Agile-ontwikkelmethoden auditen

Bij steeds meer organisaties waar software ontwikkeld wordt, gebruikt men software-ontwikkelmethoden van Agile, zoals Scrum, DSDM en Extreme Programming. Agile-methoden bieden vooral flexibiliteits- en snelheidsvoordelen boven watervalmethoden. Watervalmethoden hebben, vanuit hun decennialange historie, bij veel bedrijven een beheerst proces en een volwassen auditproces opgeleverd. Agile-ontwikkelmethoden zijn relatief jong en het Agile-gedachtegoed is er op het eerste gezicht juist niet een dat gericht is op beheersing, zodat een audit naar Agile-ontwikkeling niet eenvoudig op bekende auditobjecten kan leunen. In dit artikel beschrijven we aan welke zaken een auditor zich toch kan vasthouden.

MARCEL TRIJSSENAAR EN MERIJN VAN DER ZALM

Onder andere in het bank- en verzekeringswezen en in de publieke sector, waarin organisatiebreed aanzienlijke goederen- en/of geldstromen rondgaan, is het essentieel dat zowel de *going concern* als vernieuwing beheerst en toetsbaar verlopen. Dit is ook een eis van wet- en regelgeving (SoX, Solvency). De goederen- en/of geldstromen worden steeds afhankelijker van de software. Die software wordt steeds vaker met Agile-methoden ontwikkeld en daarom moet de IT-auditor ook in staat zijn om Agile-ontwikkelmethoden te auditen. Agile-methoden geven door de minder duidelijke beheersafspraken de IT-auditor een nieuw speelveld, dat extra uitdagingen kan opleveren. De uitdaging om Agile-methoden te auditen is meerledig: er wordt vanuit Agile-methoden geen normenkader meegeleverd waartegen te auditen, Agile richt zich qua gedachtegoed niet primair op in-control zijn, en er is in de meeste bedrijven nog geen bewezen best-practice ontwikkeld om Agile software-ontwikkeling te auditen.

In dit artikel, geschreven op basis van onze praktijkervaringen en met gebruikmaking van inzichten uit het afstudeerreferaat van Rodenburg en Vlaanderen [RODE09][1], beschrijven we hoe de flexibiliteit en snelheid van Agile en de beheersing van het ontwikkelproces in organisaties op gespannen voet met elkaar staan. We lichten de verschillen tussen de ontwikkelmethoden toe, werken uit waarom Agile tot uitdagingen voor de auditor leidt en geven de auditor vervolgens praktische handvatten om hiermee om te gaan. Waar we in dit artikel wat meer in detail gaan, gebruiken we als voorbeeld Scrum, een bekende Agile-methode.[2]

WATERVALMETHODEN EN AGILE: VERSCHILLENDE UITGANGSPUNTEN EN AANPAK

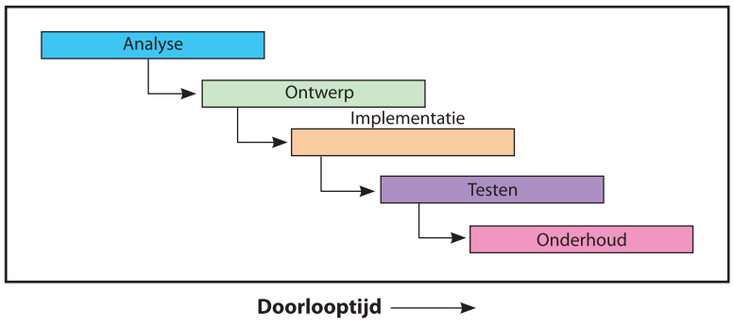
In deze paragraaf beschrijven we de verschillen tussen de softwareontwikkelingsmethoden die gebaseerd zijn op het watervalmodel respectievelijk het Agile-model.

