De kennis van goed en kwaad

— Softwaretechnologie voor leren en onderwijs —

Johan Jeuring

24 september 2014
Utrecht

Rede in verkorte vorm uitgesproken bij de aanvaarding van het ambt van hoogleraar aan de Faculteit Bètawetenschappen van de Universiteit Utrecht met als leeropdracht Softwaretechnologie voor leren en onderwijs.

Abstract

De inzet van softwaretechnologie voor leren en onderwijs maakt een stormachtige ontwikkeling door. Veel hogeronderwijsinstellingen zijn bezig met het invoeren van verschillende vormen van blended learning. Voor het goed ondersteunen van het leren van studenten en het onderwijs van docenten met behulp van softwaretechnologie moet de technologie aan allerlei voorwaarden voldoen. Zo moet de softwaretechnologie goede feedback leveren, zoveel mogelijk automatisch, en moeten de aangeboden taken en de feedback eenvoudig aanpasbaar zijn, zowel aan de wensen van de docent, als aan het niveau van de student. Fundamentele concepten zoals talen, grammatica's, en algebra's leveren de basisonderdelen voor softwaretechnologie voor het ondersteunen van leren en onderwijs, maar het gebruik van deze concepten staat nog in de kinderschoenen. Het is heel moeilijk om softwaretechnologie op een goede manier in te zetten om het leren en onderwijs te ondersteunen, maar met mijn werk hoop ik samen met collega's en studenten te bewerkstelligen dat de softwaretechnologie niet meer het probleem is, en dat docenten zich kunnen concentreren op het leren van hun studenten.

24 September, 2014.

Mijnheer de Rector Magnificus,
Dames en Heren,

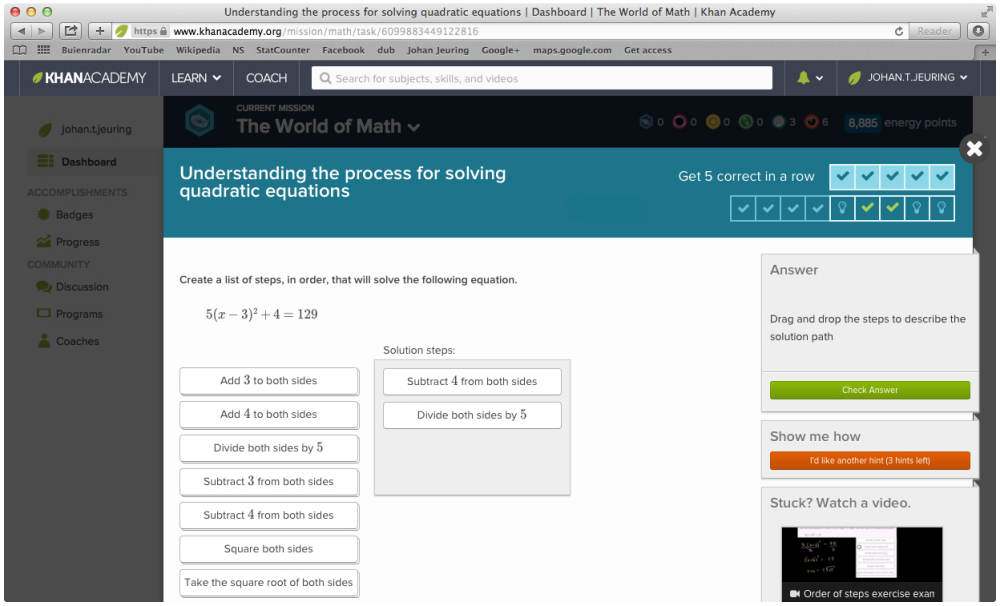
Op dit moment zijn veel hogeronderwijsinstellingen bezig met vernieuwingen van het onderwijs waarin een belangrijke rol voor ICT is weggelegd. Alleen al in Utrecht zijn de Universiteit Utrecht, het UMC Utrecht, en de Hogeschool van Utrecht bezig met miljoenenprojecten waarin het onderwijs meer gebruik gaat maken van blended learning [23]. Studenten kijken naar weblectures, slides en ander digitaal studiemateriaal, maken meer gebruik van digitale middelen bij het oefenen met de leerstof, en worden vaker digitaal getoetst.

De huidige digitale middelen bieden al veel ondersteuning voor studenten, maar aspecten zoals bijvoorbeeld het geven van individuele, geautomatiseerde feedback op het werk van een student, het geven van hints, het aanbieden van oefeningen die passen bij het niveau van een student, en het verklaren van de gevolgen van acties van studenten in een simulatie zijn nog vaak beperkt, gebrekkig, of afwezig.

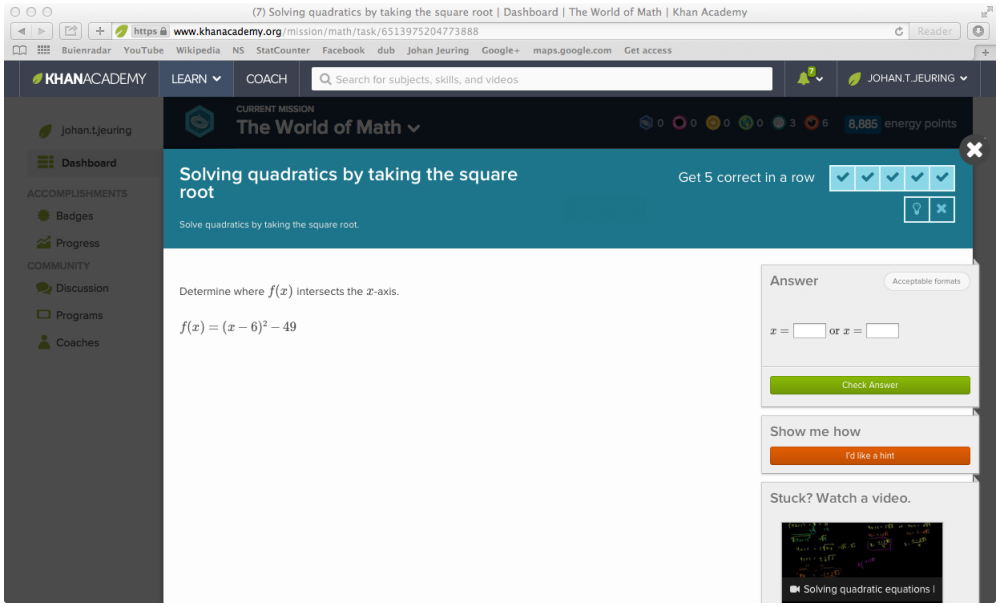
Ik zal vier voorbeelden geven van hoe bestaande digitale middelen verbeterd kunnen worden. Het eerste voorbeeld komt van de Khan Academy. De Khan Academy is een website die in 2006 begonnen is met het online zetten van korte video's waarin wordt uitgelegd hoe specifieke wiskundeopgaven worden oplost. In de afgelopen jaren is de Khan Academy enorm gegroeid, met meer dan vijfduizend video's over allerlei onderwerpen, niet alleen maar wiskunde, en heel veel oefenmateriaal. De website is ongelooflijk populair: met tien miljoen bezoekers per maand is het één van de onderwijswebsites met de meeste bezoekers wereldwijd. Stel een Nederlandse leerling in het voortgezet onderwijs wil oefenen met het oplossen van kwadratische vergelijkingen. Dan biedt de Khan Academy tenminste twee soorten opgaven aan. Zo kan de leerling oefenen met het oplossingsproces voor kwadratische vergelijkingen, zie Figuur 1[1]. In deze opgave moet de leerling de stappen die leiden tot een goede oplossing selecteren en in de juiste volgorde zetten. Khan Academy biedt ook wat foute stappen aan, zodat de leerling ook fouten kan maken. Dit is niet de manier waarop een leerling kwadratische opgaven oplost met behulp van pen en papier. Ik begrijp heel goed waarom de Khan Academy kiest voor deze oplossingsmethode: zo hoeft ze niet om te gaan met de talloze varianten die leerlingen gebruiken en fouten die ze maken bij het oplossen van een opgave. Dit was één van de grotere uitdagingen voor ons toen we strategieën voor het oplossen van wiskundeopgaven opstelden [33] voor de Europese Math-Bridge leeromgeving [68] en de Nederlandse Digitale Wiskunde Omgeving [20] (DWO) van het Freudenthal Instituut. De alternatieve manier van de Khan Academy om dit soort opgaven op te lossen ondersteunt mogelijk het leren, maar veroorzaakt wel een verschil met wat een leerling bij een toets moet doen, en zorgt er voor dat een leerling allerlei fouten niet kan maken die hij of zij met behulp van pen en papier juist wel maakt. Een minder bezwaar is dat de hint die de leerling krijgt wanneer hij of zij daarom vraagt onder de opgave verschijnt. Daar kwam ikzelf pas na een vijftal opgaven achter... In de opgave in Figuur 2 moet een leerling het eindantwoord van een berekening geven, en laat zien dat een leerling naast met de Khan Academy ook met pen en papier zal moeten werken om opgaven uit te werken.

Het tweede voorbeeld haal ik van een website die het leren programmeren ondersteunt. `code.org` is in januari 2013 van start gegaan, met als doel vooral kinderen tot achttien jaar te leren programmeren. Naar eigen zeggen hebben al bijna veertig miljoen mensen gebruik

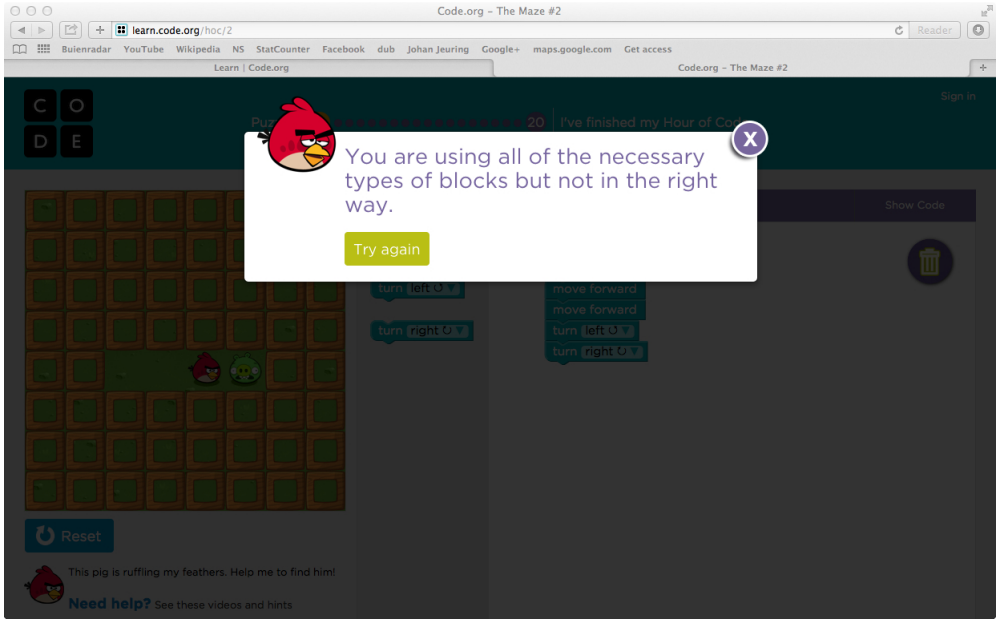
[1] All Khan Academy content is available for free at `www.khanacademy.org`.



Figuur 1: Khan Academy: kwadratische vergelijkingen oplossen: proces.



Figuur 2: Khan Academy: kwadratische vergelijkingen oplossen: eindantwoord.



Figuur 3: `code.org`: feedback bij de tweede Flappy Bird opgave.

gemaakt van de website. De tweede oefening die `code.org` aanbiedt is een opgave waarin een leerling een vogel naar een varken moet laten lopen. Kennelijk heeft het varken de veren van de vogel in de war gemaakt, en wil de vogel nu wraak nemen. Ik heb een niet volledig en wat onhandig programma daarvoor geschreven, en krijg de feedback in Figuur 3. Ik gebruik alle benodigde soorten blokken, maar niet op de goede manier? Die feedback zit er nogal naast. Ik gebruik de blokken die ik nodig heb voor de goede oplossing *wel* op de goede manier, ik loop met die blokken naar het varken toe. Maar ik gebruik ook blokken die ik niet nodig heb.

