Academische software engineers moeten brug slaan tussen onderzoek en praktijk

SOFTWARE ÍS SLEUTELTECHNOLOGIE

Informatica is niet als ‘topsector’ aangewezen maar als ‘actie-agenda’. Het vakgebied hobbelt achter allerlei andere gebieden aan, terwijl het de drijvende kracht is achter vrijwel elke activiteit. Software engineering is belangrijker dan ooit. Maar hoe komen we aan voldoende geschoolde software engineers? Paul Klint en Marko van Eekelen bespreken de mogelijkheden.

door: PAUL KLINT & MARKO VAN EEKELEN **beeld:** HOLLANDSE HOOGTE

De Wall Street Journal wees in april 2012 de software engineer aan als de beste baan in de VS. Andere bladen volgden. Terwijl de software engineer er in deze analyses goed uitkomt, moeten we ons met klem afvragen hoe we in Nederland voldoende academisch geschoolde software engineers kunnen opleiden. Het medio 2012 verschenen rapport ‘Software Technologies: The Missing Key Enabling Technology’ doet een pijnlijke constatering: software is de drijvende kracht achter vrijwel elke economische, maatschappelijke of wetenschappelijke activiteit, maar we zijn zowel op Europees als Nederlands niveau vergeten software als sleuteltechnologie te onderkennen. We zien dit ook terug in het globale Nederlandse onderzoekslandschap: Informatica is geen ‘topsector’ maar moet het doen met een ‘actieagenda’. Informatica hobbelt achteraan in de rij, achter tuinbouw, life sciences, chemie, water en vele andere gebieden. Binnen de informatica sluit software engineering de rij achter computational science, beeldherkenning, mobiele netwerken en dergelijke. Dat komt doordat tastbare toepassingen meer aanspreken dan de meer abstracte concepten uit de software engineering. Deze hekkensluiter is daarbij wel van cruciaal belang voor bestaan, vooruitgang en mogelijkheden van de meest uiteenlopende gebieden binnen en buiten de informatica. Door het snel toenemend gebruik van software om modelmatige analyses in velerlei wetenschapsgebieden toe te passen, ontstaat de verontrustende situatie dat de kwaliteit van een wetenschappelijk resultaat gelijk komt te staan aan de kwaliteit van de software om dit resultaat te bereiken. De uitkomst van een computersimulatie is immers even betrouwbaar als de kwaliteit van de gebruikte simulatiesoftware.

Verontrustend

Hoe heeft deze situatie kunnen ontstaan? Een universiteit of onderzoeksinstituut is tegenwoordig een open, transparante kennisgemeenschap die een breed scala van contacten onderhoudt. Een groot deel van het onderzoek wordt extern gefinancierd, voor een belangrijk deel met bijdragen vanuit het bedrijfsleven via subsidies, mede verkregen via NWO, STW of EU. Een prima basis voor innovatie en diffusie van onderzoekskennis, ook op het gebied van software, zou je denken. Het lijkt dus redelijk om te verwachten dat inhoudelijk ook bij informatica er een goede aansluiting is tussen wat op de universiteit wordt onderwezen en wat in het bedrijfsleven wordt toegepast. De AutomatiseringGids rapporteert echter regelmatig dat dit niet het geval is. Kees van Hee betoogde bij zijn afscheidsrede in 2011, (AG maart 2012), dat de inbreng van academische informatici bij grote IT-projecten nihil is. Theo en Hans Mulder betogen in een reeks AG-artikelen dat de kwaliteit van projectteams onder de maat is (december 2010), dat nieuwbouw risicovol is en dat moderniseren vaak spectaculaire voordelen kan bieden (maart 2011). Het geheel van bovenstaande waarnemingen is verontrustend: er is een grote vraag naar software engineers maar software wordt niet onderkend als sleuteltechnologie en de universiteiten leveren onvoldoende afgestudeerden in informatica op (inclusief software engineering) die een voor de praktijk relevante bijdrage kunnen leveren. Op welke manier kunnen de universiteiten verandering in deze situatie brengen?

Globale visie

We moeten een onderscheid maken tussen opleidingen die zich op informatica in het algemeen richten en opleidingen die zich speciaal met software engineering bezighouden. In de eerste categorie is software vooral een hulpmiddel en niet een onderwerp dat zelfstandig bestudeerd of onderwezen wordt. Bovendien ligt in algemene informatica-opleidingen de nadruk vaak op belangrijke fundamentele onderwerpen als complexiteit van algoritmen, logica en semantiek of op meer toepassingsgerichte onderwerpen als computational science, games en bio-informatica. Door deze nadruk verdwijnen factoren als schaalgrootte van software, het ontwikkelen in teams en software-evolutie uit het zicht. In opleidingen die zich speciaal op software engineering richten zijn dit bijzondere aandachtspunten in de opleiding.
Veel afgestudeerden van universitaire informatica-opleidingen zijn opgeleid voor het onderzoek en niet voor de praktijk. Ze zijn niet getraind in het analyseren van situaties die in de praktijk voorkomen. Het ontwerpen van de architectuur van een groot softwaresysteem is een goed voorbeeld dat laat zien dat academische vaardigheden van groot belang kunnen zijn voor de praktijk. Zo’n systeemontwerp lijkt erg op een academisch onderzoeksproject: in het begin zijn de doelen, randvoorwaarden, methoden en technieken vaag omdat de stakeholders nog niet weten wat ze willen. Door systematische exploratie kunnen eerst hypotheses geformuleerd en gevalideerd of verworpen worden om langzaam maar zeker tot een model te komen van een systeem waar de stakeholders zich in herkennen. Dit is de wetenschappelijke methode toegepast op een praktijkprobleem maar daar moet je dan wel in je opleiding op voorbereid zijn.
Naast het ontbreken van het besef dat de wetenschappelijke methode direct toepasbaar is in de praktijk hebben afgestudeerden van universitaire informatica-opleidingen ook te weinig technieken geleerd die rechtstreeks in de praktijk toepasbaar zijn om de problemen van de praktijk op te lossen. Daarbij gaat het er niet om of studenten wel de juiste programmeertaal hebben geleerd. Studenten leren programmeren in algemene zin – de programmeertaal is daarbij slechts een middel – en ze maken zich in korte tijd een nieuwe programmeertaal eigen. Dat is het probleem niet.
Het probleem is dat studenten tijdens hun opleiding geen globale visie hebben kunnen ontwikkelen over hoe ze hun academische en technische vaardigheden direct kunnen inzetten om praktijkproblemen op te lossen. Daarbij gaat het niet om de specifieke kennis van een programmeertaal of tool, maar om het herkennen van mogelijke oplossingspatronen in een probleem en het daarbij zoeken van technische oplossingen. Het probleem is dat de meeste universitaire informatica-opleidingen informatici opleiden en geen software engineers. Die informatici staan vaak verder van de praktijk omdat ze zich te veel in een specialistisch onderzoek gestort hebben.

