thema

Van 17 tot en met 19 november 2003 vond het door Array Publications georganiseerde congres Java-Systems plaats. Met het oog op latere publicatie had er een besloten ronde tafelgesprek met vier van de daar aanwezige sprekers plaats. Hieronder vindt u het verslag daarvan, waarin de uitingen van de sprekers met het oog op de samenhang vrijwel onverkort zijn weergegeven.

EJB-alternatieven en meer

Ronde tafelgesprek tijdens Java Systems

Het gesprek begon met een (zelf)introductie van de sprekers. Deze introduceerden zichzelf als:

'Floyd Marinescu, auteur van het boek EJB design-patterns en ook oprichter van de serverside.com, drie jaar geleden, de grootste Java site afgezien van sun.com en de grootste enterprise Java site, met allerlei dingen voor developers om ze op de hoogte te houden.'

'Olav Maassen, auteur van applied Java patterns (en van SRM en Java Magazine); J2EE developer en evangeliser.'

'Andy Longshaw, auteur van een aantal dingen waar ik hier niet op in ga, en ik doe enterprise architectuur en J2EE en .NET'.

'Marc Portier, zou me ertoe moeten zetten een boek te schrijven, maar ben ook geïnviteerd als discussie-leider. Professioneel ben ik mede-oprichter van Outerthought en van daaruit redelijk actief bij de Open Source groep Apache. Technisch werken we aan Java XML, maar we weten ook aardig wat van J2EE patterns.'

EVOLUTIE Portier: 'Laat ik beginnen met een aardige openingsvraag. Wij proberen J's zoveel mogelijk te vermijden. Denk daarbij vooral aan het beroemde citaat van Fowler, wat is de eerste regel van distributed computing: niet doen! Ik heb een aantal evoluties in EJB-land gezien, entity's hebben nu hun local homes en interfaces. Wijst dat er niet min of meer op dat we het uiteindelijk niet nodig hadden? Dat het alternatief - servlets met JDO - je hetzelfde soort flexibiliteit geeft?'

Marinescu:'Mijn favoriete onderwerp!'

Portier: 'Daarom wilde ik het als opening gebruiken.'

Marinescu: 'Ik denk niet dat het gebruik van lokale interfaces aangeeft dat EJB's niet handig zijn, ik denk dat het juist een verkeerd gebruik is van EJB's is om ze als lokale interfaces te gebruiken. Het is een poging van Sun om ze te laten gebruiken in use cases waar ze eigenlijk niet in gebruikt zouden moeten worden. EJB's zijn oorspronkelijk ontworpen om een gedistribueerde componenten specificatie te zijn, omdat er behoefte was aan zo'n soort technologie. In die tijd had je allerlei soorten aanpakken, je kon je eigen RMI-servers schrijven, Apple had Webobjects, er waren allerlei soorten oplossingen voor distributed computing. EJB is een geweldige oplossing voor het probleem: hoe communiceer je van de ene box naar de andere via het netwerk, en hoe doe je dat in een onderhoudbare en portable manier zonder je eigen server te moeten schrijven. Dat is het probleem dat EJB's oploste. Ik denk dat de marketing jongens een beetje op hol sloegen en probeerden de markt ervan te overtuigen dat je EJB's zou moeten gebruiken gewoon als een algemeen framework voor business logica. Ontwikkelaars geloofden hen, maar als je een groepje ontwikkelaars vraagt hoeveel van hen er EJB's gebruiken in een niet-gedistribueerde omgeving, dan blijkt iedereen dat te doen. Dat is omdat ze allemaal de hype geloofden, zelfs ik geloofde de hype. Iets om te weten is dat ze in de Microsoft-wereld enterprise services hebben. Die lijken op EJB, en je gebruikt ze in een gedistribueerde omgeving, anders gebruik je gewoon ASP.NET. Niemand is daar religieus over, maar anders heeft het gewoon geen zin. Ik vraag me af waarom er in de Java wereld eigenlijk niet zo'n soort aanpak is.'

Longshaw: 'Daar ben ik niet zo zeker van. In de Microsoft wereld gebruiken ze enterprise services voor declaratieve transacties en declaratieve security control. Als je het hebt over lokale EJB's: in B2B zie ik daar een groot voordeel, in het leveren van dat soort declaratieve omgevingen. Ik heb een aantal jaren geleden gediscussieerd met mensen die zeiden dat je EJB's, J2EE omgevingen en zo niet nodig hebt, omdat je alles wat ze doen ook zelf kunt doen, zoals transactie controle,

security controle. Tot op zekere hoogte ben ik het daar mee eens. Ja, je kunt het allemaal zelf doen, maar de doorsnee ontwikkelaar heeft niet genoeg kennis om dat succesvol te doen, dus zie ik liever dat ze locale EJB's daarvoor gebruiken.'

