Bij grote projecten worden regelmatig externe partijen ingeschakeld bij het reviewen van code. Quinity is een bedrijf dat regelmatig J2EE-projecten reviewt. Uit onderstaand gesprek blijkt dat daarbij soms interessante inzichten te winnen zijn. Quinity kwam enige tijd geleden bovendien in het nieuws doordat het bedrijf graag net afgestudeerde informatici in dienst neemt. Een gesprek met Jan van Reenen en Patrick van Driel, respectievelijk technisch specialist en projectleider bij Quinity.

Jan van Reenen en Patrick van Driel:

‘J2EE volwassen, maar IT nog niet’

Core business van Quinity is het ontwerpen en bouwen van administratieve systemen met Internettechnologie. Alles is browser-based: server side processing, met soms wat Java-script in de browser. Het zwaartepunt ligt dus heel sterk op Java en in het bijzonder J2EE.

Patrick van Driel: ‘Eigenlijk heeft de markt ons daartoe bewogen. Toen wij begonnen waren er eigenlijk twee stromen, J2EE en Microsoft. Wij werken voor financiële instellingen en die vragen vrijwel allemaal om J2EE, ik ken geen Internetbankieren-oplossing die niet met J2EE werkt.’

Jan van Reenen: ‘Het is sowieso wel logisch, want je ziet dat Internetapplicaties en enterprise applicaties veel overeenkomstige eigenschappen bezitten: beide hebben veel gebruikers, hoge beschikbaarheidseisen, hoge schaalbaarheidseisen, flexibiliteit, integreerbaar met andere systemen. Ze moeten goed beveiligbaar zijn, ze moeten goed performen; allemaal eisen die overeenkomen. Verder integreren we wel met Microsoft: sommige klanten willen op Windows inloggen en dan automatisch in een J2EE-systeem ingelogd zijn. Tenslotte worden webservices veel gebruikt, om technologiegrenzen te overschrijden, met name bij ketenintegratie, bij verzekeraars.’

Mooi concept *Quinity kwam vorig jaar in het nieuws, doordat het bedrijf bekend maakte - in tegenstelling tot andere IT-bedrijven - op dat moment wel degelijk geïnteresseerd te zijn in net afgestudeerde informatici.*

Van Driel: ‘We geven ook gastcolleges op de universiteiten. We willen de studenten in de exacte studies - voor zover er een IT-component in zit - heel graag hebben. Wij hoeven niet per se iemand die al vijf jaar J2EE-ervaring heeft: sterker nog; soms is het moeilijker iemand iets af te leren dan hem het goed aan te leren.’

Dat betekent ook dat er nogal wat mensen zijn die al jarenlang dingen doen waar Quinity niet zo blij mee zal zijn.

Van Driel: ‘Dat hoeft niet specificiek J2EE te zijn, maar hoe je een klantprobleem vertaalt naar een oplossing binnen de afspraken die gemaakt zijn. En dat is precies wat wij de afgelopen vijf jaar met Java-technologie gedaan hebben. We kunnen dat doen op een voorspelbare manier, omdat we een SDE (*solution delivery environment, red.*) hebben. Daar zit een stuk technologie in, een framework van Java classes, superklassen waar je functionaliteit tegenaan kunt bouwen, procedures, ontwikkelstraat-ideeën. Het is een mooi concept dat bijna nergens in de praktijk goed is uitgerold, maar bij ons wel. Als je iemand hebt die vijf jaar ervaring heeft omdat hij bij één van de grotere IT-bedrijven heeft gewerkt of bij een bank, dan heeft die persoon zijn eigen manier van werken. Die was daar als het ware de

> ‘Het is zinvol als je het goed kunt gebruiken, maar ook gevaarlijk omdat je het gemakkelijk kunt misbruiken’

Core business van Quinity is het ontwerpen en bouwen van administratieve systemen met Internettechnologie.

goeroe en die heeft er moeite mee dat hij bij Quinity nu in zo'n standaard-ontwikkelomgeving moet werken. Slimme, net afgestudeerde studenten leren heel veel. Het is op een positieve manier helemaal voorgekookt voor ze. Ze hoeven over een heleboel dingen niet meer na te denken, en kunnen zich gelijk richten op functionaliteit, en technologie implementeren.

Natuurlijk zijn er wel mensen bezig met af en toe iets technisch uit te zoeken. Kunnen ze het niet oplossen met bestaande technologie uit onze ontwikkelstraat, dan gaat iemand het uitzoeken, proof of concept doen, en daarna kan het in alle projecten weer op dezelfde manier worden ingezet.'

