparaat zonder kastje eromheen boeiender is dan een gesloten doos met een gelikt uiterlijk en een logo. Ze hebben er plezier in de techniek tot in detail te snappen. Dit geeft een gevoel van empowerment: Desnoods zou ik uit een bak transistors en weerstanden zelf een computer kunnen maken. In een wereld doordrongen van techniek zijn zij de tovernaars.

Het vak Computersystemen en Logica is het eerste vak bij HR-TI dat de studenten binnenleidt in de wondere binnenwereld van de computer. Dit is waar velen voor gekomen zijn. Plezier komt dan voor een groot deel vanzelf. Na wat zoeken en proberen is het vak de afgelopen jaren ook sterk verbeterd. Er zijn goede slides en leuke opgaven. Van een overvol struikelblok met veel theorie is het geworden tot een moeilijk, want nieuw, maar doenlijk vak met een praktische inslag en leuke opdrachten.

De naam van het vak maakt een brede lading mogelijk en wat dat betreft valt er nog veel te kiezen en te optimaliseren bij de inrichting van dit vak. Hoe zouden besturingssystemen aan de orde moeten komen? Moet het voornamelijk gaan over bestaande besturingssystemen zoals Windows en Linux en of ook over eenvoudige monitor-programma’s en loaders, zoals die op de Arduino en soortgelijke mini-boards draaien. En komt vooral de API aan de orde en een eventueel CLI (command line interface) of gaat het juist om de internals: scheduling, memory allocation en file systems. Dat laatste is vermoedelijk voor de meesten boeiender maar ook moeilijker. Wel blijkt het zich te lenen voor creatieve, uitdagende opgaven.

Moet bij het zelf maken van dingen het accent liggen op best practices, waarbij de studenten zich dus eerst moeten verdiepen in hoe bijvoorbeeld een “echte” scheduler precies werkt, of is juist de bedoeling dat ze al experimenterend zaken vooral zelf ontdekken. Enige kennis van hoe dergelijke dingen bij echte besturingssysstemen werken zorgt dat ze veel sneller “op niveau” zijn. Zelf het wiel uitvinden is misschien heel leuk voor de studenten, maar niet persee de kortste weg naar het doel van de opleiding: afleveren van competente technici.

De hardware en het besturingssysteem zijn allebei vooral een stuwmeer van praktische know how. Achter het laatste zitten dan vaak wel weer fundamentele, algemeen bruikbare algoritmen zoals beschreven in het vermaarde “The Art of Computer Programming” van Donald Knuth of in “Operating Systems” van Andrew Tanenbaum, een andere klassieker.

Gebruik makend van zulke standaardwerken kan voor sommige studenten verdiepingsstof worden geboden. Zo kan bijvoorbeeld een memory allocator een betrekkelijk eenvoudige “blokjesschuiver” zijn, maar ook een geavanceerd algoritme met suballocatie, virtual addressing, paging en shared memory.

Een andere mogelijkheid om het plezier te vergroten zou een groepsproject kunnen zijn, waarbij de studenten uit losse logische bouwstenen en met gebruikmaking van C een volledig functioneel computerssysteem bouwen. C kan worden gebruikt als de gemaakte hardware dezelfde instructieset heeft als het target van een bestaande compiler. Om te zorgen dat dit alles kans van slagen heeft, is dit wel een oefening die eerst door de docenten een keer tot een goed einde moet worden gebracht, waarna uit die ervaring aan de studenten hints kunnen worden gegeven betreffende welke hard- en software bouwblokken nodig zijn en hoe de taken die nodig zijn voor het maken van deze blokken het beste over de studenten kunnen worden verdeeld. Als uitgangspunt kan de opbouw een eenvoudige bestaande 8 bits processor met een regelmatige architectuur worden genomen, zoals de Z80 met z’n mooie, regelmatige instructieset. Emulatie in software kan natuurlijk ook maar is minder “echt” en aansprekend.

Een aantal van deze zaken zijn de afgelopen jaren binnen het team in het kader van andere vakken ook wel eens in de één of andere vorm gedaan en er is op dit moment voldoende creativiteit en kennis in het docententeam aanwezig om dit op te zetten en te ondersteunen. Samenwerking tussen meerdere expert-groepen binnen TI is hier een goede mogelijkheid.

2. Studeren met nieuwsgierigheid

Het grootste deel van de studenten is van nature nieuwsgierig naar de interne werking van de computer. Dat is precies waar ze voor komen. Deze nieuwsgierigheid betreft juist de hardware. Er zijn er nog niet heel veel die voorafgaand aan de studie wel eens een computerprogramma van enige omvang hebben geschreven. Maar velen hebben wel eens een computersysteem samengesteld uit diverse hardware-onderdelen. Dat kan zijn op apparaat-niveau, maar ook heel goed op board-niveau binnen de systeemkast. Vooral snelle game-computers zijn populair. De kennis van wat je daar voor nodig hebt aan chips en boards is bij velen up-to-date en superieur aan de kennis van docenten.

Echter, diep gaat die kennis meestal niet. Fabrikaten, typenummers van processoren, klokfrequenties, hoeveelheden geheugen en soorten videokaarten worden onderling besproken, maar hoe een processor überhaupt in elkaar zit en welke factoren bepalen dat bij gelijke klokfrequentie toch een betere prestatie wordt verkregen is meestal onbekend.

Hier heeft het vak Computersystem en Logica veel toe te voegen. Door een basiskennis aan te reiken van architecturen, kunnen de studenten snappen wat het betekent als er sprake is van caching, instructie-prefetch, vectorisering en peephole-optimizing van instructie reeksen. Die geavanceerde zaken hoeven in dit vak niet uitputtend te worden behandeld. Maar een enkele hint naar wat nu precies een processor zo geschikt maakt voor snelle games kan, in aanvulling op de basiskennis die het vak

biedt, de nieuwsgierigheid van de studenten verder aanwakkeren. Zo wordt de belangstelling voor gaming gebruikt als “trekker” om de nieuwsgierigheid naar de internals van een computer te bevorderen.

