Afstudeeronderwerpen MSc Software Engineering en MSc Computer Science

dr. Bastiaan Heeren

Keywords: tutorsystemen, programmeeronderwijs, softwarekwaliteit, problem-solving procedures

In mijn onderzoek bestudeer ik toepassingen van software technologische concepten om het onderwijs te verbeteren. In het IDEAS project (interactive domain-specific exercise assistants) worden allerlei soorten oefenomgevingen ontworpen en gebouwd, variërend van omgevingen voor het oefenen van logica (LogEx en LogAx) en wiskunde, programmeertutoren (Ask-Elle), tot het trainen van communicatievaardigheden in een serious game. Centraal staat een strategietaal waaruit feedback kan worden gegenereerd, (term)herschrijfsystemen en feedback web-services. Verder ben ik geïnteresseerd in functionele programmeertalen (Haskell) in het onderwijs en voor het modelleren van problemen.

Context van het onderzoek

Het onderzoek naar leer- en oefenomgevingen voor het onderwijs is gestart rond 2007 en wordt vanuit de Open Universiteit (OU) uitgevoerd in samenwerking met de Universiteit Utrecht (UU). Op dit moment bestaat de onderzoeksgroep uit meerdere onderzoekers, promovendi, wetenschappelijk programmeurs en studenten die verschillende aspecten van dit soort omgevingen bestuderen. Veel van de ontwikkelde software maakt gebruik van het ideas package: dit is een generieke library, geschreven in Haskell, voor het ontwikkelen van een 'domain reasoner' met domein-specifieke expertkennis op basis waarvan feedback en hints kunnen worden gegenereerd.

Er wordt samengewerkt in nationale en internationale onderzoeksprojecten, waaronder het Advise-Me project waarin we het automatisch beoordelen van tussenstappen in het voortgezet reken-wiskundeonderwijs bestuderen.

De onderzoeksgroep organiseert regelmatig bijeenkomsten in Utrecht met presentaties van (externe) onderzoekers. Bij OU-masterscripties treden Johan Jeuring (UU en OU) en Josje Lodder (OU) vaak op als tweede lezer. Op pagina 3 staan voorbeelden van afstudeeropdrachten over e-learning systemen voor logica die onder begeleiding van Josje Lodder uitgevoerd kunnen worden.

Voorkennis

Omdat veel opdrachten voortbouwen op het ideas raamwerk is enige bekendheid met functionele talen (Haskell) gewenst. Een afstudeeropdracht kan zich richten op één of meer van de volgende aspecten: (1) architectuur en ontwerp (bijv. scriptie Gideon Teeuwen); (2) formele talen (bijv. strategietaal en achterliggende theorie); (3) onderwijskundige principes (bijv. scriptie Johan Eikelboom); (4) onderhoudbaarheid en correctheid van oefenomgevingen; (5) empirische validatie (het uitproberen van systemen in de onderwijspraktijk).

Hoe kom je tot een opdracht?

De nadere invulling van een opdracht gebeurt altijd in overleg met de student, zodat rekening kan worden gehouden met voorkeuren en de omvang van het afstudeertraject. Neem gerust contact met mij op om tot een invulling van je afstudeeropdracht te komen (bastiaan.heeren@ou.nl).

Tip: bekijk de lijst met voorbeelden van geschikte afstudeeropdrachten en de aangeraden literatuur. Ga naar mijn homepage voor een actueel overzicht van publicaties en kijk ook eens naar de masterscripties die door mij zijn begeleid (met suggesties voor vervolgonderzoek).