MONKEY PROGRAMMING Maassen: 'In het begin - toen EJB's geïntroduceerd werden - kan ik me een gesprek met Sun-mensen herinneren die ons vertelden: er zijn niet zoveel goede programmeurs zoals jullie, daarom hebben we EJB's. We noemden het toen monkey-programming.'

Longshaw: 'Aan de Microsoft-kant had je een paar jaar geleden hordes Visual Basic programmeurs aan wie Com+ werd gepresenteerd. Die reageerden met: 'wat is dat in 's hemelsnaam?'. Dus zei ik, oké, laten we het simplificeren, laten we iets maken dat eruit ziet als een Visual Basic-object. Je zet die flags en die switches en dan doet het transacties voor je. Daarna was de reactie, oké, dat doen we.'

Marinescu: 'Maar om terug te komen op het idee om EJB's lokaal te gebruiken: er is een nieuwe denkschool over het gebruiken van lichtgewicht containers. Het meeste gebeurt in de Open Source beweging: Apache Avalon, het Spring framework. Twee jaar geleden zouden discussies over EJB zinloos geweest zijn, maar nu - met al die nieuwe frameworks die declaratieve transacties doen - stoppen sommigen er zelfs aspectgeoriënteerd programmeren in. Het feit dat het EJB een geweldig framework is, werkt niet meer in zijn voordeel, omdat je al die andere geweldige frameworks hebt. Dus als je me vraagt wat de use cases zijn waarvoor alleen EJB een antwoord is, dan is dat een disitributed computing probleem.'

Longshaw: 'Het probleem is de ondersteuning. Je kunt dus eindelijk een framework als Spring gebruiken, maar van hoeveel vendors kun je het krijgen? Het is mooi dat het Open Source is, maar niet iedereen wil Open Source. EJB is een vendor-ondersteunde oplossing.'

JINI Portier: 'Dan zijn we alweer in een andere discussie. Maar het lijkt erop dat jullie een punt hebben vermeden, namelijk de vraag of we het werkelijk nodig hebben voor gedistribueerde computing? Webservices bewijzen dat er andere manieren zijn, en valt RMI niet terug op Corba en dus tight coupling?'

Maassen: 'Toen EJB's uitkwamen, kwam ook Jini. Dat was geweldige technologie en die kon een hoop van de problemen van gedistribueerde computing oplossen. Het kan erg lightweight zijn en dynamisch. Veel dingen die we nu proberen op te lossen met webservices herken ik van Jini. Maar het gebeurt nu op de verkeerde manier, als je er op de dynamische manier tegen aan kijkt. Webservices zullen geaccepteerd worden omdat Microsoft en IBM erachter staan. Technologisch is het jammer want Jini is zo elegant en je kan er zoveel dingen mee doen. Waarom gebruiken we EJB? Omdat het nu geaccepteerd is.'

Longshaw: 'Jini werd het meeste gebruikt in kleinere systemen. Je kocht iets van Sony en iets van Ericsson en je verbond het met elkaar. EJB's waren echter thuis in de datacentra. De vraag is daar: waarom distribueer je, waarom ga je je business logica verwijderen van je webframe dat de UI levert. Meestal bewaar je dat afzonderlijk, puur op grond van availability. In potentie zou je de databases kunnen laten draaien op dezelfde server als je applicatieserver, maar meestal scheid je het datacluster vanwege optimalisatie. Waarom zou je de business logica ertussenuit halen, tenzij je fat clients processing hebt?'

Maassen: 'Het punt bij Jini is: op kleine apparaten is het niet helemaal volledig. De inside story is dat de Jini groep een demonstratie gaf van Jini voor marketing mensen die niet al te technisch waren. Om het concept duidelijk te maken maakten ze de demo die nu over de wereld gaat, met kleine apparaten in een zeer dynamische omgeving. Die marketing mensen zeiden: 'ok!', en binnen een week werd Jini geassocieerd met kleine apparaten, want dat is een zeer dynamische omgeving. Maar diezelfde dynamische configuratie en recovery wil ik met mijn bedrijfsapplicaties hebben. Als de ene server down gaat en de andere ervoor in de plaats komt, wil ik het meteen aan het werk hebben.'

Portier: 'De Google-aanpak.'

Marinescu: 'Ja, en dat is het idee van Jini maar de wereld heeft dat niet begrepen.'