De laatste twee voorbeelden kom ik tegen wanneer ik (serious) games of simulaties speel. De afgelopen jaren heb ik onder andere Europa Universalis III[2] en Enercities, zie Figuur 4, gespeeld. In Europa Universalis bestuurt de speler een land, en neemt beslissingen over bestuur, diplomatie, religie, cultuur, handel, innovatie, en verdediging en oorlogsvoering. Het spel begint aan het einde van de vijftiende eeuw, en loopt een paar honderd jaar. Mijn probleem is dat ik elke keer dat ik het spel speel mijn land verlies in de eerste eeuw van mijn bestuur. Als rechtgeaard informaticus heb ik de ruim honderd pagina's documentatie gelezen, maar dat heeft mij nog niet voldoende geholpen. Ik zou bijvoorbeeld een 'replay' faciliteit met op specifieke momenten feedback zoals: hier koos je ervoor om Spanje een huwelijksvoorstel te doen, maar had je beter militairen kunnen sturen, of andersom, wensen, maar het spel geeft geen andere feedback dan het trage verval van het land dat ik bestuur. In het spel Enercities bestuurt de speler een stad. In dit spel verlies ik weliswaar niet mijn stad, maar komt mijn stad helemaal zonder energie te zitten, zonder verdere verklaring. Met de ervaringen van Europa Universalis in mijn achterhoofd heb ik mij na de eerste keer spelen verdere frustratie bespaard.

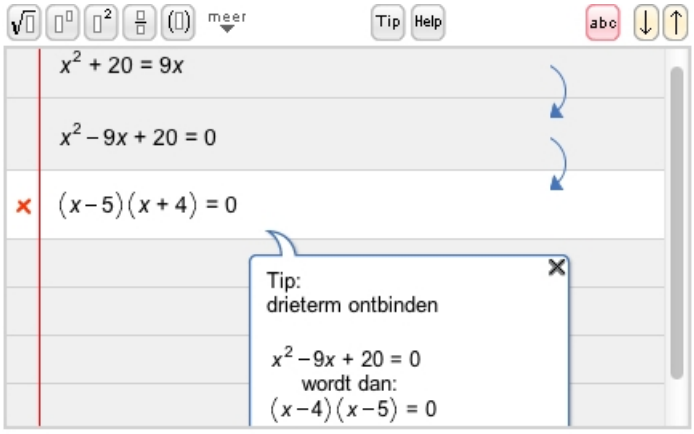
[2]Opgevolgd door `http://www.europauniversalis4.com`

Figuur 4: `http://www.enercities.eu`: stad zonder energie.

Vier populaire, en in het algemeen als uitstekend bekendstaande, toepassingen voor het ondersteunen van het leren in de wiskunde of de informatica, en het spelen met het bestuur, of energievraagstukken bieden opgaven aan die niet illustratief zijn voor de opgaven die leerlingen moeten oplossen in toetsen, geven gebrekkige of ronduit incorrecte hulp of feedback, of verklaren niet waarom mijn bestuur eindigt in rampspoed. Ik verwacht dat ik vergelijkbare problemen bij nog heel veel meer digitale middelen zal kunnen vinden.

Om betere digitale feedback te geven, om oefeningen op het niveau van een student te selecteren, en om docenten goed inzicht in de vaardigheden van hun studenten te geven moeten we soms lastige technische problemen oplossen. Fundamentele softwaretechnologische concepten zoals talen en grammatica's, algebra's, en recursie spelen een grote rol in het geven van feedback en hints in leeromgevingen en serious games. Als we goed gebruik maken van deze concepten kunnen we beter uitleggen wat een student heeft gedaan of moet doen, en kunnen we verklaren waarom, in plaats van constateren dat, een student minder goed heeft gepresteerd in een simulatie, bijvoorbeeld omdat de hele stad zonder energie zit. Dan kunnen we bijvoorbeeld de feedback in Figuur 5 automatisch genereren.

Voor het goed invoeren van vernieuwingen met behulp van ICT in het onderwijs moeten ook allerlei onderwijskundige en didactische problemen worden opgelost, maar een noodzakelijke voorwaarde voor dit soort vernieuwingen is dat de software goed werkt en makkelijk aan te passen is. Vandaag wil ik schetsen wat ik samen met collega's, promovendi, en studenten daar aan bij wil dragen.



Figuur 5: `http://ws.fisme.science.uu.nl/dwo/frameset.html`: een hint in de DWO.

1 Onderwijs aan de universiteit

Onderwijs is één van de twee door de wet voorgeschreven taken van de universiteit. Zonder studenten zouden we geen universiteit zijn. Wat doen we aan onderwijs? In het onderwijs voeren we drie soorten activiteiten uit:

- We bepalen de inhoud van een opleiding, en de cursussen die binnen een opleiding worden gegeven.
- We helpen studenten bij het leren van de inhoud van de opleiding en de cursussen door onderwijs te verzorgen.
- We toetsen dat studenten de noodzakelijke kennis en vaardigheden aan het einde van hun opleiding bezitten.

Om studenten te helpen bij het leren, en het leren te toetsen, hebben we kennis nodig van wat goed is, en wat nog moet worden verbeterd, en op welke manier. Als het goed is hebben een opleiding en haar docenten de benodigde kennis van goed en kwaad, en alles wat daar tussen in ligt.

Voor het streven naar de kennis van goed en kwaad straft God Adam, of eigenlijk de mannelijke mens, tot levenslang zwoegen (en vreemd genoeg Eva tot barenspijn). Of het een straf van God is weet ik niet, maar er bestaat geen enkele twijfel over dat we in het onderwijs hard zwoegen om onze doelen te bereiken. Vrouwen net zo goed als mannen. Uit een recent onderzoek onder Nederlandse werknemers [70] blijkt dat het ziekteverzuim in het onderwijs relatief hoog is, en dat als reden voor het ziekteverzuim significant vaker dan in andere sectoren werkdruk wordt genoemd. Mijn doel is om dit zwoegen voor zowel docenten als studenten te verlichten of effectiever te maken door gebruik te maken digitale middelen.

2 Leren

Hoe leren wij? Over dit onderwerp zijn boekenkasten volgeschreven, en sinds ik onderwijsdirecteur werd in 2007 heb ik daar een halve plank van gelezen [5, 9, 11, 27, 28, 42, 50, 54, 60, 61]. Hoewel er nog veel vragen te beantwoorden zijn, komt er uit wat er bekend is wel een patroon naar voren, en toen ik het elfde boek over leren en onderwijs las herkende ik veel van de eerdere tien boeken die ik had gelezen. Phil Race [61] stelt dat de volgende aspecten een centrale rol spelen bij succesvol leren in het hoger onderwijs.

- Willen leren. Een intrinsiek gemotiveerde student leert beter. Intrinsieke motivatie kan gestimuleerd worden door studenten echte problemen te geven, en door ze veel autonomie over hun eigen leren te geven.
- Leren met een doel. Extrinsieke motivatie, zoals de wens om een diploma te halen, ouders een trots gevoel te geven, goede cijfers te halen, of later veel geld te verdienen, is essentieel voor het leren.
- Leren door te doen. Een student leert kennis en vaardigheden door iets met die kennis en vaardigheden te doen.
- Feedback krijgen. Een student krijgt feedback van docenten, maar ook van andere studenten, en kan zelfs zijn of haar eigen werk corrigeren. Deze feedback geeft inzicht in de voortgang van de student.
- Reflecteren op het geleerde. Gedurende en nadat een student een onderwerp leert moet het een plek krijgen in het repertoir van de student. Hoe relateert het geleerde aan eerdere onderwerpen? Wat kan ik er nog meer mee? Reflectie maakt gebruik van alle eerder genoemde aspecten.

Mijn werk betreft vooral leren door te doen, en het geven van feedback.

3 Leren door te doen

We leren door te doen. Onze opleidingen en docenten zijn zich daar goed van bewust. Bij de opleiding Informatica, bijvoorbeeld, oefenen studenten door opgaven op te lossen bij de cursussen die wat dichter bij de theorie staan, programma's te schrijven, ontwerpen te maken, aan grotere projecten bij te dragen, presentaties te geven, en een onderzoek uit te voeren. Deze activiteiten passen uitstekend bij de leerdoelen van onze opleidingen. Ik denk dat we studenten nog vaker aan het werk moeten zetten, in plaats van naar een docent te laten luisteren, maar wanneer we ze aan het werk zetten geven we dat gemiddeld genomen op een goede manier vorm.

4 Feedback

Boud en Molloy [11] definiëren feedback als het proces waarmee een student informatie over zijn of haar werk krijgt zodat hij of zij de overeenkomsten met en verschillen tussen het eigen

werk en de gewenste standaarden voor een oplossing kan bepalen, met het doel een betere oplossing te construeren. Het doel van feedback is dus het verbeteren van het werk van een student. Wat voor soort informatie bevat feedback? Volgens Hattie en Timperley [29] bestaat feedback uit antwoorden op drie vragen:

- Waar ga ik naar toe? Wat is de relatie tussen het werk dat moet worden gedaan, en het uiteindelijke resultaat dat behaald moet worden? Wat zijn de leerdoelen die ik met deze taak geacht word te bereiken?
- Hoe doe ik het? Hoe staat het werk dat ik tot nu toe heb gedaan in relatie tot gewenste oplossingen?
- Hoe ga ik verder? Wat kan ik het beste als volgende stap doen? Aan welke taak kan ik nu het beste gaan werken?

Antwoorden op deze vragen worden vaak geïntegreerd gegeven. De eerste vraag gaat over de taak die opgelost moet worden, en waarom het oplossen van die taak leidt tot het behalen van de leerdoelen. De tweede vraag is de vraag die het vaakst met feedback wordt geassocieerd. De derde vraag, tenslotte, vraagt om hulp bij het verder werken aan de taak, waarbij een antwoord een hint in één of andere vorm is, of om hulp bij het bepalen van vervolgtaken. Dit soort feedback wordt ook wel eens feedforward genoemd. Antwoorden op bovenstaande vragen kunnen van verschillende aard zijn, en Narciss [53] onderscheidt meer dan tien verschillende soorten feedback. De precieze vorm van feedback doet voor deze rede niet zo ter zake, en ik zal het begrip feedback niet verder uitwerken.

Hoe geven we feedback aan onze studenten? Dat hangt af van de docent, de opleiding, en het soort werk dat de student uitgevoerd heeft. In de cursussen aan het begin van onze informatica-opleiding, waar vaak tussen de honderd en driehonderd studenten aan deelnemen, doen studenten en docenten vaak het volgende.

- Studenten maken opgaven tijdens werkcolleges, en kunnen daar feedback op krijgen van werkcollegebegeleiders. De ervaring leert dat een minderheid van de studenten aanwezig is bij de werkcolleges, en dat van die minderheid een minderheid om feedback vraagt.
- Studenten werken aan practicumopgaven, die vaak bestaan uit wat grotere programmeeropgaven. De studenten kunnen vragen stellen over hun werk en de opgave tijdens begeleidingssessies. De studenten leveren hun werk in, en krijgen meestal na één of twee weken een cijfer voor en feedback op hun opgave.
- Studenten maken twee tentamens. De tentamens worden meestal binnen twee weken nagekeken. De studenten kunnen hun resultaten inzien en feedback krijgen op hun werk, maar van die gelegenheid maken de studenten nauwelijks gebruik.

Een terugkerend thema in de gesprekken tussen de docenten bij de opleiding Informatica is dat studenten te weinig gebruik maken van de mogelijkheden om feedback te krijgen.

Hoe ervaren studenten de feedback die ze krijgen? Ik ben voorzitter van de Bama 3.0 begeleidingscommissie, een commissie die de invoering van een aantal universiteitsbrede onderwijsvernieuwingen bij de Universiteit Utrecht begeleidt. Om die invoering te begeleiden spreken we onder andere met studenten en docenten. Afgelopen februari hebben we twee

bitterballensessies georganiseerd met in totaal veertig studenten vanuit de hele universiteit. We hebben de studenten bevraagd op allerlei onderwerpen. Grosso modo waren de studenten tevreden; soms liepen de reacties nogal uiteen. Eén aspect sprong er in negatieve zin uit: de studenten vonden dat ze niet genoeg, te laat, of onbruikbare feedback kregen...

De analyse van Shute [66] laat zien dat het geven van feedback heel precies komt. De feedback moet over de taak gaan, en niet over de student. Soms is het beter om onmiddellijk feedback te geven, bijvoorbeeld als een taak als heel moeilijk wordt ervaren door een student, of wanneer de student werkt aan procedurele vaardigheden, en soms is het juist beter om even te wachten met feedback. Een student die minder goed presteert heeft eerder feedback nodig dan een student die goed presteert.

Goede feedback is belangrijk. Om dit soort algemene uitspraken over aspecten van onderwijs te ondersteunen worden meta-analyses gebruikt. Een meta-analyse neemt de resultaten van tientallen analyses samen, en berekent een soort grootste gemene deler. Omdat het onderzochte aspect bij heel veel scholen en leerlingen is getest, treedt het waarschijnlijk ook elders op. Uit meta-analyses blijkt bijvoorbeeld dat de grootte van een klas niet veel invloed heeft op de leerresultaten [28]. In het debat in de Tweede Kamer in april 2014 naar aanleiding van het burgerinitiatief "Stop de overvolle klassen" werd het werk van Hattie over meta-analyses meerdere malen aangehaald om de argumenten te ondersteunen. Het geven van feedback staat in de top 10 van interventies met het meeste effect in Hattie's werk.