Voorbeelden van afstudeeropdrachten

a) Vaak worden de regels/stappen en de oplosprocedure voor een probleemdomein (wiskunde, logica, etc.) met de hand geschreven. Een complementaire aanpak kan zijn om deze regels en procedures automatisch af te leiden uit voorbeelden en tegenvoorbeelden. Wat is de meerwaarde van deze aanpak, en wanneer werkt dit goed (of juist niet)?

b) De strategietaal is in feite een eenvoudige programmeertaal met ondersteuning voor recursie, abstractie, sequentie, en niet-deterministische keuze. Design by contract (DBC) houdt zich bezig met het specificeren van software componenten m.b.v. pre- en postcondities, invarianten en contracten. Kunnen deze technieken ook worden gebruikt voor het specificeren van procedures die in de strategietaal worden uitgedrukt, inclusief statische of dynamische controle van de specificatie?

c) De strategietaal is een bijzonder soort programmataal, en dus zijn allerlei statische en dynamische technieken en programma-analyses ook toepasbaar op strategieën. Is de cyclomatische complexiteit een betekenisvolle metriek voor de strategietaal? Hoe kunnen veel voorkomende patronen in strategieën worden ontdekt? Hoe kunnen ontwikkelaars worden geholpen om strategieën te refactoren? Wat is de 'coverage' van een strategie gegeven een verzameling uitwerkingen? Welke tooling is nodig om een strategie effectief te kunnen debuggen en in welke mate helpen traces hierbij?

d) Het kiezen van een geschikte stapgrootte (granulariteit) bij het aanbieden van hints en voorbeelduitwerkingen is zeer belangrijk: bij te grote stappen is het onduidelijk wat er gebeurt, bij te kleine stappen kan het overzicht verloren raken. Hoe kunnen meerdere niveaus van granulariteit worden ondersteund, hoe wordt hieruit een geschikte stapgrootte bepaald, en wat zijn de consequenties voor de strategietaal?

e) Puzzels met robotjes in doolhoven zijn een populair middel om programmeren en 'computational thinking' te bevorderen (bijv. in Hour of Code initiatieven). Heeft zo'n benadering, inclusief de aanwezigheid van 'gamification elementen', ook meerwaarde voor het bekend raken met de strategie-combinatoren van het ideas raamwerk? Wat zijn de beperkingen?

f) Tim Olmer beschrijft in zijn masterscriptie de Haskell Expression Evaluator (HEE) voor het stapsgewijs evalueren (uitrekenen) van Haskell-expressies. Het evaluatieproces kan ook inzichtelijk worden gemaakt door functie-tabellen met input-output waarden te presenteren; deze tabellen zouden bovendien kunnen helpen bij het opsporen van fouten. Zijn dergelijke functie-tabellen een nuttige aanvulling op HEE en Ask-Elle in het programmeeronderwijs?

g) Een tutorial (of pedagogische) strategie beschrijft wat voor soort hints en feedback er op een bepaald moment aanwezig is en hoe deze worden gepresenteerd. Hoe beschrijf je zo'n strategie, wat voor functionaliteit wordt er van verwacht en wat zijn de overeenkomsten en verschillen met de strategietaal van ideas (voor het oplossen van opgaven)?

h) Voor de logicatutoren zijn meerder front-end web-applicaties ontwikkeld (in JavaScript) die verschillende soorten logica-opgaven aanbieden (zoals herschrijven naar DNF, logische equivalentie bewijzen, of het geven van een axiomatisch of inductief bewijs). Hoewel veel functionaliteit hetzelfde is, is het hergebruik van code minimaal, met als gevolg dat de onderhoudbaarheid en uitbreidbaarheid problematisch is. Is dit te verhelpen door te kiezen voor een generieke front-end, gebaseerd op de Elm-architectuur? Kan de meerwaarde worden aangetoond in een case study naar de bestaande logica tools?

i) Software metrieken en meta-modellen zijn populair, maar zijn nog niet doorgedrongen tot de community van functionele programmeurs. Harm van den Hoven en Henrie Vos hebben in hun masterscripties een begin gemaakt met metrieken en een meta-model voor Haskell, gebaseerd op het haskell-src-exts package. Hoe kan zo'n meta-model compositioneel en relationeel worden ontworpen, en hoe belangrijk zijn die eigenschappen? Hoe kunnen uit een model (interactieve) visualisaties worden berekend? Hoe kan de evolutie van een systeem worden gerepresenteerd en bevraagd? Hoe kan de schaalbaarheid worden vergroot zodat bijvoorbeeld alle packages op Hackage (of Stack) kunnen worden doorgerekend?

