Stichting NIOC en de NIOC kennisbank

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

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

Op dit moment bevat de NIOC kennisbank alle bijdragen, incl. die van het laatste congres (NIOC2025, gehouden op donderdag 27 maart 2025 jl. en georganiseerd door Hogeschool Windesheim). Bij elkaar zo'n 1500 bijdragen!

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

Het eerstvolgende NIOC vindt plaats op 18 maart 2027 in Arnhem en wordt georganiseerd door HAN University of Applied Sciences.

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

R. Smedinga kennisbank@nioc.nl.

Vermeld bij reacties jouw naam en telefoonnummer voor nader contact.

Software Engineering: leren om beter te falen

We investeren in grote software projecten om een nieuwe wereld te scheppen. We maken dromen waar met inzet van de nieuwste technologische mogelijkheden. De complexiteit van deze projecten is enorm. We nemen tal van beslissingen die we nog niet eerder hebben genomen, en we kunnen niet direct zien of deze beslissingen goed zijn. Falen is hierbij onvermijdelijk. Onze taak als opleiding is om studenten te doordringen dat er grote kans is op fouten in hun visie en ideeën, om hun gedachten zo te presenteren dat deze fouten vroegtijdig (door anderen) gevonden kunnen worden; zodat deze fouten niet leiden tot het falen van het project.

De illusie van Rational Decision Making hebben we al lang overboord gezet. Parnas schreef het al, hoogstens kunnen we onze beslissingen gestructureerd opschrijven, maar ons creatief proces verloopt alles behalve rationeel. Het werk van Klein laat zien dat de meeste beslissingen automatisch gebeuren. In de theorie van Naturalistic Decision Making wordt dit gepresenteerd als een proces waarin we patronen herkennen en op basis daarvan eerder geleerde oplossingen kiezen. Een proces waarbij grote versimpeling plaats vindt, die ons in staat stelt snel te handelen. In zijn briljante boek "Thinking, fast and slow" laat Kahneman zien hoe ons besluitvormingsproces hierbij verstoord wordt. Waar Klein nog tal van voorbeelden laat zien van professionals die in staat zijn goede besluiten te leren nemen, stelt Kahneman dat dit alleen geldt als er een goede en directe feedback loop is. Die ontbreekt helaas bij de meeste IT beslissingen. Het verontrustende hierbij is dat wij van fouten die op een later moment blijken niet of nauwelijks leren. Schon en Argyris, auteurs van de reflective practioner, laten in hun onderzoek zien dat wij de oorzaak van deze fouten meestal neerleggen buiten ons zelf. In hun woorden: advocacy verstoort het proces van inquiry. En zelfs als we de fout neerleggen bij onszelf is het nog maar zeer de vraag of we hiervan kunnen leren wat we nu precies in de volgende situatie anders moeten doen. Het leren is te indirect geworden.

Als opleiding is het zaak om onze studenten te leren om complexe beslissingen beter te nemen. Maar ook om ze het bewustzijn te geven dat ze een aantal van deze beslissingen verkeerd zullen nemen. Hoe zorgen ze dat hun fouten door anderen gevonden kunnen worden. Hoe zorgen ze dat hun fouten vroegtijdig ontdekt kunnen worden. Hoe zorgen ze dat deze fouten niet leiden tot het falen van het hele project. Hoe kunnen ze falen zodat ze ervan leren. Hoe kunnen ze falen en achteraf toch staan voor hun beslissing.

Je kan stellen dat de taak van de opleiding uiteen valt in het volgende:
- Zorgen voor vakkennis
- Zorgen voor de juiste attitude
- Ontwikkelen van leercompetenties

Ad) Zorgen voor vakkennis

Problemen herkennen en oplossingen aanleren. Dit is de kern van vele engineering opleidingen. We beginnen hier vaak met goed gedefinieerde problemen en leren daar standaard oplossingen aan. In basis programmeertalen zitten tal van leerpunten. En op een groter niveau hebben we veel kennis vastgelegd in design patterns. Wat hierbij lastig is, is dat problemen in tal van gedaanten voorkomen, dat er een grote variatie is in oplossingen en dat het verschil tussen een goede en slechte oplossing vaak in details zit. Ook is een compositie van verschillende deeloplossingen complex, en leidt vaak tot over design. De basis is weten waar je mee bezig bent.