Gaming is maar één aspect. Veel studenten zijn ook al bedreven in het samenstellen en configureren van kleine netwerken, thuis of in het bedrijf van een familielid. Anderen hebben ervaring met repareren van computer-hardware, het overwinnen van de hardnekkige problemen die kunnen optreden bij het installeren van een nieuwe Windows-versie, of bij het installeren van Linux op een laptop, waarop oorspronkelijk Linux draaide. Bootstrap-loaders, sector-indelingen, file-systems, het zijn zaken waar velen al mee geconfronteerd zijn.

Naast het systematisch behandelen van de lesstof en het bevorderen van actieve beheersing door ze zelf dingen te laten maken is dit een derde “ingang”: nieuwsgierigheid geboren uit het zoeken naar oplossingen voor practische problemen met computersystemen.

Recept:

1. Inventariseer tegen welke problemen de studenten in hun persoonnlijke “computer-verleden” al aangelopen zijn, of ze deze konden oplossen en zo ja hoe.
2. Destilleer hieruit een klein aantal onderwerpen, bijvoorbeeld bootstrap-loaders of file-systems, en laat ze, geleid door de problemen die ze tegenkwamen, hierbij de diepte ingaan. Okee, aan de hand van die en die “recepten” op websites werd het probleem opgelost, maar wat zat er nu precies achter die reeksen ge-cut-and_paste commando’s?
3. Vraag de studenten het breder te maken. Je worstelde met GRUB of met EFI, maar hoe ziet het landschap van bootstrap-loaders er in bredere zin uit. Maak eens wat slides waarin je je medestudenten inwijdt in deze complexe wereld. Welke smaken zijn er? Hoe hangen die samen. Welke zijn oud, welke nieuw, welke goed, welke onhandig, waarom?

Op deze manier wordt een brug gebouwd van specifieke persoonlijke ervaringen naar een meer algemeen beeld van een bepaalde technologisch aspect van computersystemen. In de praktijk opgedane losse brokken kennis worden zo geïntegreerd tot een geheel.

3. Een contextrijke leeromgeving

De meest voor de hand liggende contex voor het leren kennen van computersystemen in het algemeen is natuurlijk de eigen computer van de student. Het temmen van dit weerbarstige apparaat is een klusje waarmee iedere laptopgebruiker die wel eens software intalleert wordt geconfronteert. Het zou allemaal évanzelf moeten gaan, maar voor je het weet moet je je verdiepen in de environment variabelen, zoekpaden, DLL versies en drivers.

Daarnaast zijn er de single-board systemen, zoals Arduino’s en Raspberries in hun diverse varianten waar de student mee te maken heeft. Context genoeg dus. Toch is het ook de moeite waard om de blik wat verder te richten. Super-computers, PLC’s, analoge computers, hele simpele processors zoals in

een koffieautomaat, speciale massive parallel hardware voor neurale netwerken, systemen met bijzonder eisen aan de betrouwbaarheid zoals in een auto, de wereld van “ubiquitous computing” is onbeperkt. Ook in je horloge zit een computer. Heeft die ook een besturingssysteem? En de 4-bit computer in de koffie-automaat? En je televisie? Je telefoon? Vaak zijn er meerdere chips, elk met hun eigen specifieke taak: audio, video, beeldherkenning, ga maar door.

Deze wereld van “ubiquitous computing” is minder toegankelijk dan die van PC, Arduino en Raspberry. Vaak betreft het proprietary hard- en software. Toch is dat de context waarin de ontwikkelingen plaatsvinden. De telefoon-als-computer is voor de meeste studenten belangrijker dan hun laptop. Deze wereld is zo divers dat het onbegonnen werk is deze af te dekken in het cursusmateriaal. Bovendien verandert hij voortdurend.

Betrekken van deze zeer relevante context in het vak kan het beste gebeuren door kleine “research opdrachten”, individueel of met een klein groepje. Welke soorten hardware tref je aan in een auto, het hoeft geen Tesla te zijn. Hoe zit een ABS systeem in elkaar? Of een MMU (Motor Management Unit)? Of een routeplanner? Zijn deze systemen verbonden, zo ja met wat voor soort netwerk? Specialistisch, zoals CAN-bus, of general purpose zoals Ethernet? Wat zijn de voor en nadelen van zulke keuzen? Veel industriële installaties en bijvoorbeeld schepen worden bestuurd door of via een PLC. Wat is dat precies? Waarom is daarvoor gekozen? Hoe gaan deze I/O-intensieve real time computers om met de specifieke eisen aan hun functioneren, bijvoorbeeld wat betreft betrouwbaarheid en voorspelbaarheid.

Met de opkomst van AI worden zelfs analoge computers weer relevant, omdat snelheid hier belangrijker is dan nauwkeurigheid. Onze hersenen werken niet met bits en bytes. Ze kunnen niet nauwkeurig rekenen met grote getallen, maar wel een slalom-skiëer op volle snelheid tussen de palen houden. Analog computing verschaft op zijn beurt weer een context voor analoge electronica in bredre zin, iets waar onze studenten zeer weining vanaf weten.

En wat te denken van Quantum Computing? Natuurlijk, het is op dit moment nog een esotherisch onderwerp. Maar ook wel een context die de fantasie van sommige studenten prikkelt. En, de toekomst is er voor je er erg in hebt!

4. Studeren door te leren van andere studenten en docenten

Zoals gezegd hebben de meeste studenten ervaring met het “knutselen” aan computers. En die ervaringen zijn divers. De één heeft wel eens een systeemkast gevuld met zelf uitgezochte boards, drives en powersupply. Een ander heeft juist weer ervaring met het installeren van een besturingssysteem. Weer een ander heeft ervaring met het configureren van een computer in een netwerk. Of met de aansturing van een geavanceerde videokaart. Of met het aan de praat krijgen van WiFi op een Raspberry. Deze diversiteit aan ervaringen zorgt onder de juiste omstandigheden vanzelf voor onderlinge kennisoverdracht.

Een voorbeeld van juiste omstandigheden is het met een groep vervullen van concrete gemeenschappelijke taak. Het genoemde geheel zelf bouwen van een computersysteem uit losse logische bouwstenen is een voorbeeld van zo’n gemeenschappelijke taak. Ook het samenstellen van een computersysteem uit losse boards of units valt eronder, waarna installatie van een besturingssysteem en configuratie binnen een netwerk volgt. Dit zou een hele leuke, haalbare praktische opdracht kunnen zijn. De losse bouwstenen kunnen meerdere jaren achter elkaar worden

gebruikt.