> 'De zaken die uitgezocht moeten worden, zitten vaak in de periferie, zoals interfaces met andere systemen'

FUNCTIONALITEIT Van Driel: 'De zaken die uitgezocht moeten worden, zitten vaak in de periferie, denk bijvoorbeeld aan interfaces met andere systemen. De kern van de administratieve toepassingen bestaat vaak uit dezelfde patronen: je gaat altijd zoeken, selecteren, muteren of invoeren. Dat hebben we helemaal in een framework gegoten zodat je de basis al kunt genereren. Vanuit een datamodel kunnen we Java-classes genereren en daarna ga je het uitbreiden met klant- of projectspecifieke logica. Tachtig procent bestaat uit technologie waar wij geen risico's in zien.'

De universiteitsverlaters krijgen niet alleen een technische scholing:

Van Driel: 'We hebben een PDM, een personal development manager in dienst, en die houdt zich bezig met carrièrebegeleiding op niet-technisch gebied. Wijzelf kunnen ze bijbrengen hoe het framework in elkaar zit, hoe je datamodellen maakt en hoe normaliseren werkt, maar ze moeten ook een presentatie kunnen geven. En dan niet zozeer een presentatie voor honderd man, maar de presentatie tijdens een meeting met een klant, bijvoorbeeld over het verloop van een project.'

Van Reenen: 'Het hangt ook direct samen met onze focus op functionaliteit. Natuurlijk, techniek is belangrijk, maar misschien dat het over vijf jaar helemaal niet meer belangrijk is, omdat iemand bedacht heeft dat je het kunt genereren uit een of ander generatietaaltje, of dat het in India gebeurt. Dan heb je hier alleen nog maar functioneel ontwerpers en analisten nodig. Ik chargeer het misschien een beetje, maar je kunt je niet alleen op Java richten. Dan kun je geen carrière maken binnen het team.'

CODEREVIEWS *Quinity doet ook regelmatig code-reviews van grote projecten, onder meer voor grote Nederlandse banken en een grote webwinkel.*

Van Reenen: 'In grote projecten worden zaken uitbesteed of soms zelfs geoffshored, waarbij toch gewaarborgd zal moeten worden dat de kwaliteit van de zaken die geïmplementeerd moeten worden een bepaald minimumniveau heeft. Ze willen dat toetsen, en dat kan niet altijd intern. Die eerste financiële instelling ontwikkelde zelf, waarbij wij het gereviewed hebben. Bij de webwinkel is het in Nederland uitbesteed en hebben wij de codereview gedaan (en ook een project audit overigens). Bij de huidige financiële instelling wordt het uitbesteed en voor een deel geoffshored, en doen wij de review. Dus eigenlijk komen alle varianten voor, maar wat ze gemeen hebben is dat ze zoeken naar kwaliteit.'

Van Driel: 'Bij deze laatste instelling zou je je kunnen afvragen waarom ze dan niet de bouw aan ons uitbesteden. Maar daar vonden ze ons nou net weer te klein voor. Het gaat om een heel groot project dat aan een heel grote partij uitbesteed is. Daarbij waren wij zeer

geschikt om de code die terugkomt van de leverancier te toetsen aan de criteria die ze van tevoren hebben meegekregen. Reviewen gebeurt grofweg om twee redenen, of om combinaties daarvan. Preventief, ze krijgen het terug en willen het laten toetsen voordat ze het überhaupt gaan testen, en in productie nemen om te zien of het voldoet aan de criteria. De andere reden is reactief: het komt ook voor dat we benaderd worden bij een project wat al loopt, maar niet zo lékker loopt.'

Van Driel: 'Dan ga je vaak een review doen in een context die al wat gespannen is. Zo'n leverancier, dus niet de opdrachtgever, die zit er natuurlijk niet op te wachten dat wij eventjes komen kijken in hun keuken. Ze zijn namelijk druk bezig het project nog op de rails te houden.'

Is het niet vreselijk tijdintensief om die code allemaal te lezen?

Van Driel: 'Ja, maar we doen het wel. Het is niet zo dat het drie uurtjes werk is. Je moet er wat tijd voor reserveren, maar we kunnen ook met steekproeven werken.'