Zowel docenten als studenten denken dat het geven van feedback beter kan. Er zijn verschillende manieren om de bovenstaande problemen met feedback aan te pakken. We kunnen ons onderwijs zo inrichten dat studenten de feedback die ze krijgen moeten gebruiken voor een verbeterde versie van hun werk [11]. Een andere mogelijkheid, die vaak zeer effectief is [28], is studenten zelf hun werk, of het werk van hun medestudenten, na te laten kijken. Een derde mogelijkheid is software te ontwikkelen waarmee studenten zoveel mogelijk automatisch kunnen controleren of ze op de goede weg zijn, waar ze naar toe moeten, en wat ze kunnen verbeteren. Hiermee kunnen we directe (of uitgestelde) feedback geven, zelfs bij grote aantallen studenten. Deze software moet dan op een verstandige manier in het onderwijs ingezet worden. In deze rede zal ik ingaan op welke concepten een rol spelen bij de ontwikkeling van zulke software, en hoe we zulke software kunnen ontwikkelen.

5 Software voor leren en onderwijs

Software kan helpen bij het leren van studenten, en bij het beoordelen van de kwaliteit van het werk van studenten. Uiteraard ben ik niet de eerste die deze gedachte heeft gehad. Al sinds de jaren zestig worden computers op scholen ingezet. In eerste instantie voornamelijk voor administratieve taken, maar later steeds meer om ook het leren te ondersteunen. Tegenwoordig worden computers en software voor allerlei taken gebruikt. Ik ben vooral geïnteresseerd in interactieve software waarin een student leertaken uitvoert, waarin het werk van een student beoordeeld wordt, en waarin een student om hulp kan vragen. Voorbeeldcategorieën van dit soort software zijn intelligente leeromgevingen [16] zoals de Khan Academy, de Digitale Wiskunde Omgeving en MathBridge voor het leren van wiskunde, serious games [1], zoals DragonBox en Mathbreakers voor rekenen, simulaties [39], zoals

Geogebra voor het modelleren en simuleren in de wiskunde [12], en digitale toetsomgevingen [62], zoals STACK [64]. Ik heb hier steeds wiskunde als voorbeelddomein gekozen, maar vergelijkbare software bestaat voor bijna ieder domein. Op dit moment komen studenten in plaats van met software voor leren en onderwijs, meer in aanraking met faciliterende software in het onderwijs, zoals de leerplatformen Blackboard en Moodle, een evaluatiesysteem zoals Caracal, en inleversystemen zoals Submit en DomJudge. Hoewel dit hele nuttige systemen zijn, zijn de softwaretechnologische uitdagingen bij deze systemen grotendeels opgelost, en zijn ze voor mijn onderzoek dus minder interessant.

Er zijn verschillende redenen om interactieve software voor het leren te gebruiken. Zo kan deze software een student aangepaste en precieze feedback geven, op ieder tijdstip van de dag. De Saxion hogeschool heeft haar studenten gevraagd wat ze willen met ICT in het onderwijs. De 2675 respondenten willen vooral gebruik maken van oefentoetsen om te zien waar ze staan, en ze zien als groot voordeel dat ze direct feedback krijgen op hun toetsen[3]. Een andere reden is het motiveren van studenten, waarvoor vaak games worden gebruikt. Verder zijn er taken waar een student mee moet oefenen die lastig of niet zonder simulatiesoftware geoefend kunnen worden. Oude voorbeelden hiervan zijn flightsimulators, maar dit geldt ook voor een vak als internationale relaties, of crisismanagement [2, 73]. We willen Utrecht niet laten overstromen om studenten de kans te geven om te gaan met een crisis die bestuur, bedrijvigheid en bewoners treft. Technieken die in interactieve software voor het leren worden gebruikt kunnen vaak ook worden ingezet om het werk van studenten na te kijken.

6 Het effect van software voor leren en onderwijs

Wordt het onderwijs beter van het gebruik van interactieve software voor leren? De afgelopen decennia zijn er duizenden experimenten uitgevoerd die het effect van het gebruik van technologie in het onderwijs onderzoeken. Alleen al voor het reken- en wiskundeonderwijs zijn er honderden studies te vinden die kijken naar hoe goed een bepaalde technologie voor het wiskundeonderwijs werkt. Wat zeggen die studies?

De effecten van het gebruik van computerapplicaties in het rekenonderwijs variëren nogal. Marjoke Bakker is recent gepromoveerd op een proefschrift waarin zij het effect bestudeert van het gebruik van minigames waarin wordt geoefend met vermenigvuldigen en delen in de onderbouw van het primaire onderwijs [6]. Deze games werken het best wanneer leerlingen thuis spelen met de games, en de resultaten daarna in de klas besproken worden. Alleen in de klas spelen met de games had nog steeds een positief, zij het wat minder, effect. Dit werkte beter voor jongens in groep 3, en voor meisjes in groep 4. Alleen maar thuis spelen met de games, zonder nabespreking in de klas, liet geen leereffect zien. Dit onderzoek geeft al aan dat leereffecten van onderwijstechnologie afhangen van allerlei factoren.

Uit de meta-analyse van Cheung en Slavin naar het gebruik van technologie in het reken- en wiskundeonderwijs in het primair en voortgezet onderwijs [13] blijkt dat deze toepassingen een gemiddeld effect van 0.15 hebben. Dit is een zeer bescheiden effect, nog lager dan klasgrootte (0.21). De precieze betekenis van de effectwaarden is voor deze rede niet belangrijk,

[3] `https://www.surfspace.nl/media/bijlagen/artikel-1582-7634ff6e299060d0759739f8798fc024.pdf`.

maar volgens Hattie zou een onderwijvernieuwing tenminste een effect van 0.4 moeten hebben. Als het artikel van Cheung en Slavin in handen van onze Tweede Kamerleden komt zou er zomaar een ban op investeringen in het gebruik van technologie voor het reken- en wiskundeonderwijs kunnen worden ingevoerd.

Cheung en Slavin analyseren 74 studies, waarin de effecten van tientallen verschillende systemen worden beschreven. Ze selecteren scherp: studies waarin geen controlegroep is meegenomen, die te kort duren, of waarin niet gekeken wordt naar de competenties van de studenten aan het begin van de studie, worden niet meegenomen in de meta-analyse. Het enige aspect waar ze helemaal niet naar kijken is de kwaliteit of functionaliteit van de gebruikte applicaties. Zo zijn er studies van systemen met een zeer groot leereffect (+1.20), en studies van heel andere systemen met een negatief leereffect (-0.26). Als je alleen maar het gebruik van applicaties in het rekenonderwijs meet, dan krijg je dus een zeer matig gemiddeld effect. Het samennemen van resultaten van verschillende applicaties voor het leren van wiskunde en rekenen met zulke verschillende effecten heeft echter een groot probleem: je kunt er geen enkele conclusie uit trekken. Vergelijk dit maar eens met een studie naar de kosten van behandelingen van een maagzweer in de afgelopen decennia. Voor de tachtiger jaren was een ingrijpende operatie de belangrijkste behandelingsmethode. In 1982 werd bekend dat een maagzweer ook met antibiotica en zuurremmers kan worden behandeld. De gemiddelde kosten van een maagzweerbehandeling liggen dus hoog, maar we weten dat een maagzweer nu tegen relatief lage kosten kan worden behandeld. In feite nemen Cheung en Slavin appels en peren samen, en proberen iets over het gemiddelde te zeggen. Deze kritiek op meta-analyses wordt door Hattie [28] weggewoven met de opmerking dat als je alleen maar fruit hebt, je niets anders kan doen. Bij het gebruik van technologie in het (reken)onderwijs bestaat ieder soort fruit echter weer uit zoveel verschillende aspecten, dat het niet zinvol is om verschillende soorten fruit met elkaar te vergelijken. Cheung en Slavin hebben een aantal kwalitatief goede studies verzameld, maar hun conclusie is zonder waarde.

In een andere meta-analyse vergelijkt VanLehn de effectiviteit van menselijke tutoren met de effectiviteit van intelligente leeromgevingen [72]. De leeromgevingen vervangen de docent niet, maar worden gebruikt bij het thuis of op school oefenen met de stof. In Nederland wordt een aantal van dit soort leeromgevingen voor wiskunde aangeboden en gebruikt: de DWO van het Freudenthal Instituut, MathDox van de TU/e, het SOWISO-platform, en verschillende applicaties van EduHint. VanLehn selecteert applicaties gebaseerd op wat voor soort ondersteuning ze bieden voor het leren, en niet op het inhoudelijke domein waarvoor ze worden gebruikt. Hij kijkt in de analyse vooral naar de grootte van de stappen die een student in de leeromgevingen kan maken: kan een student alleen een antwoord op een vraag geven, en krijgt zij feedback gebaseerd op dat antwoord, of kan een student een oplossing stapsgewijs construeren, en kan iedere stap van feedback worden voorzien (tijdens of na afloop van het werken aan de taak). Als je iets wilt zeggen over het *leer*effect van applicaties, dan moet je applicaties selecteren op basis van hoe ze *leren* ondersteunen. In tegenstelling tot het werk van Cheung en Slavin is deze keuze wel zinvol. De intersectie van de meta-analyses van Cheung en Slavin en VanLehn is beperkt, en bestaat uit een viertal studies. Deze studies worden door VanLehn juist niet meegenomen, omdat de controlegroep met andere tekstboeken werkte, en VanLehn alleen variatie wilde op de stapgrootte van de interactie. Uit VanLehn's meta-analyse blijkt, enigszins verrassend, dat een intelligente leeromgeving die

de stappen van een student volgt bijna even effectief is als een menselijke tutor (0.76 versus 0.79).

Bij het kijken naar de effectiviteit van het gebruik van technologie in het onderwijs is het belangrijk na te gaan wat er precies wordt vergeleken, en hoe je de resultaten van een analyse moet interpreteren. De leereffecten van applicaties voor een specifiek domein samennemen, zoals Cheung en Slavin doen voor het reken- en wiskundeonderwijs, is zinloos. De leereffecten samennemen van applicaties die leren op een vergelijkbare manier ondersteunen is zinvol, en laat bijvoorbeeld zien dat technologie die een leerling volgt bij het stapsgewijs oplossen van taken zeer effectief is.

7 Uitdagingen voor interactieve software voor leren

Wat voor uitdagingen liggen er nog bij de ontwikkeling van interactieve software voor leren? Ik presenteer eerst een aantal uitdagingen die voortgekomen zijn uit een bijeenkomst van het Europese Stellar Network of Excellence, specifiek over uitdagingen in het leren met behulp van interactieve software in La Clusaz in Frankrijk in 2011 [22]. Onder het thema 'Leerarrangementen' (orchestrating learning) worden onder andere de volgende Grand Challenge Problems (GCPs) genoemd:

- GCP7: Assessment and Automated Feedback. Deze uitdaging heeft als doel toetsing en geautomatiseerde feedback voor meer domeinen te ontwikkelen, naast voor summatieve toetsing (om een cijfer te bepalen) ook voor formatieve toetsing (om te oefenen) te gebruiken, en ook buiten de traditionele toepassing van multiple choice vragen te gebruiken.
- GCP8: One Informed Tutor per Child. Software kan vorderingen van studenten inzichtelijk maken, en een tutor kan deze informatie gebruiken om gericht hulp te bieden aan individuele studenten.
- GCP9: Improving Educational Practices Through Data-Supported Information Systems. De data die door interactieve software dat leren ondersteunt wordt opgeslagen kan goed gebruikt worden om leerprocessen te optimaliseren. Maar daarvoor moet wel relevante data worden opgeslagen en gecombineerd.

Het thema 'Leren in Context' bevat GCP16: Engaging the Brain's Reward System. Het doel van dit thema is te bestuderen hoe we met behulp van games kunnen leren, en hoe we games kunnen ontwikkelen die het leren ondersteunen. De belangrijkste vraag hier is hoe onze hersenen reageren op verschillende soorten feedback, en hoe dat het leren beïnvloedt.

Mijn werk richt zich vooral op GCP7: toetsing en geautomatiseerde feedback. De data die voor toetsing en geautomatiseerde feedback wordt verzameld en geanalyseerd kan ook voor de tutoren voor kinderen en voor het verbeteren van onderwijsprocessen worden gebruikt, en daarmee kunnen de resultaten van GCP7 worden ingezet om een bijdrage te leveren aan GPC8 en 9. Om de vragen over het gebruik van games voor het leren te kunnen beantwoorden moeten we verschillende soorten feedback aan spelers van games geven. Dit thema past uitstekend bij het Game Research focusgebied van de Universiteit Utrecht, waaraan we met het departement Informatica een belangrijke bijdrage leveren.

De voorbeelden die ik aan het begin van deze rede gaf laten zien dat de huidige technologie leerlingen niet altijd de gewenste interactieve opgaven, of de meest precieze, of zelfs correcte, feedback geeft. Op dit gebied kan nog het nodige verbeterd worden.

Mijn bijdrage op het gebied van softwaretechnologie voor leren en onderwijs zal vooral op het technische vlak liggen. Voor het oplossen van didactische of organisatorische problemen werk ik graag samen met mensen uit de onderwijskunde, vakdidactiek, beleid & management, enzovoort.

