Universiteit Utrecht

Bachelor Kunstmatige intelligentie

Scriptie van 7,5 ECTS

Gepersonaliseerd leren programmeren met adaptieve intelligente tutor systemen

Auteur:
Lianne Roest
6200508

Begeleider:
Rianne van Lambalgen

Tweede beoordelaar:
Johan Jeuring

9 juli 2020

Samenvatting

Intelligente tutor systemen streven ernaar om net als een menselijke tutoren studenten te ondersteunen bij het studeren. Daarbij is het wenselijk dat het systeem in staat is om zich aan te kunnen passen op de behoeftes en niveau van de student om zo het leren te personaliseren. Dit werk heeft als doel om te onderzoeken hoe gepersonaliseerd leren gerealiseerd kan worden binnen intelligente tutor systemen. Verschillende technieken voor adaptieve studentbegeleiding en het automatisch genereren van feedback worden met elkaar vergeleken, met als resultaat een overzicht van de toepassingen van adaptiviteit in intelligente tutor systemen.

Inhoudsopgave

Hoofdstuk 1

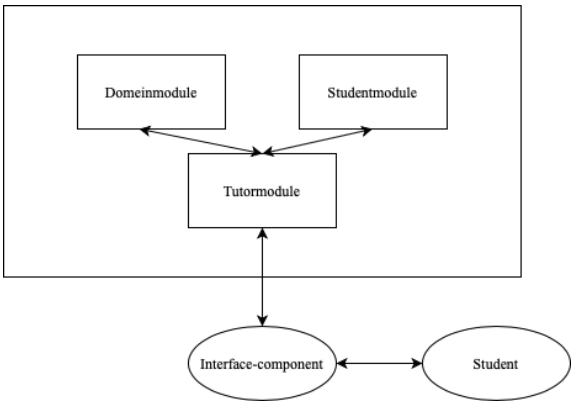
Inleiding

1.1 Intelligente tutor systemen

Van een-op-een studiebegeleiding is bekend dat het een effectieve onderwijsmethode is [1]. Bij deze methode van lesgeven is er meer aandacht voor het lerende individu en wordt het leerproces aangepast op de persoonlijke behoeftes van de student. Leren via deze manier geeft vaak betere resultaten dan leren in een klassieke klaslokaal omgeving. Het positieve effect van een-op-een studiebegeleiding heeft gediend als een van de inspiratiebronnen voor intelligente tutor systemen. Een intelligente tutor systeeme (ITS) is een computersystemen dat studenten of leerlingen ondersteuning kan bieden bij het leren [2].

In de loop der jaren zijn een aantal verschillende architecturen voor intelligente tutor systemen voorgesteld. Één daarvan wordt vaak aangehaald als de algemene architectuur van een ITS [3]. Volgens deze architectuur bestaat een ITS uit vier basiscomponenten met ieder hun eigen functie [2][4][5]. Een overzicht van deze basiscomponenten en hun relaties wordt weergeven in figuur 1.

De *domeinmodule*, ook wel expert module genoemd, bevat informatie en regels uit het domein waarover de ITS kennis moet hebben om de student te kunnen ondersteunen. De manier waarop deze kennis gerepresenteerd wordt, kan verschillen per systeem. Door de domeinmodule is een ITS in staat om te redeneren over het gespecificeerde domein, problemen uit het domein op te lossen en te kunnen redeneren over de progressie van de student. Kennis over het niveau van opdrachten en



Figuur 1.1: Architectuur intelligent tutor systeem

nakijkmodellen zijn voorbeelden van concepten die onder de domeinmodule zouden vallen.

De *studentmodule* dient als een dynamisch model over de kennis van een student. Informatie over de student wordt verzameld en verwerkt binnen de studentmodule en op basis hiervan kan een relevante representatie van de student worden gevormd. Welke opdrachten de student correct heeft gemaakt kan bijvoorbeeld worden vastgelegd in de studentmodule.

De *tutormodule* heeft als functie om het leerproces te sturen. Op basis van de informatie uit de studentmodule gecombineerd met de tutormodule kunnen beslissingen worden gemaakt over de onderwijs-strategie van de ITS. De tutormodule is onder andere verantwoordelijk voor de keuze welk materiaal aan de student wordt gepresenteerd. Als een systeem adaptief is, is het proces dat hieraan verbonden is onderdeel van de tutormodule.

Het *interface-component* maakt het mogelijk voor de student om te kunnen communiceren met het systeem.

Op het eerste oog kunnen bestaande ITSen uiteenlopende structuren hebben en lijkt de vier-componenten-structuur soms niet terug te vinden. Dit wordt veroorzaakt doordat het doel en de functie van een ITS bepalend zijn voor zijn structuur en deze zijn vanzelfsprekend niet gelijk voor elke ITS [3]. In de praktijk zijn deze vier componenten wel terug te vinden in bestaande ITSen. Door extra functies en modules lijken ITSen vaak een andere meer complexe structuur te hebben, echter kunnen deze functies vaak worden ondergebracht bij een van de vier basis modules uit het generieke model [6]. Daarnaast kunnen de implementaties de vier verschillende componenten soms gedeeltelijk in elkaar overlopen, waardoor de structuur van het systeem ook anders oogt.

1.2 ITS voor programmeren

Er zijn enkele aspecten specifiek voor intelligente tutor systemen met een domein over programmeren. Om deze reden is ervoor gekozen om een systemen te onderzoeken die onder dit specifieke domein vallen. Aspecten die specifiek zijn voor het domein over programmeren zijn onder andere het aanbod van verschillende soort opdrachten die gemaakt kunnen worden en het belang van het krijgen van persoonlijke feedback tijdens het leerproces.

De effectiviteit van het maken van een opdracht wordt bepaald door het niveau van de student en het soort opdracht. Om programmeren te leren kunnen verschillende soorten opdrachten worden gemaakt. Veel systemen maken gebruik van opdrachten waarbij de student zelf code moet schrijven of code moet debuggen. Uit een onderzoek naar het vermogen om te kunnen debuggen onder studenten bleek dat meer dan de helft van de studenten die in staat waren om te debuggen ook gevorderde programmeurs waren [7]. Daarnaast zijn beginnende programmeurs vaak niet in staat om een model te kunnen maken dat nodig is voor het construeren van code [8]. Om deze redenen zijn andere opdrachten voor beginnende programmeurs de laatste jaren in populariteit gestegen. Dit zijn opdrachten zoals Parson problemen. Bij deze opdrachten is het de bedoeling om code te herorganiseren tot het programma dat omschreven wordt, zonder hierbij zelf code te schrijven [9]. Onder beginnende programmeurs blijkt dat dit een effectievere manier is van leren. Dit

leidt ertoe dat het waardevol is om onderzoek te doen naar adaptieve studentbegeleiding.