Watervalmethoden

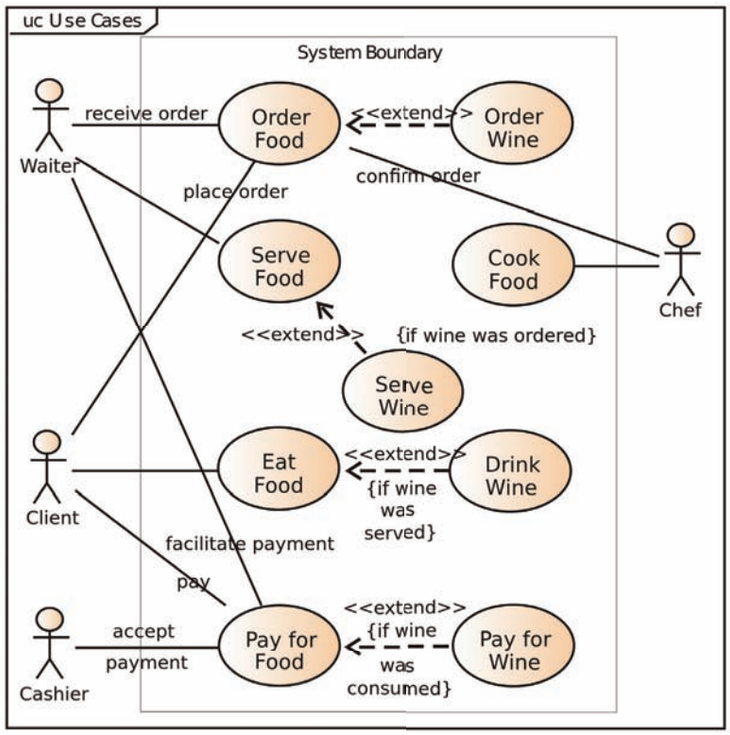
Watervalmethoden zijn beheerste methoden om nieuwe software te ontwikkelen. Er wordt stap voor stap gewerkt – van specificatie, via functioneel ontwerp, technisch ontwerp, bouw, test en acceptatie naar transitie en productie (zie figuur 1). Met een managementbril en vanuit auditoptiek bekeken, is elke fase op zichzelf staand, in die zin dat de fases helder zijn afgebakend en elke fase apart auditbaar is.

Ook het principe van eenrichtingsverkeer, stroomafwaarts, waarbij de resultaten van de voorgaande fase vastliggen en als input dienen voor de volgende fase, maakt het ontwikkelproces beheersbaar en auditbaar. Een verandering in de output van een fase ten opzichte van wat werd beoogd, wordt als *change* behandeld, wat bijdraagt aan de beheersbaarheid van het ontwikkelproces. Rodenburg en Vlaanderen typeren het watervalmodel als een solide, goed begrepen

IT Auditor_13_03.indd 10 03/09/13 5:04 PM



Figuur 1: Het principe van de waterval



Figuur 2: Voorbeeld use case [WIKI13]

model dat hoort bij een stabiel, herhaalbaar project met lage complexiteit. De software-eisen zijn stabiel en gefixeerd en het ontwikkeltraject is precies te voorspellen [RODE09].
In theorie is het voordeel van watervalmethoden bij deze omstandigheden dat het een efficiënt proces mogelijk maakt met weinig *rework* in latere fasen, omdat de specificaties vooraf volledig en correct vastgesteld zijn. In de praktijk blijkt het echter vaak lastig te zijn om vooraf de functionaliteit volledig te specificeren. Het gevolg is dat toch aanpassingen in latere fasen nodig zijn, waardoor het ontwikkelproces slechter te beheersen is, of de software minder goed aansluit bij de onderliggende klantbehoefte.

Ook heeft een watervalmethode een lange doorlooptijd van ontwerp naar productie. *Requirements* veranderen in de loop van de tijd, door veranderde klant- of gebruikerswensen, of doordat systemen waarmee de nieuwe software afhankelijkheden heeft worden gewijzigd.
In watervalmethoden wordt dit meestal opgelost door de requirements op enig moment te 'bevriezen' tot het live-moment. De requirements en daarmee de specificaties van een systeem zijn tegenwoordig aan veel snellere wijzigingen onderhevig dan voorheen en veranderingen worden meer de norm in plaats van de uitzondering. De voorwaarde van stabiele specificaties die de watervalmethoden veronderstellen, is hierdoor vandaag de dag zelden vervuld. De spanning tussen ontwikkeltijd en gewenste *time to market* wordt daarmee steeds groter. Dit maakt watervalmethoden steeds minder geschikt als ontwikkelmethode.