Ik vertaal de bovenstaande uitdagingen in de volgende vragen:

> Hoe kan ik automatisch precieze feedback geven aan studenten?

Hoe vergelijk ik een mogelijk onvolledig studentproduct met een gewenste oplossing? Kan ik herkennen wat een student gedaan heeft in twee opeenvolgende stappen? Hoe kan ik een student helpen bij het nemen van de volgende stap op weg naar een goede oplossing?

Feedback specificeren voor een multiple choice vraag is eenvoudig. Feedback specificeren bij een opgave die stapsgewijs wordt opgelost, zoals het oplossen van een kwadratische vergelijking, is al veel lastiger. Feedback specificeren bij een opgave waarin een student een probleem moet modelleren is nog weer een factor lastiger. Le, Loll en Pinkwart [43] beschrijven klassen van problemen afhankelijk van hoe moeilijk het is om correctheid van een oplossing automatisch te beoordelen.

> Hoe kan ik softwaretechnologische concepten gebruiken om softwaretechnologie voor leren en onderwijs makkelijker te ontwikkelen, gebruiken, en aan te passen?

Een meer algemene uitdaging aan interactieve software voor leren is het flexibel aanpassen aan veranderende omstandigheden. Docenten willen nieuwe of andere taken aanbieden, taken op een alternatieve manier oplossen, wel feedback, geen feedback, een beetje feedback, of andere feedback geven, en informatie over activiteiten van een student in een leeromgeving op verschillende manieren ontvangen. 70% van een groep van bijna honderd docenten van online cursussen paste hun opdrachten regelmatig aan [47]. Anderson et al. [3] noemen de gebrekkige aanpassingsmogelijkheden als een belangrijke reden waarom de door hen ontwikkelde intelligente leeromgevingen niet snel door andere docenten werden gebruikt. Tenslotte, Bokhove en Drijvers [10] geven aanpassingsmogelijkheden voor docenten als één van de vier fundamentele eisen aan leeromgevingen voor wiskunde. Afhankelijk van het niveau van een student is het beter om wel of niet onmiddellijk feedback te geven, hints ter beschikking te stellen, en de grootte van de stappen die een student kan zetten groter of kleiner te maken. Ontwikkelaars van leeromgevingen stellen taken samen uit verschillende onderdelen, en willen mogelijk sommige onderdelen vervangen door andere. Verandering en aanpassing van interactieve software voor leren moet al bij het ontwerp meegenomen worden [58], en moet ook in de uiteindelijke software eenvoudig zijn.

Een volwassen leeromgeving voor wiskunde bevat al gauw duizenden opgaven. Bij ieder van die opgaven expliciet feedback specificeren is heel veel werk, zeker als de opgave in meerdere stappen wordt opgelost. Automatisch feedback genereren voor klassen van stapsgewijze opgaven, zoals de klasse van kwadratische vergelijkingen, of de klasse van opgaven over matrices vegen, reduceert de benodigde hoeveelheid werk voor het ontwikkelen van een leeromgeving aanzienlijk. Ook maakt automatische generatie van feedback het makke-

lijker om feedback aan te passen aan de gebruiker. En als we een docent willen ondersteunen bij het aanpassen van feedback, dan moet feedback apart gerepresenteerd worden.

De eerste twee vragen betreffen het gebruik van softwaretechnologische aspecten om betere technologie voor leren en onderwijs te ontwikkelen. De derde vraag gaat over de bijdrage van het toepassingsdomein aan de methode.

> Hoe moet softwaretechnologie verder worden ontwikkeld voor een betere ondersteuning van software voor leren en onderwijs?

Software voor leren en onderwijs stelt grote eisen aan de technologie, en voor een aantal problemen zal het nodig zijn die technologie verder te ontwikkelen. Als we bijvoorbeeld een student feedback willen geven over waar precies een fout in het werk zit, dan moeten we eigenschappen waar een goede oplossing aan moet voldoen vertalen naar eigenschappen waar onderdelen van zo'n oplossing aan moeten voldoen. Dit zijn vragen die ook onafhankelijk van softwaretechnologie voor leren en onderwijs worden gesteld, maar daar vaak minder centraal staan. Softwaretechnologie voor leren en onderwijs stelt specifieke eisen aan oplossingen, die mogelijk later ook van nut zijn voor andere toepassingsgebieden.

8 Softwaretechnologie

Software ondersteunt ontzettend veel menselijke activiteiten, gerelateerd aan communicatie, financiën, gezondheid, spelen, leren; eigenlijk alles wat we doen. Wereldwijd ontwikkelen tientallen miljoenen mensen deze software. Technologie voor het ontwikkelen van software bestaat uit softwaremodelleertechnieken, softwareontwikkelingsmethoden, programmeer- en domein-specifieke talen, raamwerken, en nog veel meer. Hoewel de revolutionaire ontwikkeling van toepassingen van software anders doet vermoeden, evolueert de onderliggende technologie voor het maken van software niet heel snel. Voor mijn dagelijkse softwareontwikkeling gebruik ik nog steeds dezelfde programmeertaal als twintig jaar geleden, de functionele programmeertaal Haskell. Die taal is wel vernieuwd door de jaren heen, maar pas nu wordt die taal ook buiten de academische wereld als een veelbelovende programmeertaal gezien, en worden fundamentele concepten die al meer dan twintig jaar bekend zijn, zoals parametrisch polymorfisme en anonieme functies ook toegevoegd aan programmeertalen die door de meeste softwarebedrijven worden gebruikt. Vooral investeringsbanken die het uiterste vergen van snelheid en correctheid van software gebruiken nu functionele programmeertalen. Begin deze maand heb ik de grootste jaarlijkse conferentie over functioneel programmeren georganiseerd, onder andere met sponsorgelden van Facebook, Twitter, Google, Microsoft, en Oracle.

Het ontwikkelen van software voor een grote toepassing is uitdagend. Vaak gaat het goed, en het is indrukwekkend om te zien wat voor software er tegenwoordig ontwikkeld wordt. Maar het gaat ook nog af en toe fout. De parlementaire onderzoekscommissie naar automatiseringsprojecten bij de overheid, eufemistisch genaamd 'Tijdelijke commissie ICT', werkzaam sinds april 2014, vermoedt dat er miljarden euros geïnvesteerd zijn in mislukte automatiseringsprojecten.

De oorzaken van problemen met software liggen op heel veel verschillende terreinen, zoals

niet goed gedefinieerde requirements, slecht projectmanagement, slechte of geen communicatie tussen gebruikers, klant, en ontwikkelaars, of commerciële druk. Een aantal problemen zijn direct gerelateerd aan de technologie, zoals slordige ontwikkelpraktijken en een gebrek aan controle over de complexiteit van een project. Voorbeelden van slordige ontwikkelpraktijken in de ontwikkeling van software zijn programma's waarin stukken programma gekopieerd zijn, programma's die zo veel doen dat een nieuw bij het project betrokken programmeur onmogelijk uit kan vinden wat de effecten van een wijziging zijn, en programma's waar de code ver afstaat van het doel van de code.

Kwalitatief goede programma's zijn modulair opgezet, en zijn makkelijk te combineren en aan te passen [36]. Technieken waarmee zulke programma's geschreven kunnen worden zijn:

- hogere-orde functies, die hergebruik op een goede manier ondersteunen, en bijvoorbeeld veel voorkomende recursieve patronen faciliteren,
- datatypes, die we gebruiken om data te structureren, maar ook voor het beschrijven van de abstracte syntax van talen,
- domein-specifieke talen, die we gebruiken om programma's voor specifieke domeinen of problemen te beschrijven, en
- algebra's, die we gebruiken om te verifiëren of een programma of een taal aan één of meerdere eigenschappen voldoet.

9 Feedback en taal

Om de technieken voor Assessment and Automated Feedback te verbeteren maken we gebruik van een aantal fundamentele methoden uit de Informatica, met name talen en grammatica's. Talen en grammatica's zijn uiteraard niet uniek voor Informatica, en worden binnen andere vakgebieden al veel langer bestudeerd. Wat wel uniek is voor de Informatica, is de focus op technieken voor het automatisch herkennen, verwerken, en vertalen van zinnen.

Taal (en, daaraan gerelateerd, recursie [15]) is een unieke eigenschap van mensen. Geen andere diersoort heeft een communicatiesysteem dat werkt zoals menselijke taal, en dat in de buurt komt van de communicatiemogelijkheden die wij hebben. Maynard Smith en Szathmáry [49] onderscheiden acht transities in de evolutionaire geschiedenis van het leven. Taal is de laatste grote transitie. Kirby [41] stelt dat door evolutie binnen taal zelf compositionaliteit en grammatica's zijn ontstaan, en dat die evolutie gedreven wordt door het leren. Een compositionele taal die wordt beschreven met behulp van een grammatica kunnen we makkelijker leren. Communicatie is echter niet alleen een groot succesverhaal: "although the assumption of most people is that understanding is the default, it is probably wiser to consider misunderstanding the default, and successful understanding the accomplishment" [18]. Een belangrijke reden waarom we elkaar zo vaak niet begrijpen is dat we taal interpreteren in ons eigen referentiekader, en dat kader is verschillend voor verschillende mensen. 'How people learn' [54] geeft dit verschil in achtergrondkennis als een belangrijk aspect waar we rekening mee moeten houden bij het leren van mensen.

Natuurlijke concepten die in het beschrijven van talen naar voren komen zijn:

- woorden: woorden vormen de basiscomponenten van zinnen.
- keuze: een werkwoordelijk gezegde is een persoonsvorm *of* een persoonsvorm met een of meer andere werkwoordsvormen.
- sequentie: een bijzin bestaat uit een subject *gevolgd door* een object *gevolgd door* een werkwoord.
- permutatie: 'ik heb een appel aan Jonathan gegeven' en 'ik heb aan Jonathan een appel gegeven' mogen allebei.

Taal en grammatica's geven structuur aan informatie. Andersom kan gestructureerde informatie beschreven worden met behulp van taal en grammatica's. Bij het beschrijven van hoe we problemen oplossen maken we gebruik van vergelijkbare aspecten als die we gebruiken voor het beschrijven van taal. In de wiskunde lossen we een kwadratische vergelijking op door eerst alle termen naar links te brengen, en dan de linker expressie te factoriseren als de termen mooie factoren hebben, of anders de abc-formule toe te passen. In deze beschrijving zijn de basisacties het naar links brengen van termen, of het factoriseren van een expressie, hebben we een keuze tussen factoriseren of de abc-formule gebruiken, en zit er een volgorde in eerst de termen naar links brengen, en daarna factoriseren of de abc-formule toepassen. Volgens de psychologen (en informatici) Newell en Simon [55] kan een mens die een probleem oplost gezien worden als een informatieverwerkend systeem. De literatuur over technologie voor leren gebruikt wel 'plans' of 'model-tracing' voor het beschrijven van hoe een probleem wordt opgelost [4, 65], maar de relatie met talen en grammatica's, en, gerelateerd, parsers, processen, en algebra's is beperkt gebleven.

Om automatisch te volgen wat een student doet wanneer hij of zij aan een opgave werkt, en hoe het onvolledige werk van de student vergelijkt met een goede oplossing, moeten we weten hoe een student een goede oplossing stapsgewijs construeert. Hetzelfde geldt voor het uitvoeren van acties in een simulatieomgeving of een serious game. Idealiter kan een student iedere wenselijke manier om een opgave op te lossen gebruiken, en kan de software de student volgen, en indien gewenst hints geven of verbeteringen suggereren. Om dit mogelijk te maken, beschrijven we expliciet hoe een opgave wordt opgelost. Voor die expliciete beschrijving hebben we een taal nodig. Deze taal is een domein-specifieke taal voor het beschrijven van oplossingsstrategieën. De Act-R groep van de Carnegie-Mellon Universiteit heeft decennia lang gewerkt aan het ontwikkelen van model-tracing technieken voor intelligente leeromgevingen [3]. Voor de implementatie van de intelligentie van de leeromgevingen gebruiken ze de functionele programmeertaal Lisp, en de constructies die voor het ontwikkelen van de omgevingen worden gebruikt kunnen worden beschouwd als een domein-specifieke taal voor het beschrijven van oplossingen.