Weer een heel andere manier om studenten hiermee praktisch bezig te laten zijn, is om ze uit goedkope onderdelen een “Raspberry PC” te laten bouwen. Raspberry, schermpje, ge-3D-printe kast, touch keypad, evt. een muis en zie daar een super-low cost systeem. Hetzelfde, maar dan nog eenvoudiger en goedkoper kan met een Arduino. Dit is nog leerzamer, omdat in dit geval ook een rudimentair besturingssysteem moet worden geschreven, iets dat nu ook al in deelopdrachten voorhanden is. Zo’n eenvoudig computertje kan bijvoorbeeld worden gebruikt om opdrachtenreeksen naar een al even eenvoudig robotje te sturen. Indien de bediening simpel wordt gehouden is het resultaat een betaalbaar, inzichtgevend leermiddel voor kinderen. Wat ook wordt gemaakt, door samen aan iets concreets te werken leren de studenten van elkaar.

Leren van de docenten gebeurt natuurlijk bij dit vak ook, bijvoorbeeld wat betreft de theorie. Maar de docent kan ook zelf meebouwen. Of groepen assisteren die dat nodig hebben. Een docent die zelf iets vergelijkbaars bouwt is inpirerend. Zeker als de docenten, door zich opbouwende ervaring, iets maken dat net wat beter in elkaar zit dan waar studenten zelf op zouden zijn gekomen en hen daarmee op ideeën brengt. De docent als doorgeefluik van ervaring tussen studenten uit opeenvolgende studiejaren. Niets mis mee. Vrijwel alle goede docenten leren ook van hun studenten. Er is niets zo leerzaam als lesgeven.

5. Studeren met eigenaarschap en verantwoordelijkheid

Vrije keuze maakt eigenaarschap en verantwoordelijkheid makkelijker. Immers, wie wil er nu een verantwoordelijkheid opgedrongen krijgen, dat geeft verzet. Het vak Computersystemen en Logica omvat een breed scala aan onderwerpen. Alles wat komt kijken bij het functioneren van een practische computersystemen is al een heel divers pakket en logica komt daar nog eens bij als theoretische achtergrond. Het aldus bestreken terrein is breed genoeg om enige keuzevrijheid mogelijk te maken.

Een optie is een kernpakket aan te bieden met daarnaast verdieping van een aantal aspecten naar keuze. Die keuze kan bijvoorbeeld samenhangen met de taak in de groep die een student bij realisatie van bovenstaande mini-projecten op zich neemt.

Voorbeeld:
Iederen heeft kennis van:

1. Architectuur van een processor, ALU, ACCU, address bus, data bus
2. Principes van communicatie met randapparatuur, Memory Mapped I/O, Interupt, DMA
3. Virtual addressing en paging
4. Processes, priorities en scheduling

Daarnaast kiest elke student binnen één van deze vier terreinen expliceit een verdiepingsonderwerp, eventueel samenhangend met zijn taak binnen de boven beschreven mini-projecten, maar de student mag ook op andere wijze concrete “expert knowledge” aantonen, bijvoorbeeld:

1. Kennis van de architectuur en instructieset van een redelijk recente processor en schrijven van een stukje assembly in deze instructieset voor een functiecall en return

2. Gedetailleerde beschrijving van de communicatie tussen hoofdgeheugen en een video-kaart of harde schijf. Wat gebeurt er precies en in welke volgorde
3. Maak zelf een programma dat virtual addressing simuleert en op eenvoudige wijze visualiseert
4. Hoe werkt de scheduler van Linux of Windows precies, wat kun je eraan instellen en wat zijn de gevolgen, met zelfgemaakte demo-code

Vak: Technical Engineering Skills

Wat is het

Het vak Techical Engineering Skills is onstaan uit een aantal elementaire wiskunde-vakken voor eerstejaars. Het niveau waarop instromers op de middelbare school wiskunde hebben gehad, verschilt sterk. Instromers komen van MBO, HAVO of VWO, opleidingen die in deze volgorde een opklimmende hoeveelheid wiskunde-stof aanbieden. Echter blijft wat de studenten hier echt nog van weten ver achter bij wat ze volgens de lesprogramma's geleerd zouden moeten hebben. Vaak is de wiskunde volledig "verdwenen".

Eenvoudige algebraïsche manipulaties worden veelal niet beheerst en allerlei onbegrepen truukjes worden op het verkeerde moment toegepast. Er is nog een ander probleem: Een aanzienlijk deel van de studenten blijkt niet te kunnen rekenen, ook niet als het hele simpele basisschool-sommen betreft, zoals 40 : 8 of 60 – 15. Sommige studenten weten niet wat negatieve getallen zijn. Breuken worden vaak totaal niet begrepen, en dat iets hetzelfde blijft als je eerst met 15 vermenigvuldigt en er daarna weer door deelt is voor sommigen een raadsel. Ook bij VWO'ers blijkt een dergelijk intuïtief begrip van bewerkingen en hoeveelheden vaak afwezig.

Al met al is het op niveau krijgen van de wiskunde van instromers een moeilijke taak. Hoe repareer je in een beperkte tijd het ontbreken van een stuk basisschool-kennis? Binnen het reguliere programma is dit een groot probleem, ook indien reparatievakken worden aangeboden. Een mogelijke oplossing is een tussenjaar waarin dergelijke zaken worden aangeleerd.

In het vak Technical Engineering Skills draait het niet primair om het wegnemen van het "reken-deficiet". Toch speelt het gebrek aan inzicht in getallen en bewerkingen voortdurend een rol. Als je dat inzicht niet hebt, wordt algebra nooit meer dan een betekenisloze trukendoos. Twee prakijkvoorbeelden van misbruik van truuks:

1. Pelikaan: $2 \times (3 + 4) = 2 \times 3 + 2 \times 4$ dus ook $2 + (3 \times 4) = 2 + 3 \times 2 + 4$

Nodig: Uitleg distributieve eigenschap

2. Naar de andere kant brengen: $2 + 3 = 4$, dus $2 = 4 + 3$

Nodig: Uitleg over vergelijking als weegschaal in balans, waarbij je aan beide kanten zelfde gewichten mag bijplaatsen of wegnemen. Een gewicht naar de andere kant brengen verstoort juist de balans.