Agile-methoden

Agile software-ontwikkelmethoden zijn ruim een decennium geleden geïntroduceerd als opvolgers van de waterval-softwareontwikkelmethoden. Waar watervalmethoden uitgaan van stabiele requirements, is Agile juist bedoeld om beheerst om te gaan met veranderende requirements.
Volgens het Agile Manifesto uit 2001, richt Agile softwareontwikkeling zich op omgaan met verandering (laatste bullet) – vrij naar [BECK01]:

- *Personen en interacties* zijn belangrijker dan *processen en tools*.
- *Werkende en bruikbare software* is belangrijker dan *lijvige documentatie*.
- *Samenwerking met de klant* levert meer op dan *contractonderhandelingen*.
- *Omgaan met verandering* is belangrijker dan een *vooraf bepaald plan strak volgen*.

Een bekende methode gebaseerd op het Agile Manifesto is Scrum. Bij Scrum overleggen ontwikkelaars en gebruikers kortcyclisch, meestal eens per drie weken, over de requirements en welke functionaliteit de komende 'sprint' te implementeren. Elke sprint levert een nieuwe versie van de software. Deze software bevat soms voor de eindgebruiker extra functiona-

IT Auditor_13_03.indd 11 03/09/13 5:04 PM

liteit ten opzichte van de voorgaande versie, en soms is de nieuwe versie beter qua snelheid, stabiliteit, onderhoud of andere zaken die de eindgebruiker meestal niet direct ziet, maar die wel van belang zijn bij het gebruik van de software. Een belangrijk voordeel van het kortcyclisch opleveren van nieuwe functionaliteit is dat de gebruiker zich sneller een beeld kan vormen wat de nieuwe software precies doet, en of dat inderdaad is wat hij nodig heeft. Eventuele afwijkingen ten opzichte van de requirements komen al na enkele weken aan het licht, en niet pas na meerdere kwartalen zoals in het geval van watervalmethoden. In dit vroege stadium zijn de investeringen in de nieuwe functionaliteit nog beperkt. En, aanvullend, (geplande) wijzigingen in andere software die weer voortbouwt op de software die ontwikkeld wordt, zijn ook nog beperkt.

Een laatste voordeel is dat Agile tussenversies van de software oplevert die bruikbaar kunnen zijn voor productie. Ook de functionaliteit van een tussenversie die nog niet de volledige functionaliteit heeft, kan wel al voldoende zijn voor productie inzet. Hiermee kunnen producten of diensten die met de inzet van deze software geleverd worden al geïntroduceerd worden.
Deze verkorting van de time to market van een nieuw product of nieuwe dienst wordt voor veel organisaties steeds belangrijker.

HOE EN WAAROM LEIDT AGILE TOT EEN ANDERE AUDITAANPAK?

Wij verwachten dat systemen meer en meer bedrijfskritisch worden en dat systeemontwikkelingsprojecten een steeds belangrijker deel van de kosten van een organisatie zullen vormen en dat beheersing van die projecten daarom belangrijker aan het worden is dan ooit tevoren. Met Rodenburg en Vlaanderen [RODE09] verwachten wij dan ook dat het verschaffen van zekerheid aan het management over de beheersing van ontwikkelprojecten voor IT-auditors een steeds belangrijker werkveld zal worden.

Om meer zekerheid over deze beheersing te verkrijgen, kunnen opdrachtgevers audits inzetten. Zoals Rodenburg en Vlaanderen aangeven, voert de IT-auditor de onafhankelijke attestfunctie uit en komt tot een oordeel om zekerheid te verschaffen of een systeemontwikkelingsproject voldoende wordt beheerst door effectieve maatregelen. De IT-audit van een klassieke ontwikkelmethode is gebaseerd op de producten die gedurende het ontwikkelproces worden opgeleverd. Bij een audit worden voornamelijk de resultaten en tussenproducten van een proces getoetst en niet zozeer de totstandkoming, dus het procesverloop zelf. Bij een traditionele ontwikkelmethode zijn de mijlpaalproducten, rapportages en besluitvorming het object van onderzoek [RODE09].