In eerder werk hebben wij verder gebouwd op het Act-R werk, en een domein-specifieke taal met constructies specifiek voor het beschrijven van oplossingsstrategieën, zoals keuze, sequentie, falen, en slagen, geconstrueerd [34, 33]. De domeinspecifieke taal bevat constructies om basisstappen, zoals een herschrijfstap of een verfijningsstap, te combineren. We hebben deze domein-specifieke taal gebruikt om zogenaamde domain reasoners [26] te ontwikkelen. De literatuur over intelligente tutoringsystemen beschrijft een domain reasoner als een 'aparte component die kan redeneren over de problemen in een domein' [16], een

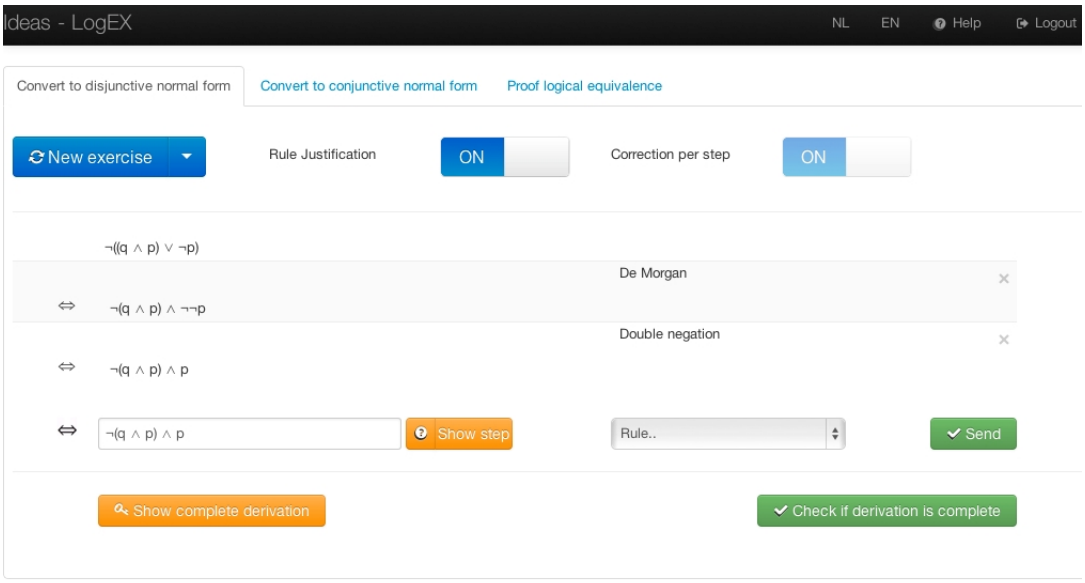
```
quadraticStrategyG :: IsTerm a => LabeledStrategy (Context a)
quadraticStrategyG =
   label "Quadratic Equation Strategy" $ repeatS $
   -- Relaxed strategy: even if there are "nice" factors, allow use of quadratic formula
      somewhere (generalForm <|> generalABCForm)
      |> somewhere zeroForm
      |> somewhere constantForm
      |> simplifyForm
      |> topForm
 where
   -- ax^2 + bx + c == 0, without quadratic formula
   generalForm = label "general form" $ ...

   ...
```

Figuur 6: Deel van de code van de domain reasoner voor kwadratische vergelijkingen

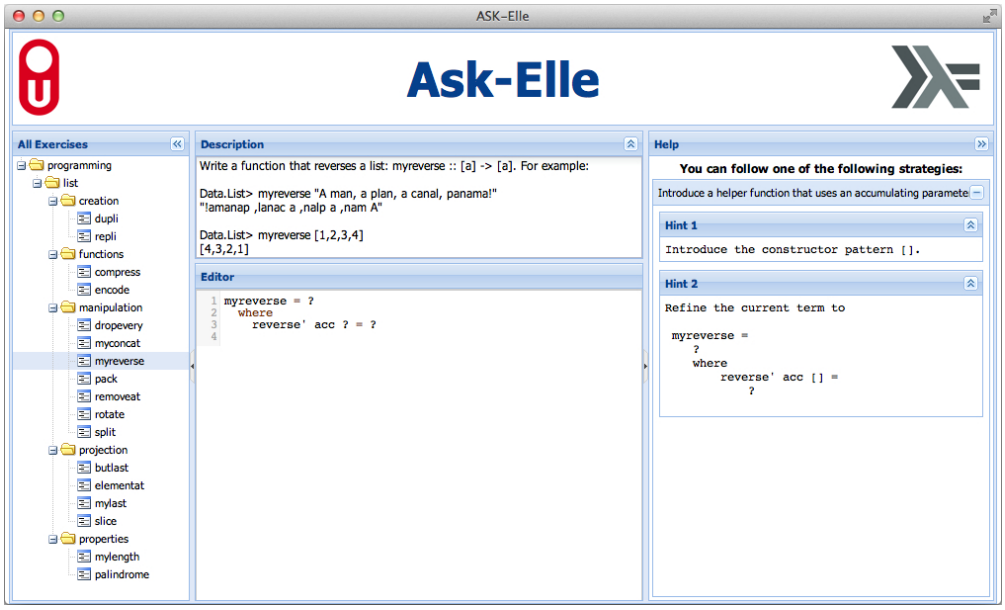
'module met expertkennis' [56], of een 'domeinexpert' [67]. We hebben bijvoorbeeld domain reasoners ontwikkeld voor het oplossen van kwadratische vergelijkingen (zie Figuur 6 voor een voorbeeld van hoe een gedeelte van de code er daarvoor uitziet), voor het oplossen van lineaire vergelijkingen, het herschrijvingen van logische expressies naar disjunctieve normaalvorm [46, 45] (zie Figuur 7 voor een screenshot van een leeromgeving waarin deze domain reasoner wordt gebruikt), het vegen van een matrix, het construeren of herschrijven van programma's in Haskell [38, 25, 57] (zie Figuren 8 en 9), en nog een twintigtal meer domeinen. We gebruiken domain reasoners in onze feedbackservices [24, 32], die aangeroepen worden door externe tools om automatische feedback te genereren wanneer studenten opgaven binnen de genoemde domeinen oplossen.

Ons werk aan een domeinspecifieke taal voor het volgen van interacties van studenten heeft onder meer geleid tot nieuwe combinatoren in softwarebibliotheken voor ontleden. Bij het oplossen van problemen komt het vaak voor dat je twee deelproblemen op moet lossen, maar de volgorde waarin deze problemen opgelost worden is niet belangrijk. Als je bijvoorbeeld de vergelijking $x^2(2x^2 - 1) = 4(2x^2 - 1)$ op wilt lossen, dan doe je dat door de twee vergelijkingen $x^2 = 4$ en $2x^2 - 1 = 0$ op te lossen. De volgorde waarin je die vergelijkingen oplost doet er niet toe. Voor dit doel hebben we de 'interleave' combinator geïntroduceerd. Dit is een constructie die in ontleders voor (programmeer)talen nauwelijks voorkomt. Een variant van de oplossing die we voor het specifieke probleem van het volgen van interacties voor studenten hebben ontwikkeld [31] is later toegevoegd aan algemene softwarebibliotheken voor het ontleden [69].

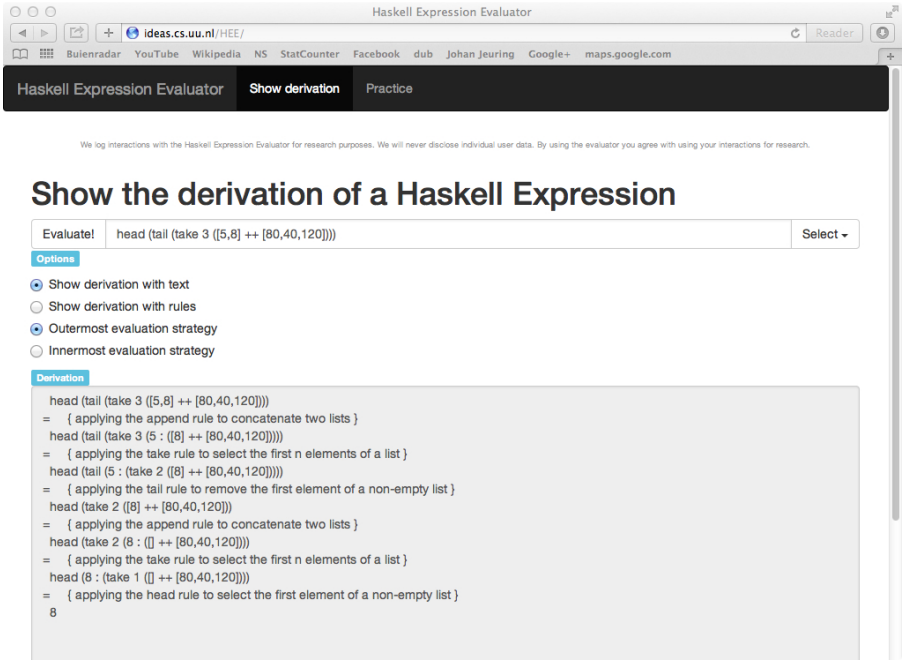
We staan pas aan het begin wanneer het gaat om het beschrijven van oplossingsstrategieën voor problemen. Een interessant probleem waar we al lang over hebben nagedacht, maar pas nu vat op beginnen te krijgen, is het probleem van het aangeven van voorkeuren in oplossingsstrategieën. Docenten vinden sommige oplossingen beter dan andere ('als je kunt factoriseren, doe dat, en pas anders de abc-formule toe'), en in een serious game verdien je met sommige acties meer punten dan met andere. Het is niet moeilijk om een taalcombinator te introduceren die aangeeft dat het linker argument de voorkeur verdient boven het rechter argument. Maar het is wel uitdagend om deze combinator samen met andere combinatoren



Figuur 7: Leeromgeving voor propositionele logica.



Figuur 8: Aske-Elle: construeer een Haskell programma.



Figuur 9: Haskell Expression Evaluator: herschrijf een Haskell programma.

te gebruiken. Stel we hebben de drie oplossingsstrategieën p, q, en r, en we willen p of q toepassen, met een voorkeur voor p, en daarna r. Als r faalt na p, maar niet na q, dan is de te gebruiken strategie dus q gevolgd door r. Maar als we niet vooruit kijken naar r, dan kunnen we deze beslissing niet nemen. We willen niet graag de hint geven dat een gebruiker p moet toepassen, om na toepassing van p er achter te komen dat we op een dood spoor zitten. Het lijkt er dus op dat we helemaal naar het einde van een oplossing moeten kijken om een student een hint te geven voor de volgende stap. Dit wordt backtracking genoemd, en is vanuit een computationeel standpunt onwenselijk. Het is van groot belang om de algebraïsche eigenschappen van de taalcombinatoren die we introduceren te onderzoeken, en te verifiëren dat de combinatoren gewenste eigenschappen hebben. In de Informatica wordt de algebraïsche aanpak voor het beschrijven van talen, processen en methoden veel gebruikt [35]. Hiermee is algebra ook een belangrijke methode voor het bestuderen van hoe we studenten kunnen volgen.

10 Feedback, types, en bewijzen

In sommige domeinen kunnen we een object automatisch analyseren, en op grond van die analyse uitspraken over partiële correctheid van het object doen. Voorbeelden hiervan zijn type-analyse voor programma's [51, 19] en dimensie-analyse van natuurkundige expressies [63, 40]. Daarnaast worden de methoden voor het automatisch analyseren van volledige correctheid van objecten zoals programma's steeds krachtiger [14, 74]. Naast het automatisch kunnen analyseren of correct bewijzen van objecten, moeten we in het geval dat zo'n

analyse of bewijs mislukt ook kunnen verklaren waarom het is mislukt. Dit probleem is vaak minstens even moeilijk als de automatische analyse of het automatische bewijs zelf [30]. Onder dit thema komen verschillende problemen naar voren. Als eerste kunnen we automatische analyses gebruiken om meer te weten te komen over waar eventuele problemen veroorzaakt worden. Als we eigenschappen waar een volledige oplossing aan moet voldoen automatisch kunnen vertalen naar eigenschappen waar de onderdelen aan moeten voldoen, dan kunnen we meer gedetailleerde feedback geven aan gebruikers. Dit probleem staat bekend als contract-inferentie, en is al in beperkte context bestudeerd [17]. Een ander probleem is het verklaren van falende bewijzen. In zijn algemeenheid is dit zeer moeilijk, maar voor problemen die beginnende studenten bewijzen kunnen we mogelijk veel bereiken.

11 Feedback en modellen

Complexe systemen zoals het klimaat en het financiële systeem hebben een grote invloed op ons leven. Dergelijke systemen worden bestudeerd in de biologie, economie, en veel andere wetenschapsgebieden [7]. Een complex systeem wordt vaak beschreven met behulp van een wiskundig model. Het beïnvloeden van een complex systeem vereist inzicht in het wiskundige model, en in het effect van het veranderen van één of meer parameters, die meestal worden geïmplementeerd als variabelen in het model. In veel gebieden leren studenten beslissingen te nemen binnen een complex systeem [37]. Voor dit doel zijn serious games of simulaties zeer geschikt. Een serious game of simulatie kan een speler de gevolgen van beslissingen laten zien in een gesimuleerde werkelijkheid.

Het is niet makkelijk om simulaties voor complexe systemen in het onderwijs in te zetten, vanwege de vaak slechte fit met de inhoud van een cursus [44]. Idealiter specificeert een docent of een leerling een wiskundig model, en geeft de beschreven simulatie feedback op de acties van de speler op het juiste niveau. Docenten en studenten moeten simulaties kunnen kiezen of aanpassen aan hun behoeften. Als dat op dit moment al mogelijk is, is het aanpassen van een bestaande simulatie vaak moeilijk, en is het ontwikkelen van een nieuwe simulatie vaak een zeer uitdagende taak die veel ontwerp- en ontwikkelcapaciteit vergt. Ik wil werken aan een raamwerk waarin we wiskundige modellen kunnen specificeren, samen met de benodigde feedbackservices die interacties van spelers analyseren om verschillende soorten feedback te geven aan de speler of de docent, om scores te berekenen, etc. Een belangrijk doel van het raamwerk is de benodigde inspanning voor het aanpassen of ontwikkelen van simulaties sterk te beperken.