Veel studenten zijn dergelijke basale algebraïsche manipulaties niet machtig. Dit speelt ze parten in alles wat ze met formules doen en maakt dat ook nieuwe stof die nieuw voor hen is, zoals functie-onderzoek of differentiëren, niet goed landt.

Uitgangspunt bij het vak Technical Engineering Skills is dat precies die wiskunde wordt aangeboden, die nodig is binnen de Technische Informatica. Daarmee valt een deel van de middelbare school stof af, en komen een aantal zaken in beeld die op de middelbare school misschien niet aan de orde komen. Bij een opleiding waarbij computers en bijvoorbeeld simulatie een rol spelen, ligt numeriek integreren en en differentiëren meer voor de hand dan een analytische aanpak. En voor elementaire ontwerp-vaardigheden op het gebied van de analoge computers, opnieuw relevant vanwege neurale netwerken, zijn complexe getallen ontzettend handig, een simpele uitbreiding op 2D vectoren. (N.B. “Complex” betekent hier niet “ingewikkeld” maar “samengesteld”).

Naast wiskunde komt in Technical Engineering Skills ook natuurkunde aan de orde. Want ook op dit terrein zijn veel studenten zo groen als gras. Concepten als kracht, afgelegde weg, energie, vermogen, stroom, spanning en weerstand zijn bij de meeste studenten onvoldoende bekend om de T in Technische Informatica inhoud te geven.

Tegelijkertijd biedt die natuurkunde een mooie gelegenheid de wiskunde in de praktijk te brengen. Wie analytisch kan integreren, hoeft de formule voor de plaats als functie van de tijd s(t) van een voorwerp waarop een constante kracht werkt, en dat dus een constante versnelling a ondergaat, niet te onthouden. Immers deze expressie wordt simpelweg verkregen door constante versnelling a twee maal naar de tijd te integreren.

$$a(t) = a$$

$$v(t) = v(0) + a\,t$$

$$s(t) = s(0) + v(0)\,t + \tfrac{1}{2}\,a\,t$$

Deze supersimpele, bij elk voer- vaar- of vliegtuig relevante toepassing kan wordt aangegrepen om het directe nut een theoretisch onderwerp als integreren duidelijk te maken. Ook op de middelbare school ligt deze bal al voor het doel. Het is goed hem er dan ook in te trappen en de studenten te laten zien hoe praktisch zo’n moeilijk onderwerp kan zijn.

Het vak Technical Engineering Skills is nog in beweging. Met de voortdurende verandering in het vakgebied TI hoort dat ook zo te blijven. De wiskunde, zoals die in het middelbaar onderwijs gegeven wordt, lijkt vaak nog een doel op zich. Iemand die op universitair niveau een studie regeltechniek gaat volgen, krijgt te maken met asymptoten van een functie zoals $(x + a) / (x + b)$, maar in het algemeen gesproken is het zo dat deze kennis zelfs op het VWO voor slechts een zeer klein deel van de leerlingen ooit relevant zal worden.

De eindeloze moeite die wordt gestoken in bijvoorbeeld meetkundige inzichten en bewijzen met behulp van driehoeken is een ander voorbeeld. Wil degene die dat *ooit* prakisch gebruikt heeft z’n vinger opsteken! Natuurlijk, het is de basis voor veel bewijzen. En wiskunde *moet* rigoreus bewezen worden. Maar dat is de taak van wiskundigen, niet van technici. Goniometrische functies daarentegen zijn cruciaal in de natuurkunde en hebben dan ook heel veel practische toepassingen.

Het is onjuist de verschuiving van analytische wiskunde en bewijzen naar numerieke wiskunde en intuïtie als acheruitgang aan te merken. De hoeveelheid kennis waarover de mensheid beschikt wordt

steeds groter. Tora-studenten kenden enorme lappen tekst uit hun hoofd. Veel meer was er in die tijd ook niet. In de renaisance was er nog de “homo universalis”, de intellectueel die alles wist wat er maar te weten was. Dat is voor niemand meer haalbaar, de hoeveelheid kennis waarover de mensheid beschikt groeit sneller dan wie dan ook kan bijhouden. Er moeten dus keuzen worden gemaakt. Met dit simpele feit in het achterhoofd is een vak als Technical Engineering Skills een schot in de roos. Het vak is voor een groot deel al ingericht volgens de vijf uitgangspunten.

Het vak door de bril van de vijf uitgangspunten

1. Studeren met plezier

Veel studenten hebben na de middelbare school een wiskunde-trauma. Als je daar in een willekeurige groep studenten naar vraagt, gaan er na wat aanvankelijk gegniffel heel wat vingers omhoog. In groep bedrijfsinformatici was de score meer dan 80%. Bij TI’ers ligt het hopelijk iets gunstiger. Plezier en wiskunde lijken op de middelbare school wel diametraal tegenover elkaar te staan. In hoog tempo volgen de onderwerpen elkaar op. Waar het allemaal voor nodig is, is voor de meesten niet duidelijk.

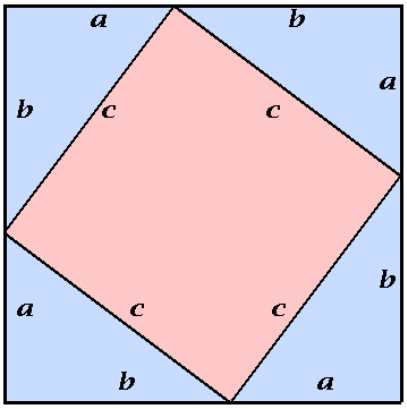
Met het maken van een groot aantal opgaven worden handelingen die weinigen echt begrijpen geautomatiseerd. De engelse kreet “mathemagic” is hier van toepassing. Natuurlijk staan in de huidige wiskunde-boeken voor de middelbare school voorbeelden van praktische toepassing. Maar deze zijn vaak heel talig. Wat enorm helpt is een bevlogen leraar met enthousiasme voor z’n vak, gevoel voor z’n leerlingen en een enorme zak vol anekdotes die telkens maar weer het praktisch nut van wat wordt geleerd laten zien. Dat gebeurt nog steeds te weinig, helaas. Veel leerlingen leren op de middelbare school van wiskunde vooral één ding: dat ze er niet goed in zijn.