Het lastige van goede feedback geven bij programmeeropdrachten is dat dit een tijdrovend proces is [10]. Voor een docent kost het veel tijd om een fout in de code van student te vinden die de oplossing incorrect maakt. Er bestaan wel systemen die automatisch kunnen testen of de output van een programma overeenkomt met de gewenste output gespecificeerd door de opdracht. Dit geeft echter vaak nog geen informatie over fouten die een student maakt. Met automatische en adaptieve feedback kan een ITS hier een belangrijke rol spelen.

1.3 Outline van dit onderzoek

Dit werk zal zich focussen op hoe gepersonaliseerd leren kan worden gerealiseerd binnen ITSen die bestemd zijn om te leren programmeren. De afgelopen jaren is er binnen de ontwikkeling van ITSen steeds meer belangstelling voor adaptiviteit [11]. Adaptiviteit zal in dit werk worden beschouwd als het vermogen van een ITS om zich aan kunnen te passen aan de student. Dit heeft als resultaat dat het leren wordt gepersonaliseerd en het systeem beter is afgestemd op de student. Om de functies van ITSen te kunnen personaliseren, moet er voldoende informatie beschikbaar zijn over de student. De studentmodule speelt dus een belangrijke rol in het verwezenlijken van adaptiviteit. Het is relatief simpel om een algemene indruk van de student te construeren die gebaseerd is op de resultaten die de student binnen het systeem heeft behaald. Dit geeft echter geen complete representatie van de student, om dit te behalen zijn van observaties van fouten en gedrag van de student over een lange termijn nodig [12]. Het is dus van belang voor personalisatie dat er genoeg gegevens over de student verzameld worden.

Naast het goed kunnen modelleren van de student, is het van belang dat het systeem flexibel is. Een systeem dat is ontwikkeld om feedback te kunnen geven moet bijvoorbeeld in staat zijn om feedback te geven die is aangepast op ingeleverde opdrachten van de student zelf. Als het systeem dezelfde feedback genereert bij verschillende opdrachten die ingeleverd zijn door verschillende studenten, noemt men dit systeem niet adaptief. Het systeem moet in die zin dus flexibel zijn om zich aan te kunnen passen aan de behoeftes van de student om adaptiviteit daadwerkelijk te kunnen verwezenlijken.

Er zal in dit werk worden onderzocht welke verschillende methodes er zijn om adaptiviteit te implementeren in ITSen. Om adaptiviteit in een ITS te implementeren zal dus in ieder geval van een het systeem worden geëist dat deze een betrouwbare representatie van de student heeft en dat deze flexibel is. Hierbij zal expliciet worden ingegaan op hoe dit wordt gerealiseerd bij het genereren van automatische feedback en adaptieve studentbegeleiding. De meeste toepassingen van adaptiviteit in intelligente tutor systemen binnen het domein van programmeren gebruiken één van deze twee concepten om adaptiviteit te verwezenlijken [11]. De exacte implementatie van adaptiviteit staat echter vrij. Er kan gevarieerd worden met elke gegevens van studenten exact worden gebruikt, hoe een systeem hiermee omgaat en hoe ervoor wordt gezorgd dat een systeem zich hierop kan aanpassen. De complexiteit van adaptiviteit van een ITS wordt bepaald door de benodigdheden om adaptiviteit mogelijk te maken.

De rest van dit werk is als volgt opgebouwd: in het tweede hoofdstuk zal adap-

tieve studentbegeleiding worden besproken aan de hand van twee concepten, adaptieve probleemselectie en adaptieve navigational support. In het derde hoofdstuk wordt dieper ingegaan op het realiseren van adaptiviteit met het genereren van persoonlijke feedback. In het vierde hoofdstuk worden een conclusie en suggesties voor vervolgonderzoek gegeven.

1.4 Wetenschappelijke relevantie

ITSen zijn in staat om intelligent gedrag te vertonen door op te treden als docent. Ze beschikken over het vermogen om zelfstandig na te kijken of studenten te voorzien van hints. Wetenschappers binnen de kunstmatige intelligentie zijn onder andere geïnteresseerd naar de intelligentie die computersystemen kunnen ontwikkelen [13]. Enerzijds heeft dit betrekking op het interne proces van deze systemen en hoe deze in vergelijking staat tot de denkwijze van de mens. Anderzijds ligt de focus meer op het gedrag van deze systemen, op het externe en wat zichtbaar is voor de buitenwereld. Hier nemen intelligente tutor systemen hun plaatsen binnen de kunstmatige intelligentie. Onderzoek naar de werking van deze systemen leiden tot meer inzicht over hoe menselijk en intelligent gedrag door computersystemen gesimuleerd kan worden.

Hoofdstuk 2

Adaptieve student-begeleiding

In dit hoofdstuk wordt besproken hoe persoonlijk leren kan worden bevorderd door studenten sturing te geven tijdens het leerproces. Dit is mogelijk door studenten te laten navigeren door de stof en opdrachten automatisch toe te wijzen. Verschillende methodes en hun toepassingen zullen individueel worden besproken en vervolgens met elkaar worden vergeleken aan de hand van vastgestelde criteria.

2.1 Adaptieve studentbegeleiding

In het tweede hoofdstuk is beargumenteerd wat de toevoegde waarde is van het maken van verschillende opdrachten die aangepast zijn op het niveau van de student. Het is van belang dat een student voldoende oefent met een specifiek concept om het onder de knie te krijgen. Een mogelijk probleem is dat studenten niet altijd in staat zijn om voor zichzelf in te schatten op welk niveau zij zitten. Dit kan als gevolg hebben dat zij of opdrachten kiezen van boven hun niveau met als resultaat dat zij gedemotiveerd raken, of dat zij opdrachten kiezen die te makkelijk zijn waardoor verveling ervoor kan zorgen dat studenten niet worden uitgedaagd om nieuwe stof te leren [14] [15]. Bovendien is het bewezen dat studenten beter presteren als de volgorde van de gepresenteerde lesstof die aan de student wordt gebaseerd op het leerproces van de student [16]. Hieruit kan geconcludeerd worden dat het wenselijk is dat een ITS in staat is om de volgorde van de leerstof die de student moet beheersen afhankelijk is van het niveau van de student. Dit kan onder andere worden geïmplementeerd met adaptieve opdrachttoekenning. Een ITS maakt tijdens het leerproces keuzes over geschikte opdrachten voor de student en baseert deze keuzes op basis van de informatie die bekend is over de student. Als alternatief voor adaptieve probleem selectie kan in een ITS *navigational support* worden geïmplementeerd. Een groot verschil met adaptieve probleem selectie is dat bij navigational support de student niet een probleem wordt opgelegd, maar de keuze over welke opdracht gemaakt wordt in de handen van de student blijft [17]. Een ITS heeft hier meer een passieve functie door de student te helpen in de vorm van suggesties voor opdrachten en nuttige bronnen uit de leerstof. Deze suggesties zijn wederom gebaseerd op informatie over de student die opgeslagen is in de studentmodule. De suggesties zullen worden weergegeven in het interface-component van een ITS en zijn zo zichtbaar voor de student [15].