Voorbeelden van eenvoudige simulaties die ik hiermee hoop te kunnen ontwikkelen zijn simulaties waarin een speler een behandeling voor kanker moet opstellen, of een productieplan voor een innovatief product op moet stellen. De groei van tumoren en de verspreiding van innovaties worden beide beschreven door (varianten van) logistieke functies. In deze games zou een speler abusievelijk kunnen denken dat de groei van tumoren of verspreiding van producten beschreven wordt door een lineaire of kwadratische functie, in plaats van een logistieke. De technologie die we nodig hebben om patronen te bepalen die het bestaan van een dergelijk misverstand signaleren is hetzelfde voor deze games, en heeft error-correctie, geavanceerde pattern-matching, en learning analytics technieken nodig.

12 Wanneer is het goed?

Wanneer ben ik tevreden met de resultaten die we hebben bereikt? Een belangrijk doel van het onderzoek is onderscheid te maken tussen goed en kwaad, maar hoe weet ik of ik dat onderscheid goed maak? Kortom, hoe kan ik mijn eigen onderzoek langs de lat van goed naar kwaad leggen?

Om de kwaliteit van een antwoord op de vraag hoe ik automatisch precieze feedback kan geven aan studenten te beoordelen, moet ik weten wat 'de' goede feedback is in een bepaalde situatie. Ik wil de automatische feedback kunnen vergelijken met de feedback die op een andere manier is verkregen, en waarvan ik weet dat de kwaliteit hoog is. Er zijn meerdere manieren om informatie te krijgen over de kwaliteit van automatisch gegenereerde feedback:

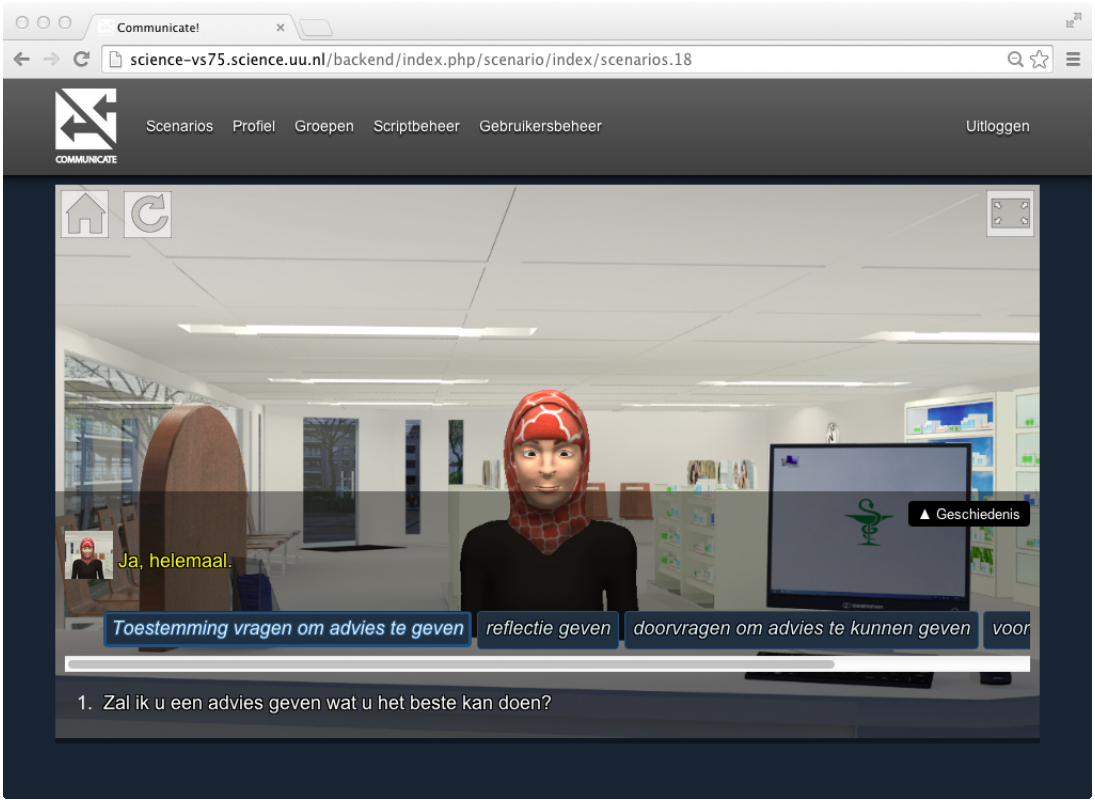
- er bestaat een algoritme dat automatisch de kwaliteit van de feedback bepaalt,
- er is een benchmark gedefinieerd die we kunnen gebruiken om de kwaliteit van feedback te bepalen,
- lesboeken beschrijven precies hoe goede oplossingen worden geconstrueerd,
- we vragen docenten om de kwaliteit van feedback te beoordelen,
- of we vragen studenten de kwaliteit van feedback te beoordelen.

Ieder van deze methoden geeft semantiek aan feedback. Goede automatisch gegenereerde feedback is *sound* en *complete*. Dit zijn bekende begrippen in de logica en informatica [52], die rechtstreeks naar het geven van feedback kunnen worden vertaald. Automatisch gegenereerde feedback is *sound* als de gegenereerde feedback klopt: als een studentproduct als (in)correct wordt gediagnosticeerd, dan is dat product ook daadwerkelijk (in)correct. Automatisch gegenereerde feedback is *complete* als het dezelfde feedback geeft die bijvoorbeeld een docent geeft in een willekeurige situatie. Vooral completeness is vaak lastig te bewerkstelligen.

Het uiteindelijke doel van ons werk is het leren van studenten te ondersteunen, maar onderwijskundige effectiviteitsonderzoeken kijken uiteraard juist naar leereffecten, en het is moeilijk zo niet onmogelijk om uit leereffecten uitspraken te doen over de kwaliteit van de automatisch gegenereerde feedback. Daarnaast kunnen dit soort onderzoeken het best worden uitgevoerd door onderwijskundigen.

Om te bepalen of softwaretechnologie voor leren en onderwijs makkelijker te ontwikkelen, gebruiken, en aan te passen is gebruiken we in eerste instantie de kwaliteitsattributen van de ontwikkelde software [8].

Tenslotte, de kwaliteit van de softwaretechnologische vernieuwingen bepalen we met de daarvoor binnen de Informatica geijkte methoden.

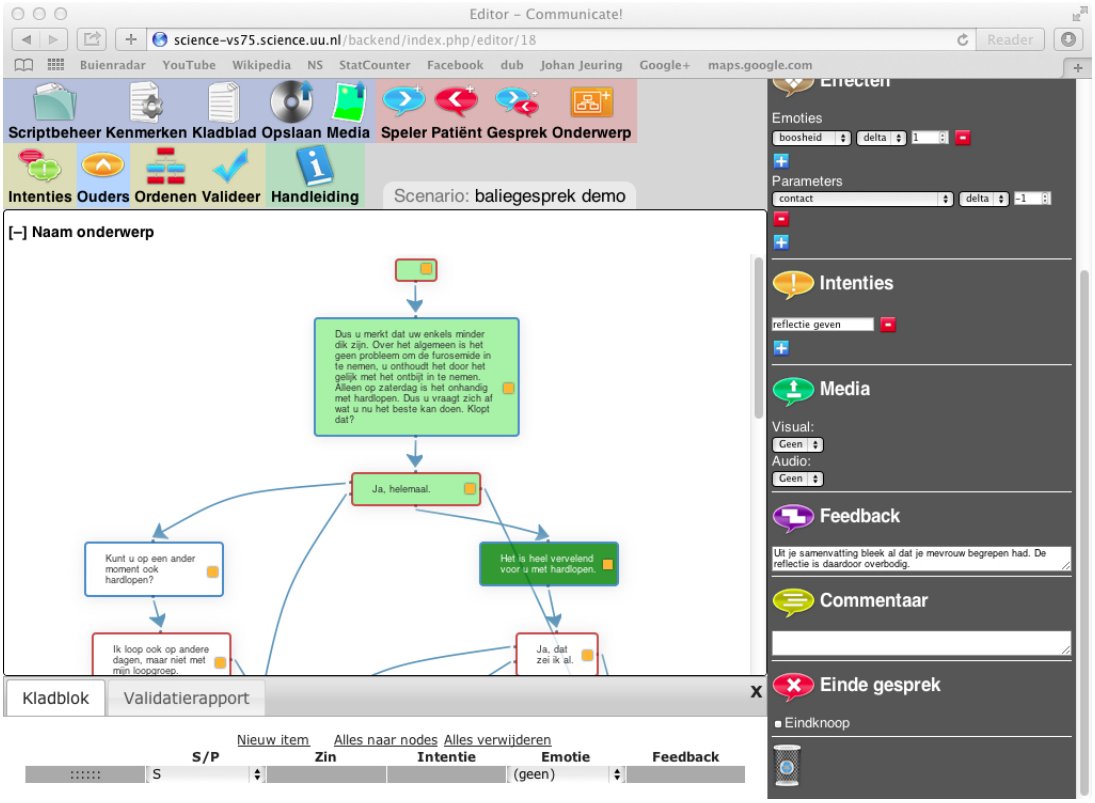


Figuur 10: Communicate! game.

13 Digitalisering van het onderwijs

Al decennia lang wordt er geëxperimenteerd met het gebruik van softwaretechnologie in het onderwijs. Sinds de opkomst van MOOCs (Massive Open Online Courses) is de aandacht voor technologie in het (hoger) onderwijs geëxplodeerd. In het afgelopen jaar is uitgebreid gesproken over de ontwikkeling van visies voor het vergroten van het gebruik van technologie in het onderwijs door de EU [21]: 'Opening up Education: Innovative teaching and learning for all through new Technologies and Open Educational Resources', het ministerie van OCW, dat werkt aan een kennisagenda op het gebied van digitalisering van het onderwijs, de League of European Research Universities (LERU) [48] waar de Universiteit Utrecht deel van uitmaakt: 'Online learning at research-intensive universities', de Universiteit Utrecht zelf: BaMa 3.next: Visie op Onderwijs en IT, en naar alle waarschijnlijkheid bijna alle andere hogeronderwijsinstellingen in Nederland.

Naast de opkomst van MOOCs en de daarop volgende interesse in het gebruik van technologie in het onderwijs is ook de interesse in het gebruik van serious games in het onderwijs sterk toegenomen de afgelopen jaren. Alleen al in het afgelopen jaar heb ik gesprekken gevoerd met docenten uit de faculteiten Geneeskunde, Diergeneeskunde, Sociale Wetenschappen, Rebo (Recht, Economie, Bestuur en Organisatie), Letteren, en uiteraard Bètawetenschappen over de ontwikkeling en het gebruik van serious games of simulaties



Figuur 11: Communicate! scenario editor.

in het onderwijs, bijvoorbeeld voor het leren van klinisch redeneren, voor het oefenen met communicatievaardigheden, en voor het leren hoe ons immuunsysteem werkt. In het vak Gameproject waarmee studenten hun bachelor Informatica met de specialisatie gametechnologie afsluiten worden veel serious games ontwikkeld, voor uitgevers, bedrijven, musea, en verschillende opleidingen van de universiteit zelf. Zo ben ik zelf betrokken bij twee projecten die gefinancierd worden door het onderwijsstimuleringsfonds van de Universiteit Utrecht: Communicate! Een serious game voor communicatievaardigheden, gestart in 2013, en Simulaties en simulation gaming in universitair onderwijs, dat een paar maanden geleden is gestart. Voor Communicate! hebben twee groepen studenten gewerkt aan een game voor het oefenen van communicatievaardigheden, samen met een editor waar docenten scenario's voor de game kunnen ontwikkelen, zie de Figuren 10 en 11 voor screenshots van deze componenten. De kunstmatige intelligentie in Communicate! maakt gebruik van ons raamwerk voor het volgen van interacties van studenten, en voor het geven van de volgende stap.

Ik denk dat de Universiteit Utrecht op de goede weg is met het gebruik van technologie en softwaretechnologie in het onderwijs. We staan pas aan het begin, maar de eerste stappen zijn gemaakt. Een aantal basisfaciliteiten, zoals het opnemen van weblectures, zijn nu goed georganiseerd, en relatief eenvoudig in te zetten in het onderwijs. Voor andere faciliteiten, zoals digitaal toetsen en het geven van digitale feedback op studentessays, worden nu voorbereidingen getroffen. Er wordt op veel plekken geëxperimenteerd met het gebruik van

softwaretechnologie in het onderwijs, en binnen het onderwijsstimuleringsfonds is ruimte om grotere projecten waarbij meerdere faculteiten zijn betrokken op te zetten.