Het is niet eenvoudig de vicieuze cirkel van gebrek aan zelfvertrouwen, angst in plaats van plezier, gevolgd door vermijdingsgedrag te doorbreken. Toch liggen de kaarten bij HR-TI gunstig. Zowel de bits en bytes van software en data, als de stromen en spanningen in de hardware zijn onzichtbaar. Het enige wat we hebben is een gedachten-model ervan. En dat gedachten-model *is* wis- en natuurkunde. Geen TI student hoeft in het duister te tasten over waarom hij iets leert. Het leidend maken van de toepassing in het vak Technical Engineering Skills is een goede keuze. Nee, het is geen opstapje naar het beroep van wis- of natuurkundige. Maar wel naar het beroep van TI’er, een technical engineer die werkt op de grens van informatica met deze twee vakgebieden. Je mannetje (M/V/X) staan op deze gebieden draagt bij aan het vertrouwen in eigen competentie. En toenemend zelfvertrouwen geeft toenemend plezier. Je steeds zekerder gaan voelen. Ontdekken dat je het *wel* kunt. En dat het eigenlijk niet gaat om magische formules maar gewoon om gezond verstand.

2. Studeren met nieuwsgierigheid

Voor wiskunde geldt bij uitstek: onbekend maakt onbemind. Om de interesse te wekken is het allereerst nodig dat studenten geconfronteerd worden met het “low hanging fruit” in de wiskunde. Een goed voorbeeld hiervan is de stelling van Pythagoras.

Voorbeeld W1: Als je 3 meter naar het noorden loopt en daarna 4 meter naar het westen, dan heb je in totaal $\sqrt{(3^2+4^2)} = 5$ meter afgelegd. De meeste studenten kennen deze stelling wel, maar hebben geen idee waarom dit altijd zo is. Het bewijs is zeer eenvoudig:



De totale oppervlakte van bovenstaande figuur is $(a + b)^2$, maar ook: $c^2 + 4ab$

Dus $(a + b)^2 = c^2 + 2ab \Leftrightarrow a^2 + 2ab + b^2 = c^2 + 2ab \Leftrightarrow a^2 + b^2 = c^2 \Leftrightarrow \sqrt{(a^2 + b^2)} = c$

Studenten blijken het enorm leuk te vinden dat zo'n bekende stelling, die de meesten wel eens hebben gebruikt, in één regel te bewijzen is. Ze willen het bewijs snappen en onthouden, zodat ze er ergens anders de blits mee kunnen maken. Kortom: Hier is sprake van plezier.

Voorbeeld W2: Een rijke man heeft 17 kamelen. Als hij overlijdt erven zijn drie kinderen die. Het oudste kind krijgt 1/2 van de erfenis, het tweede kind 1/3 en het laatste kind 1/9. Echter willen ze geen kamelen in stukken snijden. Verderop woont een arme, wijze man die 1 kameel heeft. Hoe lossen ze dit probleem op?

Antwoord: de arme man leent zijn kameel aan de drie kinderen, zodat er nu 18 kamelen zijn. Het eerste kind krijgt 1/2 x 18 = 9 kamelen, het tweede kind 1/3 x 18 = 6 kamelen, het derde kind 1/9 x 18 = 2 kamelen. Samen zijn dat 17 kamelen, dus na het verdelen van de erfenis geven ze de wijze man zijn kameel terug.

Dit voorbeeld tovert meestal een verbaasde grijns op de gezichten, waarna je de denkrimpels ziet verschijnen. Hoe kan dat? Dat kan toch niet? Een mooi opstapje naar de uitleg van breuken. Immers:

$$1/2 + 1/3 + 1/9 = 9/18 + 6/18 + 2/18 = 17/18$$

Enerzijds eenvoudig, anderzijds laat het zien hoe zeer intuïtie je op het verkeerde been kan zetten. En hoe noodzakelijk het daarom af en toe is om exact te redeneren. En exact redeneren, daar draait het in de wiskunde om.

Ook natuurkunde en plezier gaan gemakkelijk samen. Samenstellen van krachten kan worden gedaan met touwtjes die in verschillende richtingen trekken. Hefbomen, katrollen, slingers, het is allemaal eenvoudig te demonstreren.

Nog veel nuttiger voor TI is het demonstreren van eenvoudige electrische circuits en hoe je daaraan

kunt meten. Vertrouwdheid met een regelbare voeding, signaalgenerator, universeelmeter en oscilloscoop is daarbij het absolute minimum. Werken aan electronica zonder kennis en gebruik van deze instrumenten is als koken zonder je zintuigen te gebruiken.

Het werkelijk gebruiken van deze apparatuur geeft meer plezier dan een theoretisch verhaal of een plaatje in een boek. TI'ers zijn doeners. Daar kunnen we gebruik van maken, ook bij een theoretisch vak als Technical Engineering Systems.

Voorbeeld N1: Het meten van stroom, spanning en weerstand met een universeelmeter. De meter moet op de juiste wijze in het circuit worden geplaatst. Vervolgens doorrekenen van spanningsdeler en kijken of de uitkomsten kloppen met de metingen.

Voorbeeld N2: Theorieles over complexe getallen, gevolgd door het schrijven van een programma'tje dat hiermee rekent aan een laagdoorlaat-filter, gevolgd door het testen hiervan met behulp van een signaalgenerator en een oscilloscoop.

Samenvattend: Veel studenten hebben geen, of een negatief beeld van wiskunde. Ook zijn er velen die nooit natuurkunde hebben gehad. Toespitsen van de stof op wat nodig is voor TI en het voortdurend zichtbaar maken van de praktische relevantie zijn goede middelen om ook wis- en natuurkunde spannend en interessant te maken. Ook bij dit ogenschijnlijk voornamelijk theoretische vak is de praktijk niet ver weg. Bij de wiskunde is er "low hanging fruit" dat intrigeert en nieuwsgierigheid wekt. Bij de natuurkunde geldt het principe "don't tell, show". Het beste werkt het als de student zelf op een eenvoudige, voor TI relevante manier actief met de stof bezig is, bijvoorbeeld zoals in voorbeeld N1 of N2, of met hele andere practische activiteiten.