Voor beide concepten zullen implementaties worden besproken die zijn toegepast in een ITS. Hierbij zal een vergelijking worden gemaakt tussen verschillende

toegepaste methodes. Deze vergelijking wordt gebaseerd op een aantal aspecten. Ten eerste zal worden bekeken in welke mate de gegevens van de student worden gebruikt om adaptiviteit toe te passen. Ten tweede wordt gekeken naar de flexibiliteit van een ITS. Er zal worden vergeleken in welke mate het systeem kan variëren op basis van verschillende studenten. Hoe meer een systeem in staat is om te kunnen variëren, hoe beter het systeem zich kan aanpassen en de stof die gepresenteerd wordt zal aansluiten op de student. Als laatste zal de focus liggen op hoe de techniek is toegepast en getest om te garanderen dat er correcte uitspraken kunnen worden gedaan over de kwaliteit van een specifieke techniek.

2.2 Adaptieve opdrachttoekenning

Bij regel-gebaseerde opdrachttoekenning zal het proces om adaptiviteit te kunnen realiseren afhangen van regels die bepaald zijn door de makers van de ITS of experts van het desbetreffende domein. De input voor deze regels zijn gegevens over de student die verzameld zijn in de studentmodule. Vervolgens kan aan de hand van één of meerdere regels bepaald worden of een student over genoeg kennis beschikt over een bepaald concept. De exacte invulling van de regels en bijbehorende input kan net als een studentmodule verschillen per systeem. Regel-gebaseerde opdrachttoekenning is tot nu toe de enige techniek die gevonden is in de literatuur en daarnaast zowel toegepast als getest is. De enige twee gevonden systemen die adaptieve probleemselectie toepassen zullen om deze reden uitgebreid worden besproken.

Het systeem **PyKinetic** is een python tutor voor studenten die bezig zijn met een introductie in programmeren [18]. **PyKinetic** maakt gebruik van opdrachten die van tevoren zijn geclassificeerd. Iedere klasse in het systeem komt overeen met een zogenoemd *level*. De levels representeren het niveau van laag naar hoog. De lagere levels komen overeen met de opdrachten die als relatief makkelijker worden gezien, zoals Parson problemen. De hogere levels zijn opdrachten die horen bij een hoger niveau, zoals opdrachten waarbij de student zelfstandig code moet schrijven of debuggen. Om te bepalen of de student over genoeg kennis voor het volgende niveau beschikt, maakt **PyKinetic** gebruik van een zogenoemde *Activity Score*. Deze wordt berekend volgens de twee onderstaande functies.

$$ActivityScore = (0.5 \cdot TimeScore + 0.5 \cdot AttemptScore) - Penalty \tag{2.1}$$

$$Penalty = 0.17 - AttemptTime \tag{2.2}$$

Hierin is de *TimeScore* de deling van het totaal aantal tijd door de optimale tijd en de *AttemptScore* de deling van het aantal pogingen gedaan door de student door het optimale aantal pogingen. Er wordt een *Penalty* toegewezen als de student minder dan tien seconden over de poging deed. Dit wordt dan gezien als een poging waarbij de student zich niet heeft ingezet om de opdracht serieus te maken. De *AttemptTime* is dan het aantal minuten dat de student over de poging heeft gedaan. Helaas biedt het artikel geen details over de berekening van het optimale aantal pogingen of optimale tijdsduur. Ook wordt niet genoemd hoe de uiteindelijke *ActivityScore* wordt gelinkt aan het level van de volgende te maken opdracht voor de student. **PyKinetic** is getest op een groep van 32 studenten die een cursus introductie programmeren volgen. Van deze groep kreeg de controlegroep een vaste volgorde van

opdrachten, de experimentele groep kreeg opdrachten toegewezen door middel adaptieve probleemselectie. Alle studenten maakten een pre- en post-test om uitspraak te kunnen doen over de kwaliteit van de adaptieve probleemselectie. De studenten uit de experimentele groep losten significant meer problemen van hogere levels op dan studenten uit de controlegroep. Dit duidt erop dat **PyKinetic** in staat is om studenten meer uitdaging te bieden behorend bij het niveau van de studenten. Er was echter geen significant verschil tussen de verbetering van de gemiddelde scores van de pre- en post-test tussen de twee groepen.

Een andere ITS die gebruik maakt van adaptieve probleemselectie is bestemd voor het leren van C++ [19]. Om adaptieve probleemselectie te implementeren zijn binnen het systeem verschillende concepten met bijbehorende opdrachten gedefinieerd. In de studentmodule wordt voor ieder concept een aantal variabelen gedefinieerd; het aantal gegenereerde opdrachten (G), het aantal pogingen dat is gedaan (A), het aantal correct opgeloste opdrachten (R), het aantal fout gemaakte opdrachten (W) en de opdrachten van dat concept die nog niet door de student gemaakt zijn (M). Het systeem gebruikt de volgende twee regels tijdens het keuzeproces:

$$M_1 \leq A \tag{2.3}$$

$$R/A \geq M_2 \tag{2.4}$$

Met regel 2.3 wordt een minimum afgedwongen voor het aantal problemen dat de student correct moet maken. Met de tweede regel 2.2 wordt het het succespercentage berekend, dit percentage bepaalt hoeveel opdrachten de student correct moet hebben om het concept te leren. Vervolgens worden deze regels ingezet om te bepalen wat het volgende concept is waar de student zich mee bezig gaat houden. Uit dat concept wordt ook een opdracht voor de student geselecteerd. Voor het experiment met dit systeem werd een groep studenten van onbekende grootte verdeeld in twee groepen waarvan de studenten uit de controlegroep een vaste volgorde van opdrachten maakten en de studenten van de experimentele groep opdrachten kregen toegewezen met adaptieve probleemselectie. Alle studenten maakten wederom een pre- en post-test. Dit experiment is twee keer uitgevoerd. Er was geen significant verschil tussen de twee groepen wat betreft de vooruitgang die de studenten hebben gemaakt. De studenten uit de controlegroep hebben echter een significant lager aantal opdrachten gemiddeld gemaakt. Daarbij is bij de experimentele groep de gemiddelde score van de toegewezen concepten significant lager dan bij de controle groep, wat suggereert dat met adaptieve probleemselectie de tutor concepten aan de student toewijst die de student nog niet goed beheerst.