Software engineering kenmerkt zich door slecht gedefinieerde problemen. Het is onduidelijk hoe je het probleem beschrijft, en of je oplossing klopt. Een wezenlijk onderdeel van elke opleiding is om studenten te laten wennen aan deze grotere slecht gedefinieerde problemen, waar ook creativiteit een belangrijke rol speelt. Waar er een iteratieve analyse - ontwerp - proof of concept - test fase moet worden doorlopen.

Ten slotte is het van belang om studenten te leren abstraheren. Dit kan door ze kennis te laten maken met verschillende programmeerparadigma's en ze modellen en logica aan te leren.

Ad) Zorgen voor de juiste attitude
Een van de grote uitdagingen in de master is om studenten te doordringen dat falen erbij hoort. Dat ze durven experimenteren, niet bang zijn om het verkeerd te hebben. Blijven leren, ook door frequent te reflecteren.

Niets menselijks is onze engineers vreemd. Ze willen het graag goed doen en zoeken naar bevestiging. In ons vakgebied wordt dit versterkt doordat je pas het werk krijgt als je de opdrachtgever het vertrouwen geeft dat je het kan.

We werken in de master aan deze attitude. Vooraleerst door studenten opdrachten te geven waarin ze falen. Door ze te confronteren met genadeloze inhoudelijke kritiek op hun eerste ontwerpen, zonder daar een cijfer aan te koppelen. Door dat een plek te geven in feedback sessies los van de inhoud. Kritiek op je werk, staat los van kritiek op jou. We branden jou niet af, we tonen juist respect door heel goed naar je werk te kijken. We zijn niet paternalistisch, als jij weet wat er slecht aan is, kan jij het verbeteren.

Ook vertellen we de studenten over ons eigen falen. En laten we ze kennis nemen van grote gefaalde projecten, en doordringen ze ervan dat deze projecten werden gedaan door hele competente engineers, minstens net zo competent als onze eigen studenten.

Ad) Ontwikkelen van leercompetenties
Als het komen tot goede software een leerproces is, een proces van falen en verbeteren, dan is het belangrijk dat we onze studenten een aantal leer

strategieën meegeven. In vrijwel alle vakken worden verschillende strategieën aangeleerd.

Het allerbelangrijkste is zorgen voor goede feedback op je werk. Daar waar mogelijk middels een adequaat proof of concept. Anders door genadeloze refutal: ervan uitgaan dat de oplossing fout is, en vanuit die gedachte de fouten aanwijzen. Ook review technieken krijgen bij ons veel aandacht.

Een tweede belangrijk punt is geënt op de hermeneutische cirkel. Je schrijft je ideeën op, werkt ze op papier uit. Laat ze even liggen en gaat er daarna weer mee aan de slag. Zo kom je op een bepaalde manier in gesprek met jezelf. Kan je verschillende perspectieven in ogenschouw nemen. Onze studenten schrijven veel, en we leren ze daar de nodige structuren voor aan.

Een derde aspect is zorgen voor divergentie. We zien dat veel studenten al snel een consensus zoeken, te snel stellen dat een ontwerp waar is en hun tijd besteed aan het uitwerken van dat idee. Divergentie heeft niet alleen nodig dat je actief verschillende meningen verzamelt, bijvoorbeeld door minder afgestemd en meer individueel te werken, maar ook dat je niet te veel hecht aan een bepaald idee. Daarbij is het cruciaal om daar niet teveel in te investeren. We stimuleren onze studenten om ideeën snel te schetsen, en oude ideeën achter zich te laten, zonder dat te ervaren als 'we moeten weer helemaal opnieuw beginnen".

Een ander belangrijk instrument voor divergentie is het actief ontwikkelen van meerdere scenario's. Ook dit komt bij ons actief aan bod.

Een vierde punt is het ontwikkelen van meta cognitie. Het herhaaldelijk monitoren van je eigen ontwerp proces. Actief twijfelen en reflecteren. Wat kan ik weten, wat weet ik niet, zit ik nog op het goede punt, wat is ook al weer mijn doel.

Ten slotte besteden we veel aandacht aan systematisch werken en het schakelen tussen het concrete en het abstracte. In het concrete kan je eenvoudiger zien of jouw beeld overeenkomt met de werkelijkheid. In het abstractie kan je sneller denken, synergie bereiken en tot een oplossing komen.

Afrondend, leren falen heeft het nodig dat de student voldoende zelfvertrouwen heeft, dat hij niet in de war raakt als hij een fout maakt. Dat wordt geholpen door fouten te zien als inherent aan het werk en studenten te belonen voor een werkwijze waarin ze snel ideeën ontwikkelen en hierin fouten weten te vinden en te fixen.

Hans Dekkers