Het is een misverstand dat zulke practische activiteiten een solide presentatie van de theorie in de weg staan. Ze versterken deze juist. Het verschil met een project is dat hier de systematische opbouw van de theorie leidend is, niet de systematische weg naar een practisch product.

3. Een contextrijke leeromgeving

De switch die is gemaakt van losse wiskundevakken naar Technical Engineering Systems, is een duidelijke keuze voor een bepaalde context. Het gaat blijkbaar niet om wiskundige kennis die je toepast in de econometrie, bij weersvoorspellingen of bij de analyse van de verspreiding van een virus, maar in de eerste plaats om kennis die nodig is bij het ontwerpen en bouwen van technische systemen.

Een TI'er is een Technical Engineer, dus het bouwen van zulke systemen is z'n natuurlijke habitat. Voor context hoeft dan ook niet ver te worden gezocht. Soms is al een verbinding met de context gemaakt. Zo is het duidelijk dat voor beeldverwerking lineaire algebra nodig is. Echter zijn er ook nog veel links die *niet* worden gelegd.

Een voorbeeld is de link tussen statistiek en statistische informatietheorie. Practische zaken als Shannon informatie, bandbreedte en lossless of lossy codering zijn een prima illustratie van de begrippen zoals kans, kansdichtheid en (kruis)correlatie. Een bit is voor een TI student een 1 of 0. Oprekken van dit begrip richting $-^{2}\log(p)$ maakt in één klap zowel logaritmen als kansen tot relevante

onderwerpen voor informatici.

Vrijwel geen enkele student weet dat er een theoretische limiet is aan hoeveel informatie je over een bepaald onvolmaakt kanaal, wired of wireless, kunt versturen. Met een paar eenvoudige voorbeelden van statistische codering kan dat duidelijk worden gemaakt. Van “moeilijke theorie” is daarbij geen sprake, wel van het toepassen van iets dat zowiezo wordt geleerd: statistiek en kansrekening.

Ook de praktische relevantie van de regel van Bayes kan met eenvoudige voorbeelden worden duidelijk gemaakt. De afgelopen jaren was er een gastles waarin wordt behandeld hoe je delfstoffen op kunt sporen met behulp van geologische kennis en Bayes.

Nog een voorbeeld van het praktische gebruik van statistiek is statistische kwaliteitscontrole bij een productielijn, met behulp van de zogenaamde AQL, Acceptance Quality Limit. Voor elke TI’er die ooit aan een productielijn zal werken is dit interessant. En er is geen nieuwe theoretische kennis voor nodig, alleen maar toepassing van het vak Statistiek.

Wat betreft context voor natuurkunde is de zaak nog eenvoudiger. Hoeveel studenten hebben bij het liftproject door dat de kracht die de liftmotor moet leveren, omgekeerd evenredig is met de doorsnede van de kabeltrommel? Is je motor niet sterk genoeg? Dan maak je het opwind-klosje dunner.

Het onderwerp electriciteitsleer is wat dat betreft het meest contextrijk. Vaak gehoord van studenten: de motor doet het niet, terwijl ik toch de goede voedingsspanning gebruik. Het advies om eens te kijken of de voeding genoeg stroom kan leveren leidt onveranderlijk tot niet-begrijpende blikken. Het gaat toch om de spanning? Dat is toch wat op de motor staat? Dat is toch wat de knop of wijzer op de voeding aangeeft. Dat er door de inwendige weerstand sprake is van een spanningsdaling en een maximaal af te geven vermogen is volledig onbekend. Ook hier is er weer een goede mogelijkheid de theorie gelijk toe te passen op een hele praktische manier. Ook dit kan het beste weer geen papieren oefening zijn of een YouTube video, maar een practicumopgave die de studenten in kleine groepjes uitvoeren en waar individueel verslag van wordt gedaan.

Zelfs indien deze en soortgelijke opdrachten in groepjes worden gedaan is daar enige apparatuur voor nodig. Deze is echter tegenwoordig zeer betaalbaar. Het voordeel als studenten inderdaad hands-on dit soort dingen doen is enorm. Gezien de profilering van TI als opererend op de grens van hard- en software is de investering hierin volstrekt gerechtvaardigd.

4. Studeren door te leren van andere studenten en docenten

Leren van de docent is bij wis- en natuurkunde een voor de hand liggende vorm van onderwijs. Immers bij deze vakken ligt omvang en inhoud van de kennis immers exact vast. Omdat er door de jaren heen weinig aan de essentie van deze vakken verandert, weet een ervaren docent over het algemeen precies waar de problemen zitten en hoe hij die kan verhelderen. Frontaal lesgeven is hier dan ook een efficiënte vorm van kennisoverdracht.

Omdat het ingangsniveau Wis- en Natuurkunde van de studenten sterk verschilt, is leren van elkaar ook een reële optie. Bijvoorbeeld: Na uitleg door de docent maken alle studenten een bepaalde set opgaven zo goed mogelijk. Daarna gaan de studenten die niet uit een bepaalde opgave gekomen zijn op zoek naar een medestudent die deze opgave wel tot een goed einde kon brengen voor hulp. Daarbij zullen

zich over het algemeen groepjes “discipelen” rond een aantal “guru’s” vormen. De docent en eventuele peer-coaches spelen ook gewoon de rol van guru. De aanwezigheid van viltstiften, flipovers en whiteboards vergemakkelijkt deze vorm van kennisoverdracht.

Indien bij de practicumopgaven groepsgewijs wordt gewerkt, vindt ook daar weer kennisoverdracht tussen de studenten onderling plaats. Door steekproefsgewijs om uitleg te vragen, bewaakt de docent dat alle studenten uiteindelijk snappen hoe een bepaald resultaat tot stand is gekomen.

5. Studeren met eigenaarschap en verantwoordelijkheid

Het is lastig, aan studenten de werkelijke relevantie van een vak als wiskunde of natuurkunde duidelijk te maken. Het is typisch een vak dat van pas komt in een veelheid van situaties en hoe nuttig deze kennis is blijkt eigenlijk pas gaandeweg de beroepspraktijk. Toch is het de taak van de docent die relevantie zo goed mogelijk duidelijk te maken aan de hand van z’n eigen kennis van, of ervaring met de problemen die een TI’er op z’n bord krijgt.