2.3 Adaptieve navigational support

2.3.1 Adaptieve hypermedia systemen

Adaptieve *navigational support* is een verzameling van technieken die ingezet kunnen worden om de weergave van hyperlinks aan te passen binnen een adaptive hypermedia systeem (AHS). Deze technieken dienen de gebruiker te ondersteunen in het vinden van waardevolle informatie of nuttig materiaal binnen het systeem [20]. De hyperlinks in de interface van een traditioneel hypermedia zijn verbonden aan media, zoals bijvoorbeeld beeld of geluid. Hierbij zijn de voorkomens en aanduidingen van de hyperlinks gelijk voor iedere gebruiker [21]. Bij adaptieve hypermedia systemen is dit niet het geval, deze bevatten naast deze verzameling van hyperlinks ook een model waarin verscheidene gegevens over de gebruiker worden opgeslagen, zoals kennis, voorkeuren en persoonlijke doelen [21]. Deze modellen worden vervolgens ingezet om de weergave van de hyperlinks in het systeem gunstig aan te passen voor de gebruiker, hierdoor biedt het systeem ondersteuning in het vinden van nuttig materiaal.

Er bestaan over het algemeen vijf technieken voor het toepassen van adaptieve navigational support [20].

- *Direct guidance* geeft een suggestie in de vorm van een hyperlink aan de gebruiker. Het systeem kiest hiervoor de beste volgende stap voor de gebruiker.
- *Link ordening* sorteert alle zichtbare hyperlinks op een pagina, waarbij de plek afhankelijk is van de relevantie van een link. Er geldt hoe hoger de link op een pagina is, hoe relevanter deze link zou zijn voor de gebruiker.
- *Link hiding* is een techniek voor het verbergen van hyperlinks. Het doel hiervan is om de zoekruimte van de gebruiker in te perken, door alleen de relevante links voor de gebruiker zichtbaar te laten. Het beschermt de gebruiker tegen een overvloed van informatie.
- *Link annotation* voorziet hyperlinks van een extra kenmerk om de relevantie van verschillende hyperlinks aan te duiden. Dit kan bijvoorbeeld door gebruik te maken van speciale iconen of kleuren.
- *Link generation* maakt het mogelijk voor een systeem om zelf hyperlinks te creëren, zonder dat deze van tevoren zijn vastgelegd door de ontwikkelaar van het systeem.

2.3.2 Navigational support in een ITS

In de context van educatieve hypermedia systemen is het doel van adaptieve navigational support om suggesties te doen voor relevante onderwerpen en opdrachten die passen bij het niveau van de student [22]. Oorspronkelijk worden de technieken van adaptieve navigational support toegepast in hypermedia systemen, dus in ander soort systemen dan intelligente tutor systemen. Adaptieve hypermedia systemen zijn erop gericht om instructies en leiding aan de gebruiker te geven, waar bij intelligente tutor systemen de focus ligt op het bieden van ondersteuning bij het oplossen van opdrachten [5]. Adaptieve navigational support kan in een ITS echter wel van

toegevoegde waarde zijn door een student hulp te kunnen bieden bij de keuze voor geschikte opdrachten. Er bestaan grote overeenkomsten tussen de architectuur van een ITS en een AHS. Beide systemen beschikken over een domeinmodule, student- of gebruikermodule en een module die beslissingen kan maken om de gebruiker te ondersteunen [23]. Daarnaast toont onderzoek aan dat een vereniging van de functionaliteit van twee systemen ook goed mogelijk is [5][24]. Uit de combinatie van deze observaties kan geconcludeerd worden dat adaptieve navigational support goed implementeerbaar is in een ITS.

Om adaptieve navigational support te implementeren moet de inhoud van de domeinmodule gestructureerd worden, zodat er duidelijke relaties zijn tussen de verschillende concepten die door de student geleerd moeten worden. Deze relaties tussen concepten kunnen bijvoorbeeld voortkomen uit feit dat het ene concept benodigde voorkennis is voor een ander concept. Het is vaak niet duidelijk hoe de ontwikkelaars van de systemen relaties in de domeinmodule construeren. In enkele gevallen wordt alleen aangegeven dat dit handmatig gebeurt aan de hand van een boek of ondersteuning van domein-experts. Er bestaan wel methodes om het domein een vaste structuur te geven, dit kan bijvoorbeeld worden gedaan met een Bayesiaanse netwerken [25][26] of *fuzzy cognitive maps* [27]. Hierbij worden relaties tussen concepten in het domein berekend op basis van de studentmodule en tijdens het leerproces geüpdatet.

De technieken van adaptieve navigational support zijn ondertussen al veelvoudig toegepast in verschillende systemen. Hierbij zijn verschillende resultaten behaald. Studenten die gebruik maakten van adaptieve navigational support significant maakten minder stappen in het systeem en maakte dezelfde opdrachten minder vaak opnieuw [28]. Daarnaast draagt het gebruik van navigational support ook bij aan verhoging van academische resultaten [29]. Om de werking van adaptieve navigational support te illustreren zal een toepassing hiervan kort worden besproken. De **JavaGuide** is een van de weinige toepassingen die studenten kan ondersteunen in het maken van een keuze tussen de beschikbare opdrachten, andere systemen geven slechts suggesties over relevant lesmateriaal in de vorm van bijvoorbeeld collegeslides [14]. Het is helaas niet duidelijk hoe de gegevens uit studentmodule in **JavaGuide** exact wordt ingezet om adaptieve navigational support toe te passen. Dit geldt ook voor de andere genoemde systemen.

JavaGuide maakt gebruik van *link annotation* op twee niveaus, het geeft suggesties voor geschikte/relevante onderwerpen en biedt inzicht over het aantal correct gemaakte opdrachten. Voor iedere individuele opdracht wordt aangegeven of deze minstens een keer correct is ingeleverd. Elke beschikbare opdracht behoort tot een specifiek onderwerp. Per onderwerp wordt met een icoon informatie gegeven die de student zou moeten helpen met de keuze voor een te maken opdracht. Het icoon duidt aan in welke mate de student de stof al beheerst, waarbij er drie niveaus van beheersing mogelijk zijn. Daarnaast is in het icoon zichtbaar of het onderwerp een leerdoel van de student is op dat moment. Een onderwerp kan worden gezien als voorkennis, huidig leerdoel of toekomstig leerdoel. Het systeem is uiteindelijk in meerdere keren experimenteel getest in drie verschillende semesters, waaruit blijkt dat studenten die gebruik maken van navigational support significant meer opdrachten maken en in het verloop van het semester hoger scoren op vragen van een hoger niveau.

