# Stichting NIOC en de NIOC kennisbank

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

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

Op dit moment bevat de NIOC kennisbank alle bijdragen, incl. die van het laatste congres (NIOC2023, gehouden op donderdag 30 maart 2023 jl. en georganiseerd door NHL Stenden Hogeschool). Bij elkaar bijna 1500 bijdragen!

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

Het eerstvolgende NIOC vindt plaats op donderdag 27 maart 2025 in Zwolle en wordt dan georganiseerd door Hogeschool Windesheim. Kijk op www.nioc2025.nl voor meer informatie.

Wil je op de hoogte blijven van de ontwikkeling rond Stichting NIOC en de NIOC kennisbank, schrijf je dan in op de nieuwsbrief via

www.nioc.nl/nioc-kennisbank/aanmelden nieuwsbrief

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

R. Smedinga kennisbank@nioc.nl.

Vermeld bij reacties jouw naam en telefoonnummer voor nader contact.

8

# Software-architectuur in het afstandsonderwijs

*Bastiaan Heeren* - Faculteit Informatica,
Open Universiteit Nederland

*Sylvia Stuurman* - Faculteit Informatica,
Open Universiteit Nederland, Postbus 2960, 6401 DL Heerlen {bhr,stm}@ou.nl

SAMENVATTING

In dit artikel bespreken we een experimenteel ontwikkeltraject waarin de mastercursus Software Architectuur van de Universiteit Utrecht geschikt is gemaakt voor het afstandsonderwijs van de Open Universiteit Nederland. De cursusopzet moest dermate flexibel zijn dat deze ook nog inzetbaar was als onderdeel van korte studieprogramma's, waarvoor een beperkt aantal groepsbijeenkomsten wordt georganiseerd. We beschrijven de overwegingen en gemaakte keuzes bij het herinrichten van de cursus, samen met onze eigen ervaringen en reacties van de studenten.

TREFWOORDEN:

Software architectuur, afstandsonderwijs

INTRODUCTIE

Software architectuur is een vakgebied binnen de Informatica dat steeds meer aandacht krijgt vanuit de industrie, de onderzoeksinstellingen en het Informatica onderwijs. Wat er precies onder software architectuur wordt verstaan en waar de grenzen van het vakgebied liggen is niet altijd duidelijk: de onderwerpen kunnen variëren van de samenhang binnen een organisatie en de bedrijfsprocessen tot software technologische concepten zoals componentmodellen en formele architectuur beschrijvingstalen (ADL). Hoewel de Open Universiteit Nederland (OU) al vrij vroeg het belang van dit vakgebied heeft onderkend, is de cursus getiteld *Software Architectuur* pas in 2007 van start gegaan. Redenen van dit uitstel waren ondermeer de onduidelijke grenzen van het vakgebied, het ontbreken van een consensus over de inhoud van het vak, en de afwezigheid van de noodzakelijke expertise bij de OU.

De cursus is uiteindelijk tot stand gekomen in een samenwerkingsverband met de Universiteit Utrecht. In Utrecht werd het vak al geruime tijd gegeven, en was dus al de nodige ervaring opgedaan. De cursus *Software Architectuur* was één van de drie cursussen afkomstig uit deze samenwerking, maar daar waar de andere twee cursussen (over grammatica's en over het implementeren van programmeertalen) redelijk eenvoudig in te passen waren in het afstandsonderwijs van de OU was dit niet het geval bij de cursus *Software Architectuur*. De cursus in Utrecht bestond uit een reeks hoorcolleges en een omvangrijke opdracht waarvoor in vijftallen een software architectuur document opgesteld diende te worden. Iedere groep kreeg een echte (externe) opdrachtgever toegewezen om de rol van klant te vervullen. Deze onderwijsvormen zijn slecht toepasbaar binnen het afstandsonderwijs dat kenmerkend is voor de OU. Deze mismatch maakt dit specifieke geval van het hergebruiken van onderwijsmaterialen en het omvormen van de cursusopzet tot een interessante casus.