Van Reenen: 'Beide modellen worden gebruikt. We kijken onafhankelijk van het model altijd welke code het belangrijkste is voor het systeem. Je begint bij de centrale code, framework-code en dat soort zaken, en vervolgens daal je af. Voor steekproefsgewijs reviewen heb je dan de belangrijkste, meest centrale klassen wel gehad, en voor integraal reviewen begin je in ieder geval voor je gevoel bij het begin. Als dingen voortbouwen op andere zaken, heb je in ieder geval de basis eerst bekeken. Dat hebben ze gemeenschappelijk, alleen neem je bij het steekproefmodel de belangrijkste klassen, en bij de overige classes ga je letterlijk steekproeven nemen. Dan accepteer je bij voorbaat dat je niet alles hebt gezien, dus dat er gekke dingetjes door kunnen glippen. Bij het integrale reviewmodel ga je in ieder geval van tevoren afspreken waar je op gaat letten. Vaak worden er heel specifieke dingen afgesproken: coding guidelines, structuurzaken, onderhoudbaarheid, documentatie. Voor datgene wat niet expliciet wordt afgesproken, hanteren we best practices. Soms vraagt de klant ook een indicatie, of we problemen verwachten als we het gaan implementeren. Een garantie kunnen we net zo min als de bouwer geven, maar we kunnen wel kijken of ze gekke dingen hebben. Heb je iemand die uiterst creatief bedacht heeft dat je met een session bean ook iets heel anders kunt doen, dan wat je in gedachten had, tja, dan wordt het een stuk discutabeler of het allemaal op de goede manier gaat werken. Die indicatie kunnen we dus ook aangeven, maar dat is meer werk. Je zult dan veel meer analyses moeten doen op het gebied van de samenhang tussen de code en niet meer de classes en de code afzonderlijk bekijken op hun kwaliteit, afgezet tegen de richtlijnen die de bouwer heeft gekregen. Wat ook wel eens voorkomt, is dat de bouwer richtlijnen krijgt, maar dat de klant vraagt om de richtlijnen te reviewen. Dus dan kijk je niet naar het eindresultaat maar naar de vraag die gesteld wordt.'

Van Driel: 'Dat betreft dus het reviewen van statische zaken, documentatie en code, maar we reviewen ook wel runtime gedrag, dynamische zaken. Dingen worden opgeleverd waarbij we al aan de code zien dat het niet efficiënt geprogrammeerd is, maar dan moet je het ook nog eens een keertje meten. Dus dat je niet de functionaliteit test, maar het geheugengebruik, of de cpu-belasting. Dat is ook heel specialistisch werk en daar wordt vaak door ontwikkelaars niet aan gedacht, die zijn blij als het gewoon werkt, maar of het ook performt als er twee gebruikers tegelijk zijn...'

ALGORITME *Quinity maakt wel gebruik van tools om code te checken, maar ziet er ook de beperkingen van in.*

Van Reenen: 'Je kunt met tools heel veel fouten eruit halen waar je gewoon gemakkelijk overheen leest, maar je haalt er niet uit of je een slimme of een slechte structuur hebt.'

Patrick van Driel: 'We hebben het helemaal in een framework gegoten zodat je de basis al kunt genereren'

Jan van Reenen: 'Voor datgene wat niet expliciet wordt afgesproken, hanteren we best practices'

Van Driel: 'Het is meer een syntactische controle, terwijl je niet kunt zien of bijvoorbeeld het algoritme goed geïmplementeerd is.'

Neemt het reviewen en het gebruiken van externe reviewers toe?

Van Reenen: 'Wat in zijn algemeenheid toeneemt is het implementeren van kwaliteitsmaatregelen. Daar wordt in verschillende richtingen gezocht: sommige organisaties zoeken dat in CMM, andere organisaties in testen en weer andere in reviewen of in combinaties van de eerder genoemde.'

'Natuurlijk, techniek is belangrijk, maar misschien dat het over vijf jaar helemaal niet meer belangrijk is'

Van Driel: 'Het zou ook te maken kunnen hebben met het feit dat ook meer gebruik gemaakt wordt van J2EE, wat voor veel organisaties nieuwe technologie is. Er gaat ook veel mis in J2EE.'

Van Reenen: 'Dat is niet specifiek voor banken, je ziet het over de hele linie. Bij de eerste generatie J2EE-applicaties is de techniek slecht begrepen en dus ook slecht geïmplementeerd.'

SESSION BEAN *Bij navraag blijkt dat Quinity vaak ziet dat fouten die jaren eerder bij CICS gemaakt werden, in J2EE terugkomen.*