2.4 Beschouwing

Er kan geconcludeerd worden dat met adaptieve probleem selectie het niveau van de gemaakte opdrachten beter is aangepast op het niveau van de student. Een student kan sneller doorstromen naar een hoger niveau als hij daaraan toe is [18] en concepten die de student nog minder goed begrijpt worden vaker toegewezen aan de student voor voldoende oefening [19]. Wat een voordelig is aan de besproken methodes voor adaptieve probleem selectie, is dat zij niet domein specifiek zijn en relatief simpel om te implementeren.

Wat helaas nog ontbreekt bij de besproken systemen voor adaptieve probleemselectie is enkele verantwoording voor de keuzes die gemaakt worden voor implementaties. Er wordt geen argumentatie gegeven voor de waardes die verschillende parameters krijgen. Het is bijvoorbeeld niet bekend waarom $M_1 = 60$ [19] of waarom de *TimeScore* en *AttemptScore* evenveel bij moeten dragen aan de uiteindelijke *ActivityScore* [18]. Er is dus ruimte voor vervolgonderzoek waarbij gevarieerd kan worden met deze parameters met kans om het positieve effect van adaptieve probleem selectie nog meer te versterken. Daarnaast zou een eventueel probleem voor **PyKinetic** kunnen zijn dat de student nooit meer terugkeert naar een lager niveau, wat een beperking is op de flexibiliteit van het ITS. Het andere voorgestelde systeem bevat geen ordening qua niveau tussen de verschillende concepten.

Opvallend is dat de resultaten van verschillende onderzoeken met navigational support niet altijd eenduidig zijn. Zoals eerder genoemd kan navigational support het aantal gemaakte opdrachten reduceren [28] of juist verhogen [14] [30]. Beide observaties worden geïnterpreteerd als iets positiefs. In het eerste geval zou het duiden op effectievere besteding van tijd, aangezien voor beide groepen uit de studie een gelijk resultaat wordt behaald, waarbij de studenten die gebruik maakten van navigational support minder opdrachten hiervoor hebben gemaakt. In het tweede geval werd een groter totaal aantal gemaakte opdrachten gezien als een positief resultaat, omdat dit zou impliceren dat adaptieve navigational support de studenten stimuleert. Hierdoor zouden de studenten later in het semester beter presteren op de complexere opdrachten.

Een mogelijke verklaring hiervoor is dat de effectiviteit van verschillende adaptieve methodes wellicht afhankelijk is van het niveau van de studenten die het systeem gebruiken. Dit is gedeeltelijk gebleken uit een onderzoek waarbij het toepassen van adaptieve navigational support geen effect had [15].

Een nadeel van adaptieve navigational support is door de vrije invulling van de weergave van de links het positieve effect soms teniet wordt gedaan. het blijkt bijvoorbeeld dat slechts het gebruik van kleur minder effectief is dan andere annotation. Daarnaast is het noodzakelijk om al het lesmateriaal en alle opdrachten uit het domein te structureren, wat veel werk kan opleveren. Er wordt echter wel onderzoek gedaan naar het automatiseren van dit proces, wat dit eventuele nadeel weg zou kunnen nemen [31].

Hoofdstuk 3

Genereren van adaptieve feedback

In dit hoofdstuk zal worden besproken wat adaptieve feedback inhoudt en hoe dit kan bijdragen aan gepersonaliseerd leren. Er zal met name worden ingegaan op verschillende technieken voor het genereren van adaptieve feedback. Deze methodes zullen vervolgens met elkaar worden vergeleken en daarbij worden de mogelijkheden van adaptieve feedback met de verschillende technieken besproken.

3.1 Adaptieve feedback

Om programmeren te leren, wordt veel oefening van de student vereist. De student moet programma's kunnen ontwerpen, schrijven en daarbij ook ze te lezen en te begrijpen [32]. Door tussentijds en regelmatig feedback te ontvangen, wordt voorkomen dat het student tijdens het leren vastloopt en gedemotiveerd raakt. Daarmee speelt het krijgen van feedback een belangrijke rol in het leren van programmeren. Een probleem is echter dat bij de groei van interesse voor het vak programmeren, docenten niet meer in staat zijn om tijdens het leerproces elke individuele student te voorzien van feedback op het moment dat de student dat zou willen. Systemen kunnen vaak wel aangeven of een gemaakte opdracht door de student correct of incorrect is, maar voor studenten is het bij het inleveren van een incorrecte oplossing vaak lastig om de oorzaak van het incorrecte antwoord te linken naar een deel uit hun code. Het is daarom waardevol dat een ITS in staat is om de student te kunnen voorzien van persoonlijke feedback.

Adaptieve feedback kan men op twee verschillende manieren interpreteren [33]. Allereerst kan feedback als adaptief worden beschouwd als de feedback die wordt gegeven informatie bevat die aangepast is op de oplossing die door de student is ingeleverd. Voor oplossingen die worden ingeleverd door verschillende studenten zou het systeem feedback moeten geven die inhoudelijk van elkaar verschillen. Om dit te illustreren, een systeem dat *automated testing* toepast is slechts in staat om als feedback aan de studenten te melden dat de ingeleverde oplossing correct of incorrect is [34]. Om deze reden wordt feedback die wordt gegenereerd volgens deze wijze gezien als feedback die niet adaptief is. Om deze reden is die techniek verder buiten beschouwing gelaten.

Ten tweede kan men feedback als adaptief beoordelen als de gegeven feedback is aangepast op basis van de beschikbare informatie over de student. Hierbij spreekt de studentmodule een belangrijke rol net als bij adaptieve studentbegeleiding. De exacte invulling van de adaptiviteit kan verschillen. Zo zou het systeem bijvoorbeeld

verschillende soorten feedback kunnen gegeven, zoals video's flowcharts of tekstuele hints. Het uiteindelijk gebruikte type hint zou dan afhankelijk zijn van de gegevens die bekend zijn over de student. Het systeem zou ook de frequentie van de feedback kunnen baseren op informatie over de student. Dit is toegepast in **SQL-tutor** die later in meer detail zal worden besproken [35]. Er zijn echter weinig toepassingen waarin feedback volgens deze invalshoek op adaptiviteit is geïmplementeerd.