In werkelijkheid lag de situatie bij de OU nog iets ingewikkelder. Naast het geschikt maken van de cursus voor studenten in de masteropleiding van de OU was het aantrekkelijk om de cursus ook onderdeel te maken van het Certified Professional Program (CPP) getiteld Gecertificeerd Software Architect.
Zo'n programma is een kort studieprogramma voor professionals bestaande uit een aantal opeenvolgende cursussen met een duidelijke lijn, welke wordt afgesloten met een diploma. Het programma in kwestie bestond uit de cursussen Analyseren en Ontwerpen met UML, Design Patterns en Component-Based Development, en werd al enige tijd aangeboden. In dit programma worden per cursus vijf begeleidingsbijeenkomsten georganiseerd: de rest blijft over voor zelfstudie. De omvang per cursus is aanzienlijk kleiner dan die in Utrecht (4,3 ECTS in plaats van 7,5 ECTS) wat nog een ander praktisch probleem opleverde.

Zelfs in het reguliere onderwijs is de didactiek van een vak over software architectuur problematisch. Het vakgebied bestaat niet uit een welomschreven theorie met harde regels, maar eerder uit een verzameling wijze raadgevingen, enkele feitjes, en gedocumenteerde best practices. Een illustratief voorbeeld van zo'n raadgeving is de Architecture Tradeoff Analysis Method (ATAM) [2], een bekende methode om een architectuur te evalueren. Deze methode schrijft negen stappen voor om een architectuur te evalueren, en suggereert zelfs een tweedaagse agenda inclusief koffiepauzes en lunches. Waarom nu juist deze stappen tot het beste resultaat leiden wordt slechts matig onderbouwd, behalve dan dat het in de praktijk goed blijkt te werken. Het inherente gevaar van onderwijs over software architectuur is dat studenten na afloop van de cursus de raadgevingen hebben bestudeerd, maar niet in staat zijn om zelf een architectuurontwerp op te stellen of te evalueren. Een deel van de kennis kan enkel worden verkregen door het zelf te hebben gedaan. In dit artikel beschrijven we de cursus Software Architectuur van de Open Universiteit, welke na samenwerking met een reguliere universiteit geschikt is gemaakt voor het afstandsonderwijs. We bespreken de opzet van de cursus en de keuzes die zijn gemaakt. Verder blikken we terug aan de hand van onze eigen ervaringen en de evaluaties en reacties van studenten. Dit artikel is met name bedoeld voor andere universiteiten, hogescholen en commerciële opleidingen die een invulling proberen te geven aan het vakgebied van de software architectuur, of die de verschillende mogelijke werkvormen afwegen om een praktijkgericht vak zoals software architectuur te doceren.

De rest van het artikel is als volgt opgebouwd: als eerste bespreken we de nieuwe cursusopzet en hoe een reeks collegeslides is omgevormd tot een werkboek dat zelfstandig kan worden bestudeerd (sectie 2). Daarna schetsen we de aangepaste eindopdracht in sectie 3, die ondanks de beperkingen van onderwijs op afstand wel zo realistisch mogelijk moest blijven. We eindigen met een reflectie op de gekozen opzet (sectie 4) en met enkele afsluitende woorden.

NIEUWE CURSUSOPZET

De leerdoelen voor de OU cursus zijn onveranderd overgenomen. In grote lijnen moet een student in staat zijn om:

- voor een gegeven probleem verschillende oplossingen te kunnen evalueren en vergelijken, op het architectuur niveau;
- een architectuur te beschrijven en te modelleren, door middel van diagrammen en tekst;
- conflicterende belangen van stakeholders te ontdekken, en een manier vinden om die conflicten op te lossen;
- de betekenis van begrippen uit te kunnen leggen, zoals standaard architecturen, kwaliteitsmodellen, patronen voor bedrijfsapplicaties, architectuur beschrijvingstalen, modellen en de IEEE 1471 standaard.

Het materiaal voor de OU-cursus bestaat uit een tekstboek, een werkboek en een reader met artikelen. Een student wordt beoordeeld door middel van een openboek tentamen met open vragen en aan de hand van een eindopdracht. Als eerste bespreken we de totstandkoming van de reader en het tekstboek.

***Reader***

Bij het samenstellen van de reader stuitten we op twee problemen: studenten in Utrecht hebben de beschikking over de universiteitsbibliotheek, en een groot aantal relevante boeken waren ook binnen de faculteit in te zien. Bij de colleges werd dan ook verwezen naar een groot aantal boeken en tijdschriftartikelen als achtergrondinformatie of als illustratie bij wat er op het college werd verteld.