De meest bekende richtlijn voor de audit van systeemontwikkelingsprojecten, de ISACA-richtlijn voor *software development life cycle* (SDLC) reviews [ISAC03], is volgens Rodenburg en Vlaanderen gebaseerd op deze traditionele systeemontwikkeling. Aan de hand van de verschillen tussen Agile en traditionele methoden op het gebied van randvoorwaarden en *IT-general controls*, geplaatst in een *management control framework*, komen de auteurs tot de conclusie dat de ISACA-richtlijn niet goed toepasbaar is voor agile-methoden [RODE09]. De belangrijkste belemmering is dat bij Agile-methoden een productgericht onderzoek slechts beperkt mogelijk is. In Agile-trajecten zijn veel traditionele producten, zoals requirements, functioneel ontwerp en wijzigingenbeheer op basis van excepties immers niet meer aanwezig. Agile steunt niet op vastlegging, maar op menselijke factoren, feedback en ondersteunende tools.

HOE AGILE SOFTWARE DEVELOPMENT TE AUDITEN?

Belangrijk voor de auditor is van oudsher de relatie tussen een processtap en het eindresultaat. Deze relatie is binnen een Agile-ontwikkelproject minder duidelijk. Als de auditor op een traditionele manier de audit uitvoert, dan vraagt hij om documentatie. Omdat het Agile-proces niet zozeer steunt op documentatie als wel op communicatie, zal de auditor zijn controleaanpak anders moeten invullen. Hij zal daarbij in grotere mate steunen op 'soft controls' in plaats van de meer hardere controls, zoals het referentiemodel voor de SLDC die biedt. Wij werken de handvatten die wij in de praktijk tegenkomen hieronder uit.

Andere documentatie auditen

Agile vraagt in principe niet om formele documentatie. Waar er bij watervalmethoden-documentatie sprake is van een specificatie, functioneel ontwerp en technisch ontwerp, is dat er bij Agile niet. Toch worden specifieke manieren ingezet om concrete casuïstiek met de gebruikers door te spreken en in een schema af te beelden, de zogenaamde *use cases*. In figuur 2 is een voorbeeld weergegeven. De use case in dit voorbeeld beschrijft vier rollen (*waiter, client, cashier* en *chef-kok*) en hun activiteit-informatiestroom-relaties van een gestileerd consumptieproces in een restaurant.

Agile registreert hiermee de voor het ontwikkelproces relevante informatie met eenvoudige hulpmiddelen die de gebruiker met begeleidende toelichting direct doorgrondt. Een use case laat zien hoe het systeem zich gedraagt op een verzoek van buiten het systeem. Use cases zijn gebruiksvriendelijk, snel en goedkoop. Tegelijk met het bouwen van de software, en het verder verfijnen van de specificatie wordt de use case bijgesteld. Deze vorm is in het algemeen minder uitgebreid dan een ontwerp. Er wordt bijvoorbeeld vanuit systeemarchitectuur geen rekening

IT Auditor_13_03.indd 12 03/09/13 5:04 PM

gehouden met toekomstige eisen. Wel, of juist daarom, is de use case, die alleen de concrete beschrijving van de beoogde systeemwerking beschrijft, in het algemeen zeer toegankelijk voor anderen dan de opsteller ervan, dus ook voor de eindgebruiker. De inhoudelijke toetsing kan dus aan de hand van use cases plaatsvinden. Aandachtspunten voor de audit zijn:

- aanwezigheid van de use-cases;
- accordering van de use-cases door de gebruiker of opdrachtgever;
- de vraag of het format van de use-cases voldoet aan bepaalde minimale eisen zoals modelleringseisen of de aanwezigheid van alle actoren uit AO/IC beschrijvingen; de eisen aan het format moeten dan door de organisatie in overleg met de auditdienst worden vastgesteld.

Scrum stelt voor dat voorafgaand aan elke sprint vastgesteld wordt welke functionaliteiten gebouwd gaan worden. Dit wordt opgenomen in de sprintplanning. De actuele voortgang versus verwachte voortgang wordt op dagbasis gemeten en vastgelegd in de sprint *burn down chart*. De sprintplanningen en burn down charts zijn auditbaar en geven informatie over welke functionaliteit wordt opgeleverd en hoe de realisatie ten opzichte van de planning verloopt.

Audit van proces en randvoorwaarden

Scrum definieert een helder omschreven werkproces en veronderstelt een aantal randvoorwaarden voor succes. Deze twee zaken zijn in onze ogen goed auditbaar.