Verantwoordelijkheid nemen voor een vak dat je niet onmiddelijk aanspreekt vergt een volwassen houding en ook vertrouwen dat het niet voor niets in het curriculum zit. Het weglaten van franje kan dit vertrouwen bevorderen. De huidige generatie studenten is er aan gewend, overladen te worden met informatie en daaruit een selectie te moeten maken. De vraag “hoe nuttig (of interessant) is dit voor mij” is dan ook gerechtvaardigd en verdient steeds opnieuw een zo helder, oprecht en praktisch mogelijk antwoord. Indien relevantie voor de beroepspraktijk steeds leidraad is bij het verder ontwikkelen van dit vak, is die vraag zondermeer op een goede manier te beantwoorden.

===é

Vak: Databases

Wat is het

Het vak Databases gaat over het schrijven van SQL queries op een relationele database (RDB). Daarbij komen ook zaken zoals foreign keys en joins aan de orde. Tentaminering vindt plaats aan de hand van een aantal korte programmeeropgaven in SQL.

Relationele databases komen in de praktijk in alle soorten en maten voor, van grote losstaande database-servers met gebruik van bijvoorbeeld SQLServer of MySQL, die door meerdere applicaties via een netwerk-verbinding kunnen worden gebruikt, tot kleine databases die slechts lokaal binnen één applicatie worden gebruikt zoals het SQLite package, dat eenvoudig binnen Python programma's te gebruiken is.

De querytaal SQL is gestandaardiseerd en ziet er daarom in grote lijnen op elke RDB hetzelfde uit. De complexiteit van het opzetten van een database, connections en zaken zoals logging en security verschillen per platform sterk van elkaar. Omdat het vak Databases vooral draait om SQL, is die diversiteit in setup, connections en security geen probleem. Bij practisch gebruik zijn het echter helaas juist deze "bijzaken" die weerbarstig kunnen zijn, omdat een echte standaard hiervoor ontbreekt. Het aan de praat krijgen van een RDB is daarom vaak een kwestie van heel veel lezen op Internet en vervolgens net zo lang proberen "tot ie het doet". De echte experts op een bepaalde RDB hebben daar natuurlijk geen last van, maar incidentele SQL-programmeurs wel. Gelukkig is zijn die "bijzaken" bij de meeste bedrijven niet ondergebracht bij softwareontwikkeling maar bij systeembeheer of een speciale database-administration.

SQL kan op allerlei complexiteitsniveaus worden gebruikt. De meest voorkomende toepassing voor TI'ers is simpel: Een paar losse tabellen, vaak zelfs zonder gebruik van foreign keys en joins. Het vak Databases gaat echter aanzienlijk verder. Een eenvoudige rondvraag leert dat zelfs onder programmeerdocenten zaken zoals joins, foreign keys, many-to-many relations en nested queries in de praktijk nauwelijks worden gebruikt.

Het lastigste is de zogenaamde "referential integrity": Blijft alles kloppen als je instances van bepaalde entiteiten weggooid, of ontstaan er dan ongeldige verwijzingen. De situatie is te vergelijken met die van "dead links" op Internet. Handhaven van referential integrity bij het wissen van gegevens is ook voor de professionele database programmeur een lastig probleem. De enige strategie die echt werkt is om de zaken zo simpel mogelijk te houden.

De vraag hoe ver dit vak zou moeten gaan voor de doelgroep TI'ers is een fundamentele discussie waard. Echter wordt er ook hier weer voor gekozen het vak te accepteren zoals het nu is. Doel van dit document is immers, gegeven de vakinhoud, te kijken hoe de vijf uitgangspunten toegepast kunnen worden.

Het vak door de bril van de vijf uitgangspunten

1. Studeren met plezier

Queries schrijven is een (beperkte) vorm van programmeren. Vaak zijn SQL-queries ingebed in een programma dat geschreven is in een andere programmeertaal zoals C++, Java of Python. Het kunnen opslaan van data in een RDB is bij zulke talen zeker een aanvulling, die flexibeler is dan rechtstreeks gebruik van datafiles in de betreffende taal. Een uitzondering is misschien Python, dat met z'n Pickle en JSON modules ook andere laagdrempelige methoden van opslag biedt. Voor grote hoeveelheden numerieke data is een formaat als HDF5 geschikter. Maar voor opslag van de typische "platte" administratieve gegevens is een RDB handig. Vanuit object georiënteerde talen wordt vaak nog een tussenlaag gebruikt tussen programma en database, de object-relational mapping (ORM). Deze maakt gebruik van SQL veelal overbodig. Echter wordt hieraan in dit vak niet of nauwelijks aandacht besteed.

Het schrijven van uitgebreide losse SQL queries is niet een typische TI activiteit. Het schrijven van een klein stukje SQL als onderdeel van een programma in een imperatieve programeertaal is dat wel. Er vanuit gaand dat plezier en practische toepasbaarheid hand in hand gaan is dit geen ideale situatie.

Naast het naïeve gebruik ervan via SQL is nog een andere link tussen RDB's en TI: security. Waneer wordt gewerkt met embedded SQL, d.w.z. letterlijkge SQL "strings" ingebed in een andere taal, opent zich de mogelijkheid van SQL injection. Met name in een taal zoals C++ kan bij onoordeelkundig gebruik van strings een veiligheidslek ontstaan, waarbij de gebruiker via een acherdeurtje willekeurige queries op de database kan uitvoeren door in onbeschermd geheugen strings te overschrijven.

Security is een onderwerp dat tot de verbeelding spreekt bij veel studenten. Een mogelijkheid tot vergroten van het plezier is dan ook om dit aspect in het vak te betrekken, ook hier weer via een practicum-opgave. Groepjes studenten maken (of krijgen) een eenvoudige webserver die op onveilige wijze gebruik maakt van embedded SQL. De ene helft van een groepje krijgt de opdracht deze zo veilig mogelijk te maken, de andere helft moet proberen via SQL injection zoveel mogelijk schade aan te richten (uiteraard binnen een veilige "zandbak").