In de rest van het hoofdstuk zullen verschillende technieken worden besproken die worden gebruikt voor het automatisch genereren van feedback en een duidelijke categorie vormen. Daarbij zal worden besproken hoe deze techniek adaptieve feedback genereert volgens de eerst genoemde interpretatie van adaptieve feedback. Daarna worden de mogelijkheden voor de verschillende technieken besproken om de inhoud van de feedback adaptief te maken volgens de tweede invalshoek van adaptieve feedback. Hierbij ligt de focus op de mogelijkheid om gegevens van de studenten in de techniek te betrekken.

3.2 Technieken voor feedback genereren

3.2.1 Model Tracing

De techniek *Model Tracing* is erop gericht om tijdens het leerproces de stappen die de student maakt te volgen. Er wordt een set van *production rules* gebruikt, deze representeren de stappen waarmee een correcte oplossing kan worden geconstrueerd. Tijdens het maken van de opdracht houdt het systeem bij welke stappen de student maakt. Op het moment dat de student afwijkt van de stappen die zijn vastgesteld om een correcte oplossing te maken, wordt aan de student een hint gegeven op zijn oplossing te verbeteren.

3.2.2 Constraint-Based Modelling

Het idee achter *constraint-based modelling* is dat een domein bestaat uit een verzameling van kenniseenheden, de zogenoemde *constraints.*. Een ITS die constraint-based modelling toepast kan feedback op een oplossing gegeven door de ingeleverde oplossing te vergelijken met de de verzameling van constraints. Een oplossing moet aan alle constraints voldoen om als correct te kunnen worden beschouwd. Als een constraint wordt geschonden, is dit een aanwijzing voor het systeem voor een concept dat de student nog niet onder de knie heeft of verkeerd begrijpt. Een voorbeeld van een ITS met constraint-based modelling is de **SQL-tutor** [35]. In dit systeem worden constraints ingezet om zowel positieve als negatieve feedback te genereren. Als een student een oplossing inlevert, wordt de oplossing vergeleken met 700 constraints. Daarnaast wordt voor elke student een lijst opgeslagen met relevante constraints, constraints die de student heeft geschonden en constraints waar de student aan heeft voldaan. Om positieve feedback te genereren wordt door de **SQL-tutor** naast een set van regels ook de studentenmodule gebruikt om de meest relevante feedback te selecteren. Dit maakt de **SQL-tutor** tot zover de enige ITS die informatie over de student gebruikt.

3.2.3 Intention-based Diagnosis

De techniek *Intenion-based diagnosis* streeft ernaar om te achterhalen met welke ïntentie"de student een programma heeft geschreven. Om dit te realiseren maakt het systeem gebruik van een domeinmodule die een verzameling bevat van veel voorkomende fouten en plannen voor het opstellen van een programma.

3.2.4 Program Transformation

Een ITS die gebruikt maakt van *program transformation* vergelijkt de ingeleverde oplossing van de student met een modeloplossing uit de domeinmodule. Om deze vergelijking te kunnen maken zet een ITS een ingeleverd programma om naar een andere vorm, om vervolgens feedback te kunnen genereren. Een programma kan worden omgezet naar een abstractere variant, dit gebeurt dan ook voor de modeloplossing die een ITS gebruikt als vergelijkingsmateriaal. Uit de vergelijking van de twee programma's kan informatie geven over eventuele verbetering van het ingeleverde programma. Een andere mogelijkheid met program transformation is dat de modeloplossing volgens een set regels wordt omgezet naar meerdere modeloplossingen, waarvan uiteindelijk de meest overeenkomende modeloplossing.

3.2.5 Data-driven feedback

De afgelopen paar jaar is er steeds meer aandacht voor technieken die feedback genereren op basis van een dataset van oplossingen die gemaakt zijn door studenten. Op deze manier is het niet nodig om veel tijd te stoppen in het modelleren van een manier om feedback te generen. In plaatst daarvan wordt het ingeleverde programma vergeleken met programma's uit de dataset. Een concrete toepassing hiervan is de **ITAP** tutor [36]. Om feedback te kunnen genereren vereist het systeem wel te beschikken over minimaal een correcte oplossing voor elk probleem en een methode om code automatisch te beoordelen. Dit kan bijvoorbeeld door de input en output met elkaar te vergelijken. De hints komen voort uit een zogenoemde *solution space* die door het algoritme wordt geconstrueerd. Er wordt een abstracte representatie van ingeleverde programma gemaakt op een vergelijkbare manier als bij de program transformation techniek. Vervolgens wordt met een *pathconstruction* algoritme de *solution space* aangepast. Hieruit kan vervolgens bepaald worden welke acties de student moet ondernemen om de incorrecte oplossing te transformeren naar een correcte oplossing. Er wordt daarbij gezocht naar het minimale aantal correcties die te student moet toepassen om zijn oplossing correct te maken.

3.3 Beschouwing

Voor alle genoemde technieken moet er door experts van het domein kennis worden vastgelegd om op die voor data-driven feedback. Gezien de hoeveelheid werk die dit kost kan dit worden gezien als een groot voordeel voor data-driven technieken. Daar staat wel tegenover dat bij de feedback die gegenereerd wordt door data-driven technieken vaak context of informatie over de verantwoording van de feedback ontbreekt [36] [37]. De feedback die wordt gegenereerd door data-driven technieken zijn slechts directe aanwijzingen voor aanpassingen in het door de student ingeleverde programma en hierdoor kan het voor een student niet altijd duidelijk zijn waarom hij een bepaalde hint moet toepassen. Dit is een aspect dat bijvoorbeeld wel gegeven kan worden door constraint-based modelling technieken, waarbij het voor het systeem duidelijk is welke constraint wordt geschonden. Hier zou dan meer informatie kunnen worden gegeven over de oorsprong van de fout van de student.

Het is echter lastig om een duidelijke vergelijking te maken tussen de verschillende technieken om feedback te genereren. Allereerst zou dit gedeeltelijk veroorzaakt kunnen worden door het feit dat de technieken niet altijd feedback van dezelfde vorm genereren. Om de effectiviteit van de verschillende soorten feedback te testen zou eenzelfde set van opdrachten worden kunnen worden gebruikt. Hierbij zou bijvoorbeeld binnen één systeem meerdere technieken kunnen worden geïmplementeerd, zodat studenten te maken krijgen met verschillende soorten feedback die horen bij verschillende soorten technieken.