Voor de OU-cursus moesten we de omvang van deze leesopdrachten terugbrengen, en alle bronnen meeleveren omdat het voor studenten op afstand veel tijdrovender is om boeken uit een universiteitsbibliotheek te lenen. Als oplossing voor dat probleem hebben we gekozen voor het tekstboek van Albin [1] en een samengestelde reader met een beperkt aantal artikelen, met leesopdrachten in het werkboek. Op de cursuswebsite worden bovendien actuele links naar extra informatie bijgehouden.

***Werkboek***

Het werkboek is tot stand gekomen door voor elk college een leereenheid op te nemen met een inleiding, een opsomming van de leerdoelen, een aantal leesopdrachten (voor hoofdstukken uit het tekstboek of artikelen uit de reader), de kern van de leereenheid met de op het college behandelde stof, en een aantal discussievragen.

De leerdoelen vertellen een student wat van hem verwacht wordt: aan de hand van de leerdoelen kan een student bepalen of hij onderdelen van de stof moet kunnen reproduceren, of alleen begrijpen, of kunnen hanteren in een concrete situatie. De volgende leerdoelen zijn bijvoorbeeld geformuleerd voor de eerste, inleidende leereenheid.

Na het bestuderen van deze leereenheid wordt u geacht in staat te zijn om:

- uit te leggen wat de bestaansreden van een architectuur is;
- de relatie tussen stakeholders en de architectuur van een systeem te beschrijven;
- uit te leggen welke rol het sluiten van compromissen speelt bij het opstellen van een architectuur;
- het verband te beschrijven tussen architectuur en hergebruik.

De discussievragen worden bij de bijeenkomsten gebruikt om studenten te activeren en discussies uit te lokken: tijdens die discussies werd de onderbouwing van een standpunt extra benadrukt. Voor studenten op afstand helpen de discussievragen om te reflecteren op de behandelde stof. De vragen die studenten op het openboek tentamen krijgen zijn van eenzelfde aard: studenten moeten tijdens het tentamen vooral laten zien dat ze kunnen redeneren over voor- en nadelen van de verschillende keuzes waarvoor een software architect in de praktijk kan komen te staan. Naast deze discussievragen krijgen studenten ook een uitgewerkt voorbeeldtentamen om zich op het tentamen voor te bereiden.



Welkom door theatergroep Art of Events voor het avondprogramma

De keuze van de leerstof is geheel bepaald door de samenstelling van het vak zoals dat in Utrecht gegeven wordt. De leerstof bevat:

- een inleiding in het vakgebied;
- twee leereenheden over de primaire werkzaamheden van een software architect: requirements engineering, en het beschrijven en evalueren van software architecturen;
- vier hoofdstukken over patronen (gedocumenteerde bewezen oplossingen voor terugkomende problemen): architectuurpatronen, patronen voor bedrijfsapplicaties, integratiepatronen en service oriëntatie;
- twee leereenheden over onderliggende technieken: componentmodellen en design by contract.

Bij het schrijven van de leerkern is de grootste inspanning gaan zitten in het bedenken van voorbeelden ter illustratie van begrippen. Bij een hoorcollege komen voorbeelden als vanzelf ter sprake, hetzij naar aanleiding van vragen van studenten, hetzij in discussies, hetzij 'on the fly' tijdens het presenteren. Een geschreven werkboek dat alleen uitschrijft wat er op de sheets voor een college staat kan de stof niet tot leven wekken, en blijft op zo'n hoog abstractieniveau steken dat het voor studenten moeilijk wordt om te leren begrijpen wat de begrippen in de praktijk betekenen. Dat geldt in extra mate voor het vak softwarearchitectuur, omdat er voor de meeste begrippen geen scherpe definities te geven zijn: de belangrijkste functie van veel begrippen is laten zien wat de problemen zijn waar een softwarearchitect voor gesteld wordt, en wat de aanpak is om tot een oplossing te komen. Veel begrippen komen dus pas tot leven bij het werkzaam zijn als softwarearchitect.