Ook binnen de Faculteit Bètawetenschappen leven er interessante plannen voor het gebruik van technologie in het onderwijs. De nadruk bij de faculteit zal waarschijnlijk liggen op het ondersteunen van het oefenen met opgaven, en het automatisch genereren van feedback.

14 Valkuilen

Bij de digitalisering van het onderwijs ben ik door de jaren twee grote valkuilen tegengekomen:

- 'er bestaat geen tool die precies doet wat ik nodig heb, dus moeten we alles zelf ontwikkelen'. Ik heb bijvoorbeeld heel veel tools gezien die het wiskundeonderwijs ondersteunen; in bijna ieder Europees land zijn meerdere versies van dit soort technologie ontwikkeld. Ook ik heb mij daar schuldig aan gemaakt [59]. Al heel gauw realiseerden we ons dat het ontwikkelen van nog een tool voor het wiskundeonderwijs heel veel werk is en weinig toegevoegde waarde heeft. Onze focus op feedbackservices [24, 32], specifieke componenten die gebruikt worden door verschillende tools voor het wiskunde- en ander onderwijs, is daarentegen zeer nuttig en vruchtbaar gebleken. Veel beschrijvingen van intelligente tutoringsystemen onderscheiden verschillende componenten in deze systemen [67, 56, 71], zoals een editor of user-interface, een domain reasoner, en een studentmodel, maar slechts weinig systeemimplementaties bestaan uit componenten van verschillende leveranciers.
- 'er is al zoveel software ontwikkeld, dat hoeven we niet meer zelf te doen'. Ik heb vaak gezien dat bij hoger management zonder technische achtergrond het idee leeft dat software iets is wat je inkoopt of uitbesteedt. Helaas hebben organisaties die onderzoek of experimenten financieren zoals Surf ook een tijdlang dit idee aangehangen, met als gevolg dat ik in onderzoeksaanvragen heb moeten verbergen dat we ook software gingen ontwikkelen. Voor standaard software voor bijvoorbeeld spreadsheets is inkoop vaak goed mogelijk[4], maar voor meer geavanceerde software zoals software voor leren en onderwijs, inclusief serious games, gaat dit idee voorbij aan het feit dat het ontwikkelen van software verweven is met het genereren van ideeën. De ontwikkeling van onze Communicate! game was onmogelijk geweest als er geen voortdurende uitwisseling tussen het softwareontwikkelteam en de opdrachtgevers in het projectteam had plaatsgevonden. Uitbesteden van softwareontwikkeling in deze gevallen leidt tot verarming van ideeën, en vaak tot slecht bruikbare software.

Ik hoop dat ikzelf, de universiteit, organisaties die onderzoek en vernieuwingen financieren, en het ministerie deze valkuilen in de toekomst zullen ontwijken.

[4] Administratieve processen in hogeronderwijsinstellingen zijn helaas nog te complex om daar standaard software voor te gebruiken. Veel van de software die we daar nu voor gebruiken is ingekochte software die niet in nauwe samenwerking met gebruikers en niet met veel kennis van user-interfaces is ontwikkeld. Het resultaat is vaak software van dramatisch laag niveau vanuit gebruikers- en ontwikkelingsperspectief.

15 Onderwijs

Hoe komt softwaretechnologie voor leren en onderwijs terug in het onderwijs aan de universiteit? In de bacheloropleiding Informatica komt softwaretechnologie uiteraard uitgebreid aan bod, en de basistechnologieën nodig voor mijn onderzoek komen uitgebreid aan de orde. De bachelor Informatiekunde besteedt expliciet aandacht aan technologie voor leren, onder andere in het vak Intelligente Interactie.

Aan het einde van de bachelor Informatica voeren studenten in groepen van ongeveer tien studenten een groot project uit. Veel van deze groepen ontwikkelen software die het leren ondersteunt, en met name de studenten die de gametechnologie variant van de opleiding Informatica volgen ontwikkelen vaak serious games. De afgelopen jaren heb ik regelmatig als opdrachtgever voor zulke projecten gefunctioneerd, en ik verwacht dat ik dat in de toekomst vaak weer zal doen.

De afgelopen jaren heb ik meer dan tien afstudeerders, zowel op bachelor als masterniveau, begeleid met het uitvoeren van afstudeeronderzoek op het gebied van technologie voor leren en onderwijs. Daar zijn veel mooie resultaten uit naar voren gekomen, en ik hoop ook in de toekomst veel resultaten samen met afstudeerders te ontwikkelen.

Softwaretechnologie voor leren en onderwijs is een mooi onderzoeksgebied waarin veel aspecten van de informatica samen komen: softwaretechnologie, taaltechnologie, kunstmatige intelligentie, data-analyse, interactietechnologie, serious games, en meer. Naast technologische aspecten zijn ook onderwijskundige onderwerpen van belang: hoe leren studenten, hoe vertaal ik eindtermen in taken, welke conclusies kan ik trekken uit interacties van studenten, etc. Ik vind het interessant om te onderzoeken of we dit thema geïntegreerd in ons onderwijs aan kunnen bieden, als een minor in de bachelor samen met onderwijskunde en/of kunstmatige intelligentie, of als een track in een masterprogramma.

16 Valorisatie

Softwaretechnologie voor leren en onderwijs is niet alleen interessant als onderzoeksonderwerp, maar heeft ook grote potentie om gebruikt te worden door allerlei mensen en op allerlei niveaus. Ik hoop dat mijn onderzoek niet alleen kennis vermeerdert, maar dat het ook daadwerkelijk bijdraagt aan het onderwijs dat leerlingen en studenten nu en in de toekomst ontvangen. In de afgelopen jaren zijn de implementaties van de ideeën die we hebben ontwikkeld gebruikt in de DWO, in Math-Bridge, in Ask-Elle, en in een tutor voor logica. In totaal heeft onze feedbackserver honderdduizenden hits ontvangen van studenten die bezig zijn met het werken aan wiskunde-, programmeer-, en logicaopgaven.

De ontwikkeling van onze software is gefinancierd door de EU, Surf, de Open Universiteit, maar ook door een uitgever. Nog geen half jaar geleden heb ik met diezelfde uitgever gesproken over hoe feedbackservices in de software die de uitgever samen met een aantal wiskundeboeken uitlevert verder vormgegeven kunnen worden.

Ook mijn bijdrage aan projecten die gefinancierd worden door het onderwijsstimuleringsfonds van de Universiteit Utrecht zie ik als een vorm van valorisatie. Door de samenwer-

king met Volksgezondheid Utrecht ziet het er bijvoorbeeld naar uit dat Communicate! een belangrijke rol gaat spelen in de communicatievaardighedentrainingen binnen de gezondheidszorg, die vanuit de Nederlandse gemeenten worden georganiseerd.

17 Tot besluit

De komende jaren zal het gebruik van softwaretechnologie in het onderwijs zich stormachtig ontwikkelen. Naast het oplossen van allerlei onderwijskundige uitdagingen is het noodzakelijk dat de technologie die daarbij gebruikt wordt het leren en onderwijs daadwerkelijk ondersteunt, en niet belemmert. Daarvoor moet de technologie goede, voor zover mogelijk automatisch gegenereerde, feedback geven, goede taken selecteren, en aanpasbaar zijn door zowel de docent als de student. Hoewel softwaretechnologie voor leren en onderwijs al jaren de belofte heeft leren en onderwijs op deze manier te ondersteunen, is het aantal toepassingen tot nu toe beperkt gebleven. Door gebruik te maken van fundamentele concepten zoals talen, grammatica's en algebra's kunnen we ervoor zorgen dat softwaretechnologie beter studenten volgt, en makkelijker aanpasbaar is. De op deze concepten ontwikkelde technologie kunnen we toepassen in interactieve leeromgevingen, digitale toetssystemen, simulaties, en serious games. Over twintig jaar zal het landschap van technologie voor leren en onderwijs ingrijpend zijn veranderd, en ik hoop daar samen met collega's en studenten een bijdrage aan te kunnen leveren.

18 Dankwoord

Als eerste wil ik het departement Informatica, de Faculteit Bètawetenschappen, en de Universiteit Utrecht bedanken voor het in mij gestelde vertrouwen. De afgelopen jaren heb ik veel met collega's van de faculteit Bètawetenschappen, van andere faculteiten, en van de directie onderwijs en onderzoek van de Universiteit Utrecht samengewerkt, en heb dat als verrijkend ervaren. Ook binnen het departement Informatica heerst er een constructieve sfeer, en de samenwerking binnen de softwaretechnologie groep vind ik prettig, constructief, en collegiaal.

Samenwerken is belangrijk voor het formuleren, begrijpen, en oplossen van problemen. Soms leidt een enkele opmerking van een collega tot nieuwe inzichten, soms besteed ik uren voor een whiteboard met één of meerdere collega's en studenten om samen een probleem op te lossen. Het samen met collega's precies beschrijven van onderzoeksproblemen, en het vinden van oplossingen voor die problemen, vind ik één van de meest bevredigende activiteiten die er bestaan.

Door de jaren heen heb ik met heel veel mensen samengewerkt. In mijn eerste stappen als onderzoeker ben ik begeleid door Jan Terlouw, Lambert Meertens, Maarten Fokkinga en Jaap van der Woude. De STOP-groep waar ik toen deel van uitmaakte heeft jarenlang een zeer vruchtbare context gevormd, op het CWI, in Utrecht, en op Ameland, waarbij Doaitse Swierstra en Roland Backhouse een grote rol speelden. In de jaren daarna heb ik met veel plezier samengewerkt met de AIO's die ik heb begeleid: Patrik Jansson, Andres Löh, Mar-

tijn Schrage, Frank Atanassow, Alexey Rodriguez, Pedro Magalhaes, Alex Gerdes, en Sean Leather.

De faculteit Informatica van de Open Universiteit heeft mij de mogelijkheid en de context gegeven om me te ontwikkelen op het gebied van softwaretechnologie voor leren en onderwijs. Harrie Passier en Evert van de Vrie waren betrokken bij de eerste stappen, en Josje Lodder draagt al heel lang bij aan de inhoud. De grootste vooruitgang heb ik vooral samen met Bastiaan Heeren gemaakt. Onderhand hebben we meer dan twintig artikelen samen geschreven over softwaretechnologie voor leren en onderwijs, leveren we feedback voor verschillende leeromgevingen, en wisselen we nog steeds bijna dagelijks ideeën uit. Ik verwacht dat we met dat wat we de afgelopen jaren hebben opgebouwd, in de komende jaren nog veel plezier kunnen beleven, en veel kunnen bereiken.

Deze rede is door Eddy Boeve, Mariska Jeuring, en Bastiaan Heeren van commentaar voorzien.

Ik ben blij dat mijn moeder hier kan zijn, en vind het jammer dat mijn vader dit niet meer mee kan maken. Zij hebben me gestimuleerd in mijn ontwikkeling, en mij de gelegenheid gegeven mijn eigen weg te vinden.

Tenslotte, Cilia, Jonathan en Elisa vormen het liefste en fijnste gezin dat ik mij zou kunnen wensen, en ik hoop ook in de toekomst nog heel veel samen te delen. Vanaf nu hoef ik in ieder geval geen zondagen meer aan mijn oratie te besteden, want

Ik heb gezegd.

Referenties

[1] Clark C. Abt. *Serious Games*. New York: The Viking Press, 1970.

[2] Clark Aldrich. *The Complete Guide to Simulations and Serious Games: How the Most Valuable Content Will be Created in the Age Beyond Gutenberg to Google*. Pfeiffer, San Francisco, 2009.

[3] John R. Anderson, Albert T. Corbett, Kenneth R. Koedinger, and Ray Pelletier. Cognitive tutors: lessons learned. *Journal of the Learning Sciences*, 4(2):167–207, 1995.

[4] John R. Anderson and Ray Pelletier. A development system for model-tracing tutors. In L. Birnbaum, editor, *Proceedings of the international conference of the learning sciences*, pages 1–8, 1991.

[5] Ken Bain. *What the Best College Teachers Do*. Cambridge, MA: Harvard University Press, 2004.

[6] Marjoke Bakker. *Using mini-games for learning multiplication and division: A longitudinal effect study*. PhD thesis, Utrecht University, 2014.

[7] Yaneer Bar-Yam. *Dynamics of complex systems*, volume 213. Addison-Wesley Reading, MA, 1997.

[8] Mario Barbacci, Mark H. Klein, Thomas A. Longstaff, and Charles B. Weinstock. Qua-

lity attributes. Technical Report CMU/SEI-95-TR-021, Software Engineering Institute, Carnegie-Mellon University, 1995.

[9] John Biggs and Catherine Tang. *Teaching for Quality Learning at University*. McGraw-Hill International, 2007.

[10] Christian Bokhove and Paul Drijvers. Digital Tools for Algebra Education: Criteria and Evaluation. *Int. Journal of Comp. for Math. Learning*, 15(1):45–62, 2010.