Van Reenen: 'Ik denk dat de grootste problemen in de sfeer van de EJB's hebben gezeten. Maar ook session beans: er zijn organisaties die hebben één session bean, die verantwoordelijk is voor zo'n beetje alle functionaliteit. Net zo goed als dat de geschiedenis zich herhaalt voor wat betreft de technologie, gebeurt dat ook bij de fouten. Vroeger werden vaak CICS-transacties gemaakt met verschillende parameters die naar verschillende subsystemen verwezen. Op een gegeven moment had één van de achterliggende subsystemen een probleem en het enige wat je zag: de transactie doet het slecht. Je had geen idee waardoor het kwam, want afhankelijk van de parameters die je meegaf deed de applicatie totaal iets anders. Tegenwoordig weet iedereen dat je dat allemaal moet uitsplitsen: alle transacties worden in losse modules gebouwd. Zo is J2EE ook bedoeld, want het zijn natuurlijk ook voor een deel dezelfde mensen die aan die technologie hebben gewerkt, bij IBM en Sun. Je moet allemaal transacties definiëren die je kunt implementeren met session beans. Soms kun je het recht toe recht aan implementeren, maar als je ze toepast, dan zijn je session beans gewoon je transacties, zoals je ook een CICS-transactie had. Ook dat is verkeerd gegaan. Veel mensen hebben in feite één session bean gemaakt voor alle transacties. Dus als je achterliggende systeem een probleem had, had je session bean een probleem. Maar ja, je had geen idee waardoor het kwam.

Dat is een heel herkenbaar voorbeeld, maar er zijn ook andere: bij het MVC-concept komt een request binnen op een servlet, het ding gaat wat doen en het resultaat wordt weergegeven met behulp van een JSP. De developer had begrepen, dat hij servlets en JSP's moest gebruiken en wat gebeurde er: er kwam een verzoek binnen op de servlet, de servlet stuurde een redirect terug naar de browser en die redirect leidde ertoe dat de JSP rechtstreeks werd opgevraagd en dat vervolgens de logica werd uitgevoerd. Dat soort dingen zie je gebeuren in de praktijk: de technologie wordt nog niet goed genoeg begrepen om hem goed te kunnen toepassen. Dat is natuurlijk een gevaar wat sowieso op de loer ligt wanneer je nieuwe technologie wilt toepassen.'

Zijn er nu ook nieuwe technieken waar veel mensen op inspringen en die nu fout gaan?

Van Reenen: 'Moeilijke vraag, vooral gekoppeld aan het verkeerde gebruik. Wat natuurlijk een belangrijke nieuwe technologie is die door Sun wordt gepromoot, is Java server faces. Daarin zijn alle belangrijke zaken die in Struts geregeld zijn, meegenomen. Struts is ook bepaald misbruikt door de nodige developers. Het ligt dus in de lijn der verwachting dat dit bij JSF ook gaat optreden. Daar heb ik nog geen concrete (foute) voorbeelden van gezien, maar dat is in ieder geval een concrete technologie waarvan ik zeg: als wij het binnen Quinity gaan toepassen, dan zou het een kandidaat zijn om eerst een proof of concept te doen, om het te standaardiseren, goede richtlijnen te ontwikkelen en netjes op te nemen in de ontwikkelstraat. Iedereen gaat het dan op dezelfde manier doen, zodat je ook zeker weet dat het op de goede manier wordt toegepast. Want als je het niet voorziet van de juiste richtlijnen, dan is dat ook weer een kandidaat om veel verkeerd te doen. Een ander voorbeeld: messaging. Dat is natuurlijk geen nieuwe technologie maar een integraal onderdeel van transactiesystemen, maar wat je natuurlijk ziet, is dat de gemiddelde Java-developer niet uit de hoek van de transactiesystemen komt, en vanuit die achtergrond dus ook niet weet hoe je messaging moet positioneren en benutten in je systeem. Dat ligt dus niet zozeer op implementatie-niveau als wel op ontwerpniveau. Ik heb ook in de praktijk vaak gezien dat daar de nodige dingen verkeerd gaan. Dat zie je nu natuurlijk meer gebeuren naarmate messaging meer een standaard onderdeel van de applicatie wordt. Maar als je de achterliggende concepten en doelstellingen niet kent, ligt er ook weer een enorme valkuil, als je het verkeerd toepast. Dus dat is zeker een technologie waarmee je moet oppassen. Het is heel zinvol als je het goed weet te gebruiken, maar ook heel gevaarlijk omdat je het gemakkelijk kunt misbruiken.'

> 'We spreken vaak heel specifieke dingen af: coding guidelines, onderhoudbaarheid, documentatie'

Jan van Reenen (r) en Patrick van Driel, respectievelijk technisch specialist en projectleider bij Quinity.

Belofte *Aan de andere kant: zijn er ook ontwikkelingen die hoopgevend zijn?*

Van Reenen: 'Ik denk wel dat nu J2EE een paar jaar oud is, de technologie wel een succes is. Dat wat beloofd werd, namelijk dat je er bedrijfskritische transactiesystemen mee kunt bouwen die zeer schaalbaar zijn en die je kunt draaien op een Windows systeem, op een Linux-systeem maar ook op een mainframe, dat die belofte wel degelijk wordt ingelost. Dat is wel mooi om dat nu te zien.'

Tekst en fotografie: Dré de Man