Het werkproces

Scrum beschrijft gedetailleerd welke producten opgeleverd worden. Dit zijn bijvoorbeeld de *sprint backlogs* en use cases. En Scrum beschrijft welke communicatie minimaal plaatsvindt. De doorlooptijd van een sprint wordt vooraf vastgesteld. Deze doorlooptijd vormt een *timebox*, wat inhoudt dat uitloop in de tijd niet is toegestaan. Een sprint wordt gestart met een planningssessie, de *sprintmeeting*, waarin het ontwikkelteam en de business gezamenlijk vaststellen welke functionaliteit in de sprint gebouwd gaat worden. Tijdens de sprint wordt dagelijks de voortgang gemeten in de sprint-burndown chart. Aan het einde van de sprint is er een *retrospective*, waarin het proces en de geboekte voortgang op de inhoud binnen de sprint geëvalueerd worden, als input voor de komende sprintplanning. Het is belangrijk voor een beheerst Scrumproces dat dit proces gevolgd en waar nodig bijgestuurd wordt. Dit is goed auditbaar.

Randvoorwaarden

De belangrijkste randvoorwaarden die de IT-auditor moet toetsen zijn:

- een vast team met ervaren softwareontwikkelaars;
- een vast aanspreekpunt uit de business met voldoende kennis en ervaring om de gewenste toekomst te kunnen verwoorden en met voldoende senioriteit en de juiste positie in de gebruikersorganisatie om de business-wensen te prioriteren, de zogenaamde 'product owner';
- goede communicatie tussen ontwikkelaars en business-vertegenwoordigers.

Het is aan te bevelen om van sprint naar sprint te auditen of het proces wordt gevolgd en of voortdurend de randvoorwaarden aanwezig zijn.

Projectbesturing op de Scrum-softwareontwikkeling en op het project

Systeemontwikkeling heeft een plaats binnen een meeromvattend project. De Scrum-besturing wordt daarom overkoepeld door de besturing vanuit de projectstuurgroep. Projectmatig werken en de beheersmechanismen daarvoor zijn in veel organisaties geïnstitutionaliseerd en in de praktijk beproefd. De software kan naar behoefte de enige project deliverable zijn, dan wel gecombineerd worden met andere project deliverables zoals een business-implementatie of gebruikersopleiding.

De projectbesturing heeft bij Agile echter andere accenten dan bij watervalmethoden. Bij watervalmethoden wordt de functionaliteit vooraf gespecificeerd en vastgelegd. Nu deze projectstuurparameter vaststaat, stuurt het projectmanagementteam vervolgens vooral op tijd en geld. Bij Scrum zijn het juist tijd en geld die zich goed lenen om vooraf vast te zetten. De tijd is door het werken met timeboxen goed voorspelbaar en wordt geoperationaliseerd door het aantal ontwikkelsprints, de transitieperiode en de tijd die nodig is voor inproductiename vooraf vast te leggen. De kosten in termen van mensuren zijn ook goed voorspelbaar, die zijn het product van doorlooptijd en totale projectteamgrootte. Bij Scrum is het effectief om gedurende het project uit te gaan van een vast, op elkaar ingewerkt, softwareontwikkelteam aangevuld met business-vertegenwoordigers. De projectstuurparameters bij Scrum zijn de te leveren functionaliteit en kwaliteit. Elke sprint wordt de nieuw te leveren functionaliteit bepaald in de sprintmeeting.

Een niveau hoger wordt het ontwikkelteam, inclusief de business-vertegenwoordiger, gestuurd door de Stuurgroep. Wat bij Scrumprojecten opvalt, is de toename van de dynamiek van de business case. In de meeste (niet-Scrum) projecten worden deze voorafgaand aan een project opgesteld en hooguit enkele keren bijgesteld en na afronding gebruikt voor evaluatie. Bij een Scrum-ontwikkelproces is dat per definitie veel vaker. Er is daarbij een niveauverschil tussen de kosten, baten en risico's van een specifieke softwarefunctionaliteit (change) op basis van gebruikersfeedback en de kosten, baten en risico's van samenhangende groepjes changes ('business functionaliteiten'), die op stuurgroepniveau beter als project- of procesveranderingen herkenbaar zijn. ◘

IT Auditor_13_03.indd 13 03/09/13 5:04 PM

Omdat door het afvallen of herprioriteren van changes bij elke sprintmeeting op basis van veranderde gebruikerswensen de situatie verandert, moet de stuurgroep steeds opnieuw prioriteiten aangeven op basis van een bijgestelde project business case per samenhangend groepje changes. Het is dan noodzakelijk dat de Stuurgroep steeds goed inzicht heeft in de voortgang en de tot dan toe geleverde functionaliteit – en vervolgens sturing geeft aan het project, zodat de business case positief blijft. Op basis van stuurgroepverslagen en de projectdocumentatie kan een auditor vervolgens zien of de besturing werkt.