[11] David Boud and Elizabeth Molloy, editors. *Feedback in higher and professional education: understanding it and doing it well*. Routledge, 2012.

[12] Lingguo Bu and Robert Schoen. *Model-centered Learning: Pathways to Mathematical Understanding Using GeoGebra*, volume 6. Springer, 2012.

[13] Alan C.K. Cheung and Robert E. Slavin. The effectiveness of educational technology applications for enhancing mathematics achievement in k-12 classrooms: A meta-analysis. *Educational Research Review*, 9:88–113, 2013.

[14] Koen Claessen, Moa Johansson, Dan Rosén, and Nicholas Smallbone. Automating inductive proofs using theory exploration. In Maria Paola Bonacina, editor, *Automated Deduction CADE-24*, volume 7898 of *LNCS*, pages 392–406. Springer, 2013.

[15] Michael C. Corballis, editor. *The Recursive Mind – The Origins of Human Language, Thought, and Civilization*. Princeton University Press, 2011.

[16] Albert T. Corbett, Kenneth R. Koedinger, and John R. Anderson. Intelligent tutoring systems. In M. Helander, T. K. Landauer, and P. Prahu, editors, *Handbook of Human-Computer Interaction, Second Edition*, pages 849–874. Elsevier Science, 1997.

[17] Patrick M. Cousot, Radhia Cousot, Francesco Logozzo, and Michael Barnett. An abstract interpretation framework for refactoring with application to extract methods with contracts. In *Proceedings of OOPSLA '12: the ACM International Conference on Object Oriented Programming Systems Languages and Applications*, pages 213–232. ACM, 2012.

[18] William Croft. Evolution: Language use and the evolution of languages. In Philippe Binder and Kenny Smith, editors, *The language phenomenon – Human communication from milliseconds to millennia*, The Frontiers Collection, pages 93–120. Springer, 2013.

[19] Luis Damas and Robin Milner. Principal type-schemes for functional programs. In *Proceedings of POPL'82: the 9th ACM SIGPLAN-SIGACT symposium on Principles of Programming Languages*, pages 207–212. ACM, 1982.

[20] Michel Doorman, Paul Drijvers, Peter Boon, Sjef van Gisbergen, and Koeno Gravemeijer. Design and implementation of a computer supported learning environment for mathematics. In *Earli 2009 SIG20 invited Symposium Issues in designing and implementing computer supported inquiry learning environments*, 2009.

[21] European Commission. Opening up education: Innovative teaching and learning for all through new technologies and open educational resources. Communication from the commission to the European parliament, the council, the European economic and social committee and the committee of the regions, 2013.

[22] Frank Fischer, Fridolin Wild, Rosamund Sutherland, and Lena Zirn, editors. *Grand Challenges in Technology Enhanced Learning – Outcomes of the 3rd Alpine Rendez-Vous*. SpringerBriefs in Education. Springer International Publishing, 2014.

[23] D. Randy Garrison and Heather Kanuka. Blended learning: Uncovering its transformative potential in higher education. *The Internet and Higher Education*, 7(2):95–105, 2004.

[24] Alex Gerdes, Bastiaan Heeren, Johan Jeuring, and Sylvia Stuurman. Feedback services for exercise assistants. In D. Remenyi, editor, *ECEL*, pages 402–410. Acad. Publ. Ltd., 2008.

[25] Alex Gerdes, Johan Jeuring, and Bastiaan Heeren. An interactive functional programming tutor. In T. Lapidot, J. Gal-Ezer, M.E. Caspersen, and O. Hazzan, editors, *Proceedings of ITICSE 2012: the 17th Annual Conference on Innovation and Technology in Computer Science Education*, pages 250–255. ACM, 2012.

[26] George Goguadze. *ActiveMath - Generation and Reuse of Interactive Exercises using Domain Reasoners and Automated Tutorial Strategies*. PhD thesis, Universität des Saarlandes, Germany, May 2011.

[27] Jo Handelsman, Sarah Miller, and Christine Pfund. *Scientific Teaching*. W.H. Freeman and Company, 2007.

[28] John Hattie. *Visible Learning: A Synthesis of Over 800 Meta-Analyses Relating to Achievement*. Routledge, 2008.

[29] John Hattie and Helen Timperley. The power of feedback. *Review of Educational Research*, 77(1):81–112, 2007.

[30] Bastiaan Heeren. *Top Quality Type Error Messages*. PhD thesis, Utrecht University, 2005.

[31] Bastiaan Heeren and Johan Jeuring. Interleaving strategies. In *Proceedings of MKM 2011: the 10th International Conference on Mathematical Knowledge Management*, volume 6824 of *LNAI*, pages 196–211. Springer, 2011.

[32] Bastiaan Heeren and Johan Jeuring. Feedback services for stepwise exercises. *Science of Computer Programming*, 88:110–129, 2014. Software Development Concerns in the e-Learning Domain.

[33] Bastiaan Heeren, Johan Jeuring, and Alex Gerdes. Specifying rewrite strategies for interactive exercises. *Mathematics in Computer Science*, 3(3):349–370, 2010.

[34] Bastiaan Heeren, Johan Jeuring, Arthur van Leeuwen, and Alex Gerdes. Specifying strategies for exercises. In *Proceedings of MKM 2008: the 7th International Conference on Mathematical Knowledge Management*, volume 5144 of *LNAI*, pages 430–445. Springer, 2008.

[35] Tony Hoare. *Communicating Sequential Processes*. Prentice Hall International, 1985.

[36] John Hughes. Why functional programming matters. *The Computer Journal*, 32(2):98–107, 1989.

[37] Michael J. Jacobson and Uri Wilensky. Complex systems in education: Scientific and educational importance and implications for the learning sciences. *The journal of the*

learning sciences, 15(1):11–34, 2006.

[38] Johan Jeuring, Alex Gerdes, and Bastiaan Heeren. A programming tutor for Haskell. In Viktória Zsók, Zoltan Horváth, and Rinus Plasmeijer, editors, *Proceedings of CEFP 2011: Lecture Notes of the Central European School on Functional Programming*, volume 7241 of *LNCS*, pages 1–45. Springer, 2011.

[39] Ton de Jong. Computer simulations – technological advances in inquiry learning. *Science*, 312:532–533, 2006.

[40] Andrew Kennedy. *Programming Languages and Dimensions*. PhD thesis, University of Cambridge, 1996.

[41] Simon Kirby. The evolution of linguistic replicators. In Philippe Binder and Kenny Smith, editors, *The language phenomenon – Human communication from milliseconds to millennia*, The Frontiers Collection, pages 121–138. Springer, 2013.

[42] Diana Laurillard. *Rethinking university teaching – a framework for the effective use of learning technologies*. RoutledgeFalmer, 2002.

[43] Nguyen-Thinh Le, Frank Loll, and Niels Pinkwart. Operationalizing the continuum between well-defined and ill-defined problems for educational technology. *IEEE Journal Transactions on Learning Technologies*, 6(3):258–270, 2013.

[44] Jonathan Lean, Jonathan Moizer, Michael Towler, and Caroline Abbey. Simulations and games use and barriers in higher education. *Active learning in higher education*, 7(3):227–242, 2006.

[45] Josje Lodder, Bastiaan Heeren, and Johan Jeuring. A domain reasoner for propositional logic. Technical report, in preparation, 2014.

[46] Josje Lodder, Johan Jeuring, and Harrie Passier. An interactive tool for manipulating logical formulae. In M. Manzano, B. Pérez Lancho, and A. Gil, editors, *Proceedings of the Second International Congress on Tools for Teaching Logic*, 2006.

[47] Susan Lowes. Online teaching and classroom change: The impact of virtual high school on its teachers and their schools. Technical report, Columbia University, Institute for Learning Technologies, 2007.

[48] Sally Mapstone (Ed), Simone Buitendijk, and Eva Wiberg. Online learning at research-intensive universities. LERU advice paper No.16, June 2014.

[49] John Maynard Smith and Eörs Szathmáry. *The Major Transitions in Evolution*. Oxford University Press, 1995.

[50] Jeroen J.G. van Merriënboer and Paul Kirschner. *Ten Steps to Complex Learning: A Systematic Approach to Four-Component Instructional Design*. New Jersey: Lawrence Erlbaum, 2007.

[51] Robin Milner. A theory of type polymorphism in programming. *Journal of Computer and System Sciences*, 17(3):348–375, 1978.

[52] John C. Mitchell. *Foundations of Programming Languages*. MIT Press, Cambridge, MA, USA, 1996.

[53] Susanne Narciss. Feedback strategies for interactive learning tasks. In J.M. Spector, M.D. Merrill, J.J.G. van Merriënboer, and M.P. Driscoll, editors, *Handbook of Research on Educational Communications and Technology*. Mahaw, NJ: Lawrence Erlbaum Associates, 2008.

[54] National Research Council. *How People Learn – Brain, Mind, Experience, and School*. National Academy Press, 2000.

[55] Alan Newell and Herbert A. Simon. The theory of human problem solving. In Allan Collins and Edward E. Smith, editors, *Readings in Cognitive Science – A Perspective from Psychology and Artificial Intelligence*, pages 33–51. Morgan Kaufmann Publishers, Inc., 1988 (originally 1972).

[56] Hyacinth S. Nwana. Intelligent tutoring systems: an overview. *Artificial Intelligence Review*, 4(4):251–277, 1990.

[57] Tim Olmer, Bastiaan Heeren, and Johan Jeuring. Evaluating Haskell expressions in a tutoring environment. To appear in Electronic Proceedings in Theoretical Computer Science, special issue on Trends in Functional Programming in Education, 2014.

[58] Claus Pahl. Managing evolution and change in web-based teaching and learning environments. *Computers & Education*, 40(2):99–114, 2003.

[59] Harrie Passier and Johan Jeuring. Feedback in an interactive equation solver. In M. Seppälä, S. Xambo, and O. Caprotti, editors, *Proceedings of the Web Advanced Learning Conference and Exhibition, WebALT 2006*, pages 53–68. Oy WebALT Inc., 2006.

[60] Michael Prosser and Keith Trigwell. *Understanding learning and Teaching - The Experience in Higher Education*. The Society for Research into Higher Education and Open University Press, 1999.

[61] Phil Race. *Making learning happen – A guide for post-compulsory education*. Sage, 2005.

[62] Thomas C. Reeves. Alternative assessment approaches for online learning environments in higher education. *Journal of Educational Computing Research*, 23(1):101–111, 2000.

[63] Mikael Rittri. Dimension inference under polymorphic recursion. In *Proceedings of FPCA 1995: the SIGPLAN- SIGARCH-WG2.8 Conference on Functional Programming Languages and Computer Architecture*, pages 147–159. ACM, 1995.

[64] Chris Sangwin. *Computer Aided Assessment of Mathematics*. Oxford University Press, 2013.

[65] John Self. Model-based cognitive diagnosis. *User Modeling and User-Adapted Interaction*, 3(1):89–106, 1993.

[66] Valerie J. Shute. Focus on formative feedback. *Review of Educational Research*, 78(1):153–189, 2008.

[67] Valerie J. Shute and Joseph Psotka. Intelligent tutoring systems: Past, present and future. In D. Jonassen, editor, *Handbook of Research on Educational Communications and Technology*. Scholastic Publications, 1996.

[68] Sergey Sosnovsky, Michael Dietrich, Eric Andrès, George Goguadze, Stefan Winterstein, Paul Libbrecht, Jörg Siekmann, and Erica Melis. Math-bridge: Closing gaps in european remedial mathematics with technology-enhanced learning. In Thomas Wassong, Daniel Frischemeier, Pascal R. Fischer, Reinhard Hochmuth, and Peter Bender, editors, *Mit Werkzeugen Mathematik und Stochastik lernen Using Tools for Learning Mathematics and Statistics*, pages 437–451. Springer, 2014.

[69] Doaitse S. Swierstra and Atze Dijkstra. Parse your options. In Peter Achten and Pieter Koopman, editors, *The Beauty of Functional Code*, volume 8106 of *LNCS*, pages 234–249. Springer, 2013.

[70] TNO/CBS. Nationale enquête arbeidsvoorwaarden. 2012.

[71] Kurt VanLehn. The behavior of tutoring systems. *International Journal of Artificial Intelligence in Education*, 16(3):227–265, 2006.

[72] Kurt VanLehn. The relative effectiveness of human tutoring, intelligent tutoring systems, and other tutoring systems. *Educational Psychologist*, 46(4):197–221, 2011.

[73] Jennifer J. Vogel, David S. Vogel, Jan Cannon-Bowers, Clint A. Bowers, Kathryn Muse, and Michelle Wright. Computer gaming and interactive simulations for learning: A meta-analysis. *Journal of Educational Computing Research*, 34(3):229–243, 2006.

[74] Dana N. Xu, Simon L. Peyton Jones, and Koen Claessen. Static contract checking for Haskell. In Zhong Shao and Benjamin C. Pierce, editors, *Proceedings of POPL 2009: the 36th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, pages 41–52. ACM, 2009.