2. Studeren met nieuwsgierigheid

Gebruik van SQL vergroot de mogelijkheiden om met handig en met weinig code met opgeslagen data en de relaties ertussen om te gaan. Dat op zich is al intrigerend genoeg om de nieuwsgierigheid naar dit tool te wekken. Beheersing ervan zorgt dat de student opeens veel meer kan, er gaat zogezegd een wereld van data open.

Een andere "inroad" naar nieuwsgierigheid is het relationele model zelf in samenhang met het wiskundige begrip "relatie". Een tabel is een "relatieverzameling" en een relationele database is een database van buiten af gezien uitsluitend is opgebouwd met zulke "relaties", tabellen dus. Aan deze solide wiskundige achtergrond dankt de RDB z'n flexibiliteit. Het is namelijk aan te tonen dat alle mogelijke verbanden tussen data ermee weergegeven kunnen worden. Hoe zit dat precies? Opdracht: Schrijf een kort artikel dat de brug slaat tussen wiskundige relaties en het gebruik van het woord relatie als synoniem voor tabel. Deze begrippen komen één op één overeen. Laat studenten dit in eigen woorden zo helder mogelijk uitleggen. Nodig is wel dat de docent dit allereerst helder uitlegt,

bijvoorbeeld aan de hand van “An Introduction to Relational Databases” van

Meestal wordt een RDB als gegeven beschouwd, maar hoe zit hij eigenlijk van binnen in elkaar? De vraag hoe iets “in elkaar zit” is een typische TI vraag. Een manier om interesse te wekken is de studenten zelf een eenvoudig RDB te laten maken. In plaats van “echte” SQL kunnen de studenten een Table class met een aantal methods schrijven die de mogelijke statements in SQL zo goed mogelijk emuleren. Zo worden ze aangemoedigd, te kijken wat bepaalde statements in SQL *precies*doen. Ze moeten immers die functionaliteit zo goed mogelijk nabootsen.

Een voorbeeld van hoe gebruik van zo’n zelfbouw SQL er uit kan zien:

```
primateCursor = (
        FROM (menu) .
        WHERE (lambda r: (r.plantSpecies, 'cultivated') in (
                FROM (primateFood) .
                SELECT (lambda r: (r.species, r.kind))
        )) .
        ORDER_BY (lambda r: r.name)primateCursor = (
        FROM (menu) .
        WHERE (lambda r: (r.plantSpecies, 'cultivated') in (
                FROM (primateFood) .
                SELECT (lambda r: (r.species, r.kind))
        )) .
        ORDER_BY (lambda r: r.name)
)
)
```

Iets dergelijks is met heel weinig code te maken. Niet iedere student hoeft hier even ver in te komen. Het gaat er alleen maar om dat studenten zich realiseren wat een statement zoals

FROM … SELECT …

nu eigenlijk precies doet. Zelf proberen iets soortgelijks te maken levert een leerervaring op die veel dieper gaat dan oppervlakkig begrip.

3. Een contextrijke leeromgeving

Leerboeken en cursussen over RDB’s en SQL plaatsen voorbeelden onveranderlijk in een administratieve context. De tabellen heten *werknemer*, *afdeling* en *salarisschaal* of bijvoorbeeeld *hotelkamer*, *reservering* en *gast*. Voor studenten met een technische interesse zijn dergelijke voorbeelden al gauw saai. “Is dit iets waar ik in mijn werk mee te maken wil hebben?”

Voorbeelden uit de TI wereld liggen meer voor de hand: *servo, leverancier* en *fabrikant*, of *systeemkast*, *motherboard* en *power supply*.”Hé, iets dergelijks zou handig kunnen zijn, misschien leuk

om eens te maken.”

Het vervangen van begrippen uit de adminstratieve wereld door begrippen uit de TI wereld is een eenvoudige ingreep die direct al effect sorteert. Veel leuker zijn natuurlijk projecten of practicum-opgaven waarin de RDB en de TI wereld echt vervlochten zijn. Het bank-project is daarvan enigszins een voorbeeld. Toegangscontrole en het opslaan van een serie handelingen voor een robot of van weersgegevens zijn andere voorbeelden.

In het algemeen gesproken is het belangrijk dat de onderwerpen die deel uitmaken van het vak Databases ook werkelijk op een praktische manier terugkeren in de rest van de TI opleiding. Dat is nu slechts gedeeltelijk het geval. Indien Databases op het huidige niveau gegeven blijft worden, is het nodig dat ook de wat lastigere queries ergens binnen een ander vak of een project worden toegepast. Op dit moment liggen de diepgang waarmee SQL wordt onderwezen en de diepgang waarmee SQL in een context wordt toegepast ver uitelkaar.

4. Studeren door te leren van andere studenten en docenten

SQL is ook weer een welomschreven onderwerp dat op eenduidige wijze kan worden behandeld. Wat dat betreft lijkt het een beetje op wiskunde. Dat is logisch, immers een tabel is niets anders dan een wiskundige relatieverzameling. Daarmee leent ook dit onderwerp zich prima voor frontale presentatie door een docent, het bekijken van een YouTube video of het zelfstandig bestuderen van een boek. Dit laaste ligt lastig, lezen van iets dat verder gaat dan een klein tekstblokje blijkt voor veel TI studenten een barriere.

Er zijn veel goede boeken over SQL/RDB en behalve volledig zelfstandig bestuderen hiervan door individuele studenten is het ook mogelijk met een groepje studenten een korte uitleg op slides te maken van bijvoorbeeld één hoofdstuk. Hierbij hebben de minder goede lezers steun aan studenten in hun groepje die daar geen moeite mee hebben en onduidelijkheden kunnen toelichten. Deze slides plus bijbehorende uitleg kunnen vervolgens aan de andere groepen worden gepresenteerd. Studenten uit die andere groepen mogen vragen stellen over het onderwerp. De docent wijst steeds een student uit de presenterende groep aan die de betreffende vraag beantwoord. Dit maakt het voor de presenterende groep nodig elkaar goed “bij te spijkeren” over het betreffende onderwerp alvorens er een presentatie over te geven. Actief op verschillende manieren met de lesstof bezig zijn is het resultaat.

5. Studeren met eigenaarschap en verantwoordelijkheid