We merken dat dit voor stuurgroepen een behoorlijke omslag is, waarvan we het belang niet moeten onderschatten. In plaats van hun traditionele focus op uitzonderingsmaatregelen bij uitloop in tijd en/of geld moeten ze zich nu richten op de prioritering van inhoudelijke functionaliteiten. Dit kost ze vaak in eerste instantie meer tijd, maar versnelt wel de plan-do-check-act cyclus. De IT-auditor moet er alert op zijn of een stuurgroep feitelijk werkt op de manier de Scrum van ze verwacht. Zie tabel 1.

Parameters	Focus van de stuurgroep	
Sturingsaspect	*Waterval*	*Agile (Scrum)*
Functionaliteit	x	**X**
Kosten	**X**	x
Doorlooptijd	**X**	x
Kwaliteit	x	**X**

Tabel 1: Verschillen in accenten voor project-stuurparameters

Aandachtspunten voor de audit zijn:
- of de lengte van de sprints, de projectplanning en frequentie van stuurgroepoverleggen op elkaar afgestemd zijn;
- of de stuurgroep inzicht heeft in de per sprint geleverde functionaliteit;
- of niet alleen de business gebruiker de jure in de stuurgroep vertegenwoordigd is, maar dat hij ook de facto aanwezig is geweest;
- of er besluiten zijn genomen over functionaliteit op basis van prioritering;
- of er communicatie is geweest over gewijzigde prioritering vanuit de stuurgroep naar het scrum-team.

CONCLUSIE

Wij merken in de praktijk dat het werken volgens Agile-principes opdrachtgevers in organisaties veel flexibiliteit en snelheid geeft, maar dat onzekerheid over de beheersing van de veranderingen deze opdrachtgevers ook zorgen baart. Met een auditwerkwijze die aansluit op de controleerbare objecten die in Agile-methoden worden gerealiseerd, zoals Agile-ontwikkeldocumentatie (use cases en burndown charts), de werking van het Agile-proces en de bijbehorende randvoorwaarden en andere project-stuurparameters, kan Agile-softwareontwikkeling goed geaudit worden. Het gebruik van deze handvatten geeft opdrachtgevers voldoende zekerheid om Agile-softwareontwikkeling met de tijdige betrokkenheid van een auditor met een gerust hart in te zetten. ■

Drs. Merijn van der Zalm is programmamanager en adviseur bij Verdonck, Klooster & Associates (VKA). Hij publiceert regelmatig over de toepassing en beheersing van ICT vanuit een bedrijfskundig perspectief: merijn.vanderzalm@vka.nl.

Dr. ir. Marcel Trijssenaar is senior projectmanager bij Zorgverzekeraar Coöperatie VGZ met meerdere jaren ervaring in het aansturen van Scrum-projecten: m.trijssenaar@vgz.nl.

Literatuur

[BECK01] Beck, Kent, et al., *Manifesto for Agile Software Development*, februari 2001, http://agilemanifesto.org/.

[ISAC03] ISACA, *IS auditing Guideline System development life Cycle (SDLC) reviews*, IT Audit and Assurance Guidelines, document G23, ISACA, 2003.

[RODE09] Rodenburg, Jan en Vincent Vlaanderen, *Rol van de IT auditor bij agile systeemontwikkeling*, Referaat – Postgraduate IT-audit opleiding, VU Amsterdam, mei 2009, http://www.vurore.nl/images/vurore/downloads/911_rol_IT_auditor_bij_agile_systeemontwikkeling_Rodenburg_Vlaanderen.pdf.

[WIKI13] http://en.wikipedia.org/wiki/File:Use_case_restaurant_model.svg

Noten

1 Het afstudeerreferaat van Rodenburg en Vlaanderen biedt een zeer interessante verdieping naar de aanzet tot een theoretisch onderbouwd referentiemodel/toetskader.

2 Oorzaak falen Agile-projecten in 'mensen en cultuur'. In: Automatisering Gids, 15 maart 2013.

IT Auditor_13_03.indd 14 03/09/13 5:04 PM