Bruggenbouwers

Omdat academische software engineers uitstekende probleemoplossers en ‘bruggenbouwers’ tussen theorie en praktijk zijn, zouden universiteiten in het algemeen meer aandacht moeten besteden aan software engineering zowel in onderwijs als in onderzoek. Dit kan als onderdeel van een algemene informatica-opleiding of als een gespecialiseerde opleiding software engineering. Daarbij moet aandacht besteed worden aan requirements, ontwerp, implementatie, analyse, testen en evolutie van software. Te vaak worden deze onderwerpen in een algemeen informatica-curriculum afgedaan in één onderdeel van een groter college. Gelukkig zijn er ook positieve ontwikkelingen te melden.
Bij diverse universiteiten bestaat een track software engineering als onderdeel van een informaticamaster. De universiteit van Amsterdam heeft al sinds tien jaar een gespecialiseerde master software engineering. Bij de Open Universiteit is vorig jaar een vergelijkbare master van start gegaan.
Ander goed nieuws is dat universiteiten en onderzoeksinstituten niet alleen bezig zijn met onderzoeksprojecten maar ook een rol spelen in het adviseren en beoordelen van praktijkprojecten. Zo werken de Technische Universiteit Eindhoven en de Radboud Universiteit Nijmegen samen in LaQuSo (Laboratory for Quality Software) in projecten waarin de kwaliteit van softwareartefacten uit de praktijk worden beoordeeld met behulp van op onderzoek gebaseerde tools en technieken. De CWI-spin-off Software Improvement Group is commercieel erg succesvol in het meten van softwarekwaliteit. Dit zijn voorbeelden binnen onze eigen gezichtskring maar er zijn nog allerlei andere voorbeelden te noemen.
Als de universiteiten er serieus werk van maken om meer academische software engineers op te leiden, dan zal in de IT de smalle brug tussen universiteit en praktijk langzaam maar zeker steeds breder en steeds beter begaanbaar worden. «

BRUG THEORIE-PRAKTIJK

Om een permanente brug te slaan tussen universiteit en praktijk moet het curriculum software engineering constant gevoed worden door de laatste ontwikkelingen in het vakgebied. Zo is bijvoorbeeld de KAOS-methode van Axel van Lamsweerde zeer geschikt om ingenieurs op een gestructureerde technische manier hoogwaardige requirements te laten produceren. Er zijn geavanceerde analysetechnieken voor de correctheid van software die al enige tijd praktisch toegepast worden. Met name voor bedrijfs- en veiligheidskritische software (en welke software is dit eigenlijk niet?) is dit van vitaal belang. Domeinspecifieke talen, statische analyse, modelgebaseerd testen, model checking en theorem proving zijn technieken die hierbij het verschil kunnen maken. De studenten moeten de principes van deze verificatie- en validatietechnieken leren en zich de toepassing ervan door middel van tools eigen maken. Het garanderen van de security van IT-systemen vormt daarnaast een steeds grotere uitdaging in een wereld waarin deze systemen bijna dagelijks worden blootgesteld aan aanvallen. In staat zijn om kennis van een breed scala aan technieken voor analyse, constructie en security toe te passen is cruciaal om software te ontwikkelen die tegen al deze aanvallen bestand is. Ook technieken voor het onderhoud en de uitbreiding van bestaande software zijn onmisbaar voor een software engineer. Software evolueert immers voortdurend door toevoegingen en veranderingen. Academische tools zoals Rascal helpen daarbij zowel in het onderwijs als in de praktijk om systemen te doorgronden en te verbeteren.

Paul Klint is Research Fellow bij het Centrum Wiskunde & Informatica en hoogleraar Software Engineering (tevens opleidingsdirecteur van de Master Software Engineering) aan de Universiteit van Amsterdam.

Marko van Eekelen is hoogleraar Software Technologie (tevens programmaleider van de Master Software Engineering) aan de Open Universiteit. Bovendien is hij verbonden aan de Radboud Universiteit Nijmegen waar hij de ‘GipHouse’ Software Engineering-vakken en de Cyber Security Bachelor coördineert.

Voor reacties en nieuwe bijdragen van IT-experts:
Henk Ester,
070 3780397
h.ester@sdu.nl