Een voordeel van de OU hierbij is dat studenten veelal werkzaam zijn binnen de IT, en zich daardoor goed voor kunnen stellen wat de rol van een software architect bij een complex project is. Aan de andere kant geldt dat niet voor elke student: cursussen moeten zo zijn ingericht dat ook studenten die niet zo'n werkomgeving hebben de studie kunnen volgen.

Extra lastig bij het schrijven van een werkboek aan de hand van colleges is dat studenten op afstand veel minder snel een vraag zullen stellen wanneer iets ze niet duidelijk is. Bij het uitschrijven van een werkboek moet de auteur er dus alert op zijn dat lastige stof liefst op verschillende manieren wordt uitgelegd, en dat de voorbeelden die ter illustratie worden gebruikt geen onduidelijkheid scheppen.

De punten van aandacht bij het geschikt maken van een bestaand vak voor afstandonderwijs bestaan er dus uit dat expliciet gemaakt moet worden wat er van studenten verwacht wordt, dat de leerstof concreet gemaakt moet worden met voorbeelden die aansluiten bij de voorkennis die we veronderstellen, en dat er een evenwicht gezocht moet worden tussen extra heldere uitleg (omdat de student over het algemeen geen vragen zal stellen) en het zelf leren redeneren van de student (te stimuleren door middel van discussievragen waar geen eenduidig antwoord op valt te geven).

### EINDOPDRACHT

Aan de eindopdracht is noodzakelijkerwijs een hele nieuwe invulling gegeven. De eindopdracht bestaat, net als in Utrecht, uit het opstellen van een software architectuur document van een realistisch systeem. Dit maakt de cursus praktisch, en het helpt studenten om de behandelde theorie beter te begrijpen door deze zelf toe te passen. De opzet uit Utrecht was om verschillende redenen niet direct bruikbaar:

- De opdracht werd in groepen van 4 à 5 personen uitgewerkt. Zo'n groepsopdracht vereist onderlinge coördinatie en overleg. Dit is problematisch aangezien OU studenten vaak een verschillend studietempo hebben, en vanwege banen en overige werkzaamheden voor andere studiemomenten kiezen. Er zijn bovendien niet voldoende masterstudenten actief om meerdere startmomenten per jaar voor een dergelijke groepsopdracht te faciliteren.
- De omvang van de opdracht was te groot, deels omdat het op vijftallen was afgestemd, deels omdat er meer studiepunten aan de cursus verbonden zijn (7,5 ECTS ten opzicht van 4,3 ECTS bij de OU). In de nieuwe opzet was er plaats voor een opdracht van ongeveer 40 uur per student.
- Alle groepen in Utrecht kregen een externe opdrachtgever toegewezen om de rol van klant te vervullen. Onderdelen zoals het achterhalen van de requirements door middel van een interview en het presenteren van een ontwerp aan de stakeholders kregen door deze aanpak een bijzonder realistisch karakter. In het afstandsonderwijs is dit niet te realiseren door het vele afstemmen dat deze werkvorm vereist.

Vanwege de bovenstaande argumenten is er besloten om de opdracht individueel te laten maken, en deze volledig te herschrijven. De moeilijkheid bij het herformuleren van de opdracht was om de opdracht toch zo realistisch mogelijk te houden, met echte problemen uit de praktijk, maar met minder middelen en in minder tijd. Bij voorkeur zouden bestaande documenten gebruikt moeten worden om de 'echtheid'' van de opdracht te garanderen.

De keuze is uiteindelijk gevallen op het opstellen van een software architectuur document voor de Bachelor Master Schakelsite (BaMaS) [5] van de Digitale Universiteit. Deze website geeft informatie over de schakelmogelijkheden tussen de verschillende bachelor- en masteropleidingen in Nederland, aan bachelorstudenten, maar ook aan middelbare scholieren en voorlichters. De website is vrij toegankelijk en te vinden op www.bamas.nl, wat het systeem zeer concreet maakt (maar zie ook de kritische kanttekening in de volgende sectie). Omdat de OU betrokken is geweest bij het ontwerpen en realiseren van het BaMaS systeem waren er meerdere ontwerpdocumenten voorhanden. De documenten over BaMaS zijn ook opgenomen in het TISO project [10] dat dossiers van authentieke automatiseringsprojecten verzamelt en beschikbaar stelt.