Afstudeeropdrachten bij Josje Lodder

j) Veel e-learning systemen proberen studenten opgaven te laten maken op een manier die vergelijkbaar is met een pen en papier uitwerking. Tegelijkertijd willen deze systemen fouten van studenten kunnen herkennen. Bij opgaven zoals herschrijven van logische formules naar een normaalvorm, zullen studenten op papier vaak verschillende regels in één stap toepassen. In een e-learning systeem dat dit toestaat is het in principe niet mogelijk om te achterhalen wat een student precies heeft gedaan en welke fouten er mogelijk zijn gemaakt. Toch kan een docent vaak wel zien wat er mis is gegaan. Hoe kan een e-learning systeem in een beperkt aantal gevallen toch fouten van studenten herkennen?

k) E-learning systemen kunnen variëren in de manier waarop opgaven worden aangeboden, van gesloten multiple choice vragen tot geheel open uitwerkingen. Een tussenvorm bestaat uit opgaven waarbij de student een bewijs of berekening samen moet stellen uit een verzameling van gegeven beweringen. Deze verzameling kan behalve correcte delen van het bewijs ook afleiders bevatten. Hoe kun je, gegeven een (logica)opgave een geschikte verzameling beweringen genereren?

l) Vertalingen van zinnen in natuurlijke taal naar predikaatlogische formules zijn in het algemeen lastig te geven, maar een deelklasse van dit soort zinnen lijkt generiek te vertalen te zijn. Hoe kan een tutor uitwerkingen van dit soort opgaven controleren, en van feedback voorzien?

Gesuggereerde literatuur

Intelligent Tutoring Systems (algemeen):

- VanLehn, K.: The behavior of tutoring systems. International Journal of Artificial Intelligence in Education 16(3), 227–265 (2006)
- Anderson, J.R., Corbett, A.T., Koedinger, K.R., Pelletier, R.: Cognitive tutors: lessons learned. Journal of the Learning Sciences 4(2), 167–207 (1995)
- Murray, T.: An overview of intelligent tutoring system authoring tools: updated analysis of the state of the art. In: Murray, T., Blessing, S.B., Ainsworth, S. (eds.) Authoring Tools for Advanced Technology Learning Environments, pp. 491–544 (2003)
- Koedinger, K.R., Brunskill, E., Baker, R.S.J.D., McLaughlin, E.A., Stamper, J.: New potentials for data-driven intelligent tutoring system development and optimization. AI Magazine 34(3), 27–41 (2013)

Geselecteerde publicaties:[1]

- Bastiaan Heeren and Johan Jeuring. An Extensible Domain-Specific Language for Describing Problem-Solving Procedures. In Proceedings of Artificial Intelligence in Education: 18th International Conference, AIED 2017, pages 77-89, 2017. Springer International Publishing.
- Josje Lodder, Bastiaan Heeren, and Johan Jeuring. Generating Hints and Feedback for Hilbert-style Axiomatic Proofs. In Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education, SIGCSE '17, pages 387-392, 2017. ACM.
- Alex Gerdes, Bastiaan Heeren, Johan Jeuring, and Thomas van Binsbergen. Ask-Elle: an adaptable programming tutor for Haskell giving automated feed- back. In International Journal of Artificial Intelligence in Education, volume 27 (1), pages 65–100, 2017.
- Hieke Keuning, Johan Jeuring, and Bastiaan Heeren. Towards a Systematic Review of Automated Feedback Generation for Programming Exercises. In Proceedings of the 2016 ACM Conference on Innovation and Technology in Computer Science Education, ITiCSE '16, pages 41-46, 2016. ACM.
- 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.
- Bastiaan Heeren, Johan Jeuring, and Alex Gerdes. Specifying Rewrite Strategies for Interactive Exercises. Mathematics in Computer Science, 3(3):349–370, 2010.

[1] Voor een volledig overzicht, zie http://www.open.ou.nl/bhr/.