De enige toepassing van een ITS die gebruik maakt van informatie over de student om feedback te genereren is de **SQL-tutor**. Doordat constraint-based modelling is toegepast in de **SQL-tutor** heeft het systeem meer inzicht op de stof die door de student wel of niet wordt beheerst. Dit wellicht op een soortgelijke manier worden gedaan met model tracing of intention-based diagnosis. Systemen die deze technieken toepassen zouden bijvoorbeeld informatie over veelgemaakte fouten makkelijk kunnen opslaan. Voor systemen die een data-driven techniek toepassen is dit mogelijk lastig te combineren. Er is geen enkele relatie met het informatie uit de domeinmodule over het soort fout dat de student maakt, wat het lastig maakt om in te spelen op bijvoorbeeld veelgemaakte fouten. Er is nog geen toepassing waarbij gevarieerd wordt met het soort feedback dat door het systeem wordt gegeven. Eerder is beargumenteerd dat het niveau van een programmeur bepalend is voor bepaalde vaardigheden die studenten wel of niet hebben, beginnende programmeurs hebben bijvoorbeeld moeite met het modelleren van problemen en het debuggen van programma's. Het zou voor beginnende programmeurs dan misschien van waarde zijn om feedback te ontvangen die niet specifiek gericht is op het verbeteren van een ingeleverd programma op het niveau van kleine aanpassingen in code, maar op het conceptuele niveau over het ontwerp van code.

Hoofdstuk 4

Conclusie

In dit werk werd onderzocht hoe persoonlijk leren gerealiseerd kon worden in een ITS dat bestemd is om studenten te ondersteunen bij leren programmeren. Er zijn twee verschillende methodes geanalyseerd; adaptieve studentbegeleiding en adaptieve feedback. Een systeem kan de student tijdens het leerproces begeleiden met adaptieve probleemselectie en adaptieve navigational support. Hiervan is adaptieve probleemselectie een nog vrij onbekend terrein en kan nog veel mee worden geëxperimenteerd. Adaptieve navigational support is in meer systemen al toegepast, maar liet soms nog tegenstrijdige, maar wel positieve resultaten zien. Dit maakt het moeilijk om een vergelijking te maken tussen de twee technieken. Een mogelijk nadeel van adaptieve probleemselectie is dat de vrijheid om de leerstof zelf te ontdekken de student wordt ontnomen. Daarentegen is het mogelijk dat voor sommige studenten deze sturing juist als prettig en effectief kan worden ervaren.

Daarnaast zijn er verschillende methodes geanalyseerd voor het genereren van persoonlijke feedback. Er zijn nog erg weinig toepassingen waarbij een ITS gebruik maakt van informatie over de student om de feedback hierop aan te passen. Daarnaast zijn er wel verschillende technieken ontwikkeld die automatisch feedback kunnen genereren. Deze feedback is

Dit onderzoek is geen systematisch literatuur onderzoek. Dit kan ertoe zou hebben geleid dat sommige systemen met bijbehorende resultaten buiten beschouwing zijn gelaten. Daarnaast is het lastig om concrete conclusies te trekken over adaptieve probleemselectie of adaptieve feedback die gedeeltelijk gebaseerd zijn op informatie over de student. Uitspraken over deze onderwerpen zijn door gebrek aan veelvoudige toepassingen gebaseerd op eigen observaties en analyses in plaatst van een structurele vergelijkingen.

Opvallend is dat elk systeem één methode gebruikt om persoonlijk leren te realiseren, dus of adaptieve studentbegeleiding of persoonlijke feedback. In de toekomst zou onderzoek gedaan kunnen worden naar het effect dat het combineren van deze methodes zou kunnen hebben. Daarnaast is het duidelijk dat onderzoek naar sommige aspecten van adaptiviteit binnen intelligente tutor systemen nog ontbreekt. Er bestaan weinig systemen met adaptieve probleemselectie en zo goed als geen systemen die feedback genereren volgens de opvatting dat feedback adaptief is als deze gebaseerd is op informatie over de student. De verwachting is, dat studenten van verschillende niveaus en leerstijlen baat hebben bij verschillende types feedback. Een voorstel is om meer te experimenteren met adaptieve probleemselectie en te onderzoeken hoe adaptieve feedback op een andere manier geïmplementeerd kan

worden.

Dit werk heeft een overzicht geboden van verschillende methodes en bijbehorende technieken om adaptiviteit te kunnen implementeren in intelligente tutor systemen. Er is onderzocht hoe het positieve effect van een menselijke tutor gesimuleerd kan worden, door het toepassen van technieken die een ITS. Een ITS is in staat om zelfstandig keuzes te maken over het leerproces van de student en informatie over de student hierin mee te laten tellen. Zo kunnen intelligente tutor systemen langzamerhand steeds beter bijdragen aan een persoonlijk leerproces.

Bibliografie

[1] Benjamin Bloom. "The 2 Sigma Problem: The Search for Methods of Group Instruction as Effective as One-to-One Tutoring". In: *Educational Researcher* 13 (jun 1984), p. 4–16. DOI: 10.3102/0013189X013006004.

[2] Roger Nkambou, Riichiro Mizoguchi en Jacqueline Bourdeau. *Advances in Intelligent Tutoring Systems*. Deel 308. Jan 2010, p. 1–11. DOI: 10.1007/978-3-642-14363-2.

[3] Hyacinth Nwana. "Intelligent Tutoring Systems: an overview". In: *Artificial Intelligence Review* 4 (dec 1990), p. 251–277. DOI: 10.1007/BF00168958.

[4] Martha Polson en J. Richardson. *Foundations of Intelligent Tutoring Systems*. Jan 1988. ISBN: 978-0-8058-0054-8.

[5] Pipatsarun Phobun en Jiracha Vicheanpanya. "Adaptive intelligent tutoring systems for e-learning systems". In: *Procedia - Social and Behavioral Sciences* 2 (dec 2010), p. 4064–4069. DOI: 10.1016/j.sbspro.2010.03.641.

[6] Ali Erümit en İsmail Çetin. "Design framework of adaptive intelligent tutoring systems". In: *Education and Information Technologies* (apr 2020). DOI: 10.1007/s10639-020-10182-8.

[7] Ahmadzadeh Marzieh, David Elliman en Colin Higgins. "An analysis of patterns of debugging among novice computer science students". In: deel 37. Sep 2005, p. 84–88. DOI: 10.1145/1151954.1067472.

[8] Leon Winslow. "Programming pedagogy—A psychological overview". In: *ACM SIGCSE Bulletin* 28 (sep 1996), p. 17–22. DOI: 10.1145/234867.234872.