De documenten over BaMaS beschrijven diverse kwaliteitseisen en wensen van de verschillende betrokken partijen, zonder daarbij oplossingen in het ontwerp aan te dragen. Zo zijn de documenten bijvoorbeeld erg expliciet over de gewenste schaalbaarheid (het aantal gebruikers dat de website gelijktijdig moet kunnen bezoeken), de uitbreidbaarheid (tot Europees niveau), de portabiliteit (met betrekking tot de webbrowsers) en de beveiliging voor het auteursgedeelte om de content van de schakelmogelijkheden in te voeren. De meeste studenten zijn bij een ontwerpcursus over UML en patronen al in aanraking gekomen met het BaMaS systeem, wat een bijkomend voordeel is.

Aan het te schrijven document worden wel eisen opgelegd. Zo moet het worden opgesteld volgens de geldende normen, wat in de praktijk neerkomt op het volgen van de IEEE 1471-2000 standaard [9] waarin enkele algemene richtlijnen te vinden zijn, en het 4+1 model van Kruchten [6]. Dit model geeft vier gezichtspunten van waaruit een architectuurontwerp gegeven moet worden, waarbij de +1 staat voor een verzameling usecases om de vier views aan elkaar te relateren. Het is toegestaan om in plaats van Kruchten een alternatieve verzameling van gezichtspunten te gebruiken voor de opdracht, zoals die van het Software Engineering Institute (SEI) [3] of van Rozanski en Woods [7]. Hoewel sommige cursisten moeite hebben met deze extra vrijheid, maakt het hen wel meer bewust van de voor- en nadelen.


Studenten van het ROC Midden Nederland verzorgden de beveiliging

Er heerst geen wereldwijde consensus over de precieze indeling van een software architectuur document. De formulering van de opdracht laat de indeling dan ook vrij. Wel zijn er verschillende templates in omloop die mogen worden gebruikt, maar zelfs met zo'n template is de opdracht verre van een invuloefening. De geboden vrijheid stelt onze cursisten in staat om een template te kiezen dat aansluit bij de werksituatie, bijvoorbeeld RUP op Maat [4] of een template dat door het bedrijf zelf is ontwikkeld. Afgezien van de keuze van het template wordt er wel verlangd dat de stakeholders en de requirements zo helder mogelijk worden verwoord, dat er een kwaliteitsmodel wordt gebruikt om de requirements te classificeren (zoals Quint2 [8]), en dat alle ontwerpkeuzes en beslissingen worden beargumenteerd.

Tijdens de eerste run van de cursus in een kort studieprogramma is besloten om alle studenten de mogelijkheid te geven een conceptversie op te sturen en om deze te voorzien van commentaar. Het bleek erg nuttig om tussentijds al suggesties aan te kunnen dragen, ze te wijzen op onvolkomenheden en advies te geven bij onduidelijkheden. Dit verving tot op zekere hoogte de onderlinge discussies die er bij een teamopdracht wel zijn. De conceptversie had als neveneffect dat het stimuleerde om vroegtijdig aan de opdracht te beginnen: bij deeltijdopleidingen is het constant houden van een studietempo nog al eens een probleem.

## ERVARINGEN EN EVALUATIES

De cursus Software Architectuur is in het najaar van 2007 gestart. Inmiddels is de cursus aan vijf groepen gegeven (met de extra begeleidingsbijeenkomsten), met in totaal zo'n 65 deelnemers. Daarnaast hebben verschillende studenten van de masteropleiding de cursus afgerond zonder deze bijeenkomsten. De cursus is bijzonder goed ontvangen, wellicht door de ervaring die al was opgedaan in Utrecht. Aan de ontwikkelde eindopdracht is nog wel geschaafd, om onduidelijkheden weg te halen, en omdat slechts weinigen deze binnen de gestelde 40 uur af kregen. Toch werd juist dit onderdeel als leerzaam ervaren door de studenten.

### *Toegevoegde waarde bijeenkomsten*

Bij het aanpassen van de cursus is rekening gehouden dat deze op afstand zou worden aangeboden. Aangezien er ook een variant is inclusief begeleidingsbijeenkomsten is de vraag wat de toegevoegde waarde is van deze bijeenkomsten ten opzichte van de zelfstudie variant. Vooral de discussies en de onderlinge interactie hadden een positieve bijdrage, en deze kregen na verloop van tijd een steeds prominentere rol tijdens de bijeenkomsten.

Per onderwerp zijn er meerdere discussievragen opgesteld die opzettelijk open zijn geformuleerd. Voorbeelden van vragen die tot een discussie leidden zijn:

- Do all stakeholdersconcerns carry equal weight? If not, what criteria exist for arbitrating among them?
- What is the role of software architecture with respect to agile methods such as Extreme Programming?
- What tactics would you advocate to promote the security sub-characteristic?

Cursisten met een andere achtergrond hebben dikwijls hele andere opvattingen over software architectuur, en kijken anders tegen dit soort vraagstukken aan. Dit maakt dat er inhoudelijk echt gediscussieerd kan worden over deze vragen, wat de bijeenkomsten een grote meerwaarde geeft. Studenten in de zelfstudie variant lopen dit helaas mis.

### *Ervaringen met eindopdracht*

Uit de evaluatie (en persoonlijke communicatie) is gebleken dat de deelnemers de eindopdracht als leerzaam ervaren, maar toch ook als omvangrijk. Vooral het beginnen met het schrijven van het document blijkt bij de meeste deelnemers een obstakel te zijn. Een verklaring hiervoor is wellicht dat de gangbare theorie en de definities niet voldoende concreet zijn om ze direct in de praktijk toe te kunnen passen. De verschillende templates voor een SAD helpen wel het schrijfproces te structureren, maar verhelpen de moeizame start niet.

Een aardige contradictie in de opdracht over BaMaS is dat er een ontwerp moet worden gegeven voor een systeem dat al bestaat. In de praktijk blijkt dit niet zo'n probleem te zijn. De functionaliteit van BaMaS is goed te overzien, wat natuurlijk wenselijk is om de opdracht niet te uitgebreid te laten zijn. De opmerking dat het systeem een rechttoe rechtaan informatiesysteem is en dus eigenlijk te simpel is voor het opstellen van een volledig software architectuur document is zo nu en dan door deelnemers geplaatst. Toch zijn er uit de opgestuurde uitwerkingen hele verschillende ontwerpen naar voren gekomen, en zijn er meerdere decomposities van het systeem denkbaar.

### *Academisch niveau*

De cursus is afgestemd op het masteronderwijs. Enige ervaring met het ontwikkelen van software en alles wat daar bij komt kijken is absoluut noodzakelijk om de rol van een software architect te kunnen begrijpen. De gemiddelde student bij de OU heeft al de nodige praktijkervaring opgedaan om de noodzaak van software architectuur in te zien.

De open formulering van de opdracht en de geboden vrijheid om het probleem aan te pakken is kenmerkend voor de masterfase. De opdracht maakt een duidelijke splitsing mogelijk tussen het inventariseren van de stakeholders, de requirements en de conflicten tussen de requirements enerzijds (de probleemanalyse), en een beargumenteerde technische uitwerking anderzijds (de oplossing). Vooral het beargumenteren van de gemaakte ontwerpkeuzes blijkt een vaardigheid te zijn die doorgaans niet aanwezig is bij onze studenten.

### *Technische onderwerpen*

In de gekozen cursusopzet is een duidelijke focus aangebracht op het software architectuur document. Een alternatief is om ook enkele technische onderwerpen te behandelen in de cursus. Een goede softwarearchitect hoort tenslotte voldoende geïnformeerd te zijn om te kunnen oordelen over de haalbaarheid van een bepaalde technische oplossing, en de consequenties van de invoering.

In de cursus komen wel enkele technische onderwerpen aan bod, maar deze spelen een minder belangrijke rol. Zo worden er enkele architectuur beschrijvingstalen genoemd, waarvan sommige een stevige theoretische onderbouwing hebben (zoals de theorie van de Communicating Sequential Processes). Dit wordt vaak als 'te academisch' ervaren. Andere voorbeelden zijn de invarianten, pre- en postcondities in het kader van de design by contract benadering, en de co-variantie en contravariantie die wordt besproken bij type systemen.