[9] Dale Parsons en Patricia Haden. "Parson's programming puzzles: A fun and effective learning tool for first programming courses". In: deel 52. Jan 2006, p. 157–163.

[10] Ryo Suzuki e.a. "Exploring the Design Space of Automatically Synthesized Hints for Introductory Programming Assignments". In: mei 2017, p. 2951–2958. DOI: 10.1145/3027063.3053187.

[11] Tyne Crow, Andrew Luxton-Reilly en Burkhard Wünsche. "Intelligent tutoring systems for programming education: a systematic review". In: jan 2018, p. 53–62. DOI: 10.1145/3160489.3160492.

[12] Jacqueline Bourdeau en Monique Grandbastien. "Modeling Tutoring Knowledge". In: deel 308. Sep 2010, p. 123–143. DOI: 10.1007/978-3-642-14363-2_7.

[13] Stuart Russell en Peter Norvig. *Artificial Intelligence: A Modern Approach*. Jan 2009.

[14] I-Han Hsiao, Sergey Sosnovsky en Peter Brusilovsky. "Guiding students to the right questions: Adaptive navigation support in an E-Learning system for Java programming". In: *Journal of Computer Assisted Learning* 26 (jul 2010), p. 270–283. DOI: 10.1111/j.1365-2729.2010.00365.x.

[15] Peter Brusilovsky. "Adaptive Navigation Support in Educational Hypermedia: the Role of Student Knowledge Level and the Case for Meta-Adaptation". In: *British Journal of Educational Technology* 34 (sep 2003). DOI: 10.1111/1467-8535.00345.

[16] Evangelos Triantafillou e.a. "The value of adaptivity based on cognitive style: An empirical study". In: *British Journal of Educational Technology* 35 (jan 2004), p. 95–106. DOI: 10.1111/j.1467-8535.2004.00371.x.

[17] Peter Brusilovsky. "Adaptive Hypermedia: From Intelligent Tutoring Systems to Web-Based Education". In: deel 1839. Jun 2000, p. 1–7. DOI: 10.1007/3-540-45108-0_1.

[18] Geela Venise Chee, Antonija Mitrovic en Kourosh Neshatian. "Adaptive Problem Selection in a Mobile Python Tutor". In: jul 2018, p. 269–274. DOI: 10.1145/3213586.3225235.

[19] Amruth Kumar. "A Scalable Solution for Adaptive Problem Sequencing and Its Evaluation". In: jun 2006, p. 161–171. DOI: 10.1007/11768012_18.

[20] Peter Brusilovsky. "Adaptive Navigation Support". In: deel 2. Jan 2004, p. 263–290. DOI: 10.1007/978-3-540-72079-9_8.

[21] Peter Brusilovsky. "Adaptive Hypermedia". In: *User Modeling and User-Adapted Interaction* 11 (mrt 2001). DOI: 10.1023/A:1011143116306.

[22] Peter Brusilovsky. "Adaptive Educational Systems on the World-Wide-Web: A Review of Available Technologies". In: sep 1998.

[23] Paul De Bra en Geert-Jan Houben. "AHAM: A Dexter-based Reference Model for Adaptive Hypermedia". In: (mrt 1999).

[24] Aleksandra Milicevic e.a. "Integration of recommendations and adaptive hypermedia into java tutoring system". In: *Comput. Sci. Inf. Syst.* 8 (jan 2011), p. 211–224. DOI: 10.2298/CSIS090608021K.

[25] Danial Hooshyar e.a. "SITS: a solution-based intelligent tutoring system for students' acquisition of problem-solving skills in computer programming". In: *Innovations in Education and Teaching International* (jun 2016). DOI: 10.1080/14703297.2016.1189346.

[26] Cory Butz, Shan Hua en Brien Maguire. "A Web-Based Intelligent Tutoring System for Computer Programming." In: jan 2004, p. 159–165. DOI: 10.1109/WI.2004.10104.

[27] Konstantina Chrysafiadi en Maria Virvou. "A knowledge representation approach using fuzzy cognitive maps for better navigation support in an adaptive learning system". In: *SpringerPlus* 2 (dec 2013), p. 81. DOI: 10.1186/2193-1801-2-81.

[28] Peter Brusilovsky en Leonid Pesin. "Adaptive Navigation Support in Educational Hypermedia: An Evaluation of the ISIS-Tutor". In: *Journal of Computing and Information Technology* 6 (mrt 2003).

[29] Yasir Mustafa en Sami Sharif. “An approach to adaptive e-learning hypermedia system based on learning styles (AEHS-LS): Implementation and evaluation”. In: *International Journal of Library and Information Science* 3 (jan 2011), p. 15–28.

[30] Sergey Sosnovsky e.a. “Re-assessing the Value of Adaptive Navigation Support in E-Learning Context”. In: deel 5149. Jul 2008, p. 193–203. DOI: 10.1007/978-3-540-70987-9_22.

[31] Michael Yudelson, Peter Brusilovsky en Sergey Sosnovsky. “Accessing Interactive Examples with Adaptive Navigation Support.” In: jan 2004, p. 0-. DOI: 10.1109/ICALT.2004.1357682.

[32] Amruth Kumar. “Learning Programming by Solving Problems”. In: deel 117. Jan 2002, p. 29–39. DOI: 10.1007/978-0-387-35619-8.

[33] Nguyen-Thinh Le. “A Classification of Adaptive Feedback in Educational Systems for Programming”. In: *Systems* 4 (mei 2016), p. 22. DOI: 10.3390/systems4020022.

[34] Hieke Keuning, Johan Jeuring en Bastiaan Heeren. “A Systematic Literature Review of Automated Feedback Generation for Programming Exercises”. In: *ACM Transactions on Computing Education* 19 (sep 2018), p. 1–43. DOI: 10.1145/3231711.

[35] Antonija Mitrovic, Stellan Ohlsson en Devon Barrow. “The effect of positive feedback in a constraint-based intelligent tutoring system”. In: *Computers and Education* 60 (jan 2013), p. 264–272. DOI: 10.1016/j.compedu.2012.07.002.

[36] Kelly Rivers en Kenneth Koedinger. “Data-Driven Hint Generation in Vast Solution Spaces: a Self-Improving Python Programming Tutor”. In: *International Journal of Artificial Intelligence in Education* 27 (okt 2015). DOI: 10.1007/s40593-015-0070-z.

[37] Wei Jin e.a. “Program Representation for Automatic Hint Generation for a Data-Driven Novice Programming Tutor”. In: deel 7315. Jun 2012, p. 304–309. ISBN: 9783642309496. DOI: 10.1007/978-3-642-30950-2_40.