De gemiddelde student bij de OU heeft een minder formele achtergrond dan de student bij enkele technisch georiënteerde masteropleidingen waar software architectuur ook wordt gegeven. Hoewel hier rekening mee is gehouden bij het aanpassen van de cursus vinden we het wel belangrijk om ze hier iets van mee te geven.

## CONCLUSIE

Software Architectuur is een lastig vak om te doceren omdat het vooral praktische vaardigheden zijn die moeten worden geoefend. Het trainen van deze vaardigheden met de beperkingen van onderwijs op afstand is een extra uitdaging. In dit artikel hebben we de cursus besproken zoals die aan de Open Universiteit wordt gegeven. Deze cursus is geschikt voor zelfstudie en is door de studenten zeer positief ontvangen.

In de groepen van de korte studieprogramma's zaten meerdere studenten die in het dagelijks leven werkzaam zijn als software architect, of een vergelijkbare functie vervullen. Ook voor deze groep bleek de cursus een waardevolle aanvulling te zijn. Door de zeer levendige discussies die tijdens de bijeenkomsten plaatsvonden is de cursus steeds verder verfijnd.

De cursus is tot stand gekomen in samenwerking met de Universiteit Utrecht, na een experimenteel ontwikkeltraject om cursusmaterialen aan te passen aan de onderwijssituatie van de OU. De aanpassingen hebben toch nog relatief veel tijd gekost: de materialen konden niet zondermeer worden overgenomen. Wel heeft de samenwerking en het hergebruik een betere cursus opgeleverd, wat het traject zeer zeker de moeite waard heeft gemaakt. Nog steeds vindt er regelmatig overleg plaats tussen de twee universiteiten over software architectuur.

*Onze dank gaat uit naar iedereen die betrokken is geweest bij het realiseren van de cursus. In het bijzonder noemen we Johan Jeuring, Ella Roubtsova, Marleen Sint, en Lex Bijlsma die in de tijd van de samenwerking het vak Software Architectuur doceerde in Utrecht. Gert Florijn is de oorspronkelijke docent van het vak uit Utrecht en heeft dit mede vorm gegeven.*

### *Over de auteurs*

Bastiaan Heeren is als uitvoerend docent betrokken bij de cursus *Software Architectuur* zoals die wordt onderwezen binnen de korte studieprogramma's. Sylvia Stuurman is het aanspreekpunt voor studenten die de cursus op afstand volgen, en is tevens betrokken geweest bij het schrijven van het werkboek en het opstellen van de eindopdracht.

[1]
Stephen T. Albin. The Art of Software Architecture: Design Methods and Techniques. Indianapolis: John Wiley & Sons, 2003.
[2]
P. Clements, R. Kazman, and M. Klein. Evaluating Software Architectures: Methods and Case Studies. Addison-Wesley, 2002.
[3]
Paul Clements, Felix Bachmann, Len Bass, David Garlan, James Ivers, Reed Little, Robert Nord, and Judith Stafford. Documenting Software Architectures: Views and Beyond. Addison-Wesley, 2003.
[4]
Remi-Armand Collaris and Eef Dekker. RUP op Maat. Sdu uitgevers, 2008.
[5]
Humphrey Ferdinandus, Joop Kaldeway, Wolter Kaper, Aad Slootmaker, and Evert van de Vrie. Schakelen tussen bachelor- en masteropleidingen. Tinfon, (4):119-121, Oktober 2004.
[6]
Philippe Kruchten. Architectural blueprints: the 4+1 view model of architecture. IEEE Software, 12(6):42-50, 1995.
[7]
Nick Rozanski and Eóin Woods. Software Systems Architecture: Working With Stakeholders Using Viewpoints and Perspectives. Addison-Wesley Professional, April 2005.
[8]
Software Engineering Research Center (SERC). Quint2, the Extended ISO Model of Software Quality. http://www.serc.nl/quint-book/.
[9]
Software Engineering Standards Committee of the IEEE Computer Society. IEEE Recommended Practice for Architectural Descriptions of Software-Intensive Systems, Std 1471-2000, 2000.
[10]
TISO: onderwijs en onderzoek met authentieke dossiers van automatiseringsprojecten, http://www.tisoweb.nl/.