Longshaw: 'Daarom is Jini een goede oplossing als je distributie nodig hebt, maar de kernvraag is: heb je het nodig? Mensen distribueren dingen waar het niet nodig

Floyd Marinescu

is, of om onduidelijke redenen.'

Maassen: 'Bijvoorbeeld availability.'

Longshaw: 'In sommige gevallen gebruiken mensen de schaalbaarheid in de business tier. Waarom is het schaalbaarder in die systemen dan in andere systemen?'

Marinescu: 'Je zou nog steeds je presentatie- en businesslaag op dezelfde doos kunnen clusteren. Maar ik vroeg me af: jij zegt, dat je beter niet kunt distribueren. Ben je nooit in een situatie geweest dat je de middle tiers apart moest schalen, of is dat alleen maar een mythe?'

Longshaw: 'Nee, het gebruik van resources van je applicatieserver is wezenlijk anders dan dat van je webserver. Ze moeten anders geconfigureerd worden en gebruiken meer geheugen. Eén van de belangrijkste punten bij requirements zijn de kosten. Daar hoor je nooit iemand over. Voor een webservice kun je zeggen: "Geweldig, we zetten daar Apache neer, en we installeren TomCat als onze servletcontainer. We kunnen er drie nemen voor onze zesduizend gebruikers. Dan hebben we een paar applicatieservers nodig om het te ondersteunen". Misschien heb je - als het lichte businesslogica is - maar één applicatieserver nodig. Als je IBM Websphere kiest bijvoorbeeld, dan is dat één licentie in plaats van drie. Maar als je op moet schalen naar zes applicatieservers, heb je zes licenties nodig, maar misschien zou je je hardware anders kunnen configureren zodat je geld kunt sparen met disks. Bij een vergelijking zijn heel veel variabelen betrokken: de kosten van licenties, van hardware, wat de overall systeem kosten zijn, of je een chain systeem hebt of een inverted triangle systeem: een grote front-end en een kleine back-end. Het ligt er helemaal aan wat je calculaties zijn en wat het kost, aan de verschillende lagen en de diverse fysieke tiers.'

> "Toen EJB's net geïntroduceerd werden, noemden we het monkey-programming"

Marinescu: 'Dus er zijn gevallen waarin je een aparte schalende middle tier nodig hebt en er is een case voor EJB?'

CIRKELREDENERING Portier: 'Nou, ik ben vast geïnfecteerd door de Open Source gedachte, maar is het argument van het drukken van licentiekosten als rechtvaardiging voor het binnenhalen van dure EJB-servers (en hun licenties) niet een beetje een zichzelf bevestigende cirkelredenering?'

Longshaw: 'Ik denk nog steeds dat je een EJB-laag nodig hebt, tenzij je fat clients hebt. Neem het typische Amazon-systeem, daar heb je een development vrije zone, met je webservice erin. Je webservices lopen op Apache op high end Unix systemen. Ze doen reverse proxying op iedere jsp of server content door de firewall naar de applicatieservers die in het midden zitten. De applicatieservers hebben de J2EE webcontainer en EJB containers op dezelfde machine. Afgaande op wat ik gezien heb, kan ik concluderen dat er niet veel mensen zijn, die de webcontainer scheiden van de J2EE-container.'

Portier: 'Maar zelfs als je het scheidt, waarom zou de communicatie dan via RMI gaan, en niet XML over http?'

Marinescu: 'Je noemt een belangrijk punt, zullen webservices de gedistribueerde value propositie van EJB wegnemen?'

Portier: 'Nee, ze nemen het niet weg. Mijn vraag is, bieden ze een bruikbaar alternatief?'

Marinescu: 'Ik stelde die vraag aan iemand van de J2EE spec: zouden webservices met J2EE een alternatief kunnen vormen voor je EJB-business logica? Ze gaven een standaardantwoord over de component framework voordelen, zoals pooling en transaction. Ik denk dat webservices gebruikt zouden moeten worden voor integratie, niet voor communicatie tussen tiers. Er is teveel overhead mee gemoeid om een zwaargewicht product als webservices voor te gebruiken, dus als je moet distribueren tussen Java dan zou je EJB's moeten gebruiken.'

Portier: 'Ik zou zelfs verder gaan. Ik ben geen webservices man, maar ReST (je weet wel: representation state transfer) is een geweldig goede beweging tegen SOAP zoals het toen was, en ze werden sterk beïnvloed toen SOAP 1.2 kwam. Ze heeft zo ook grotendeels de 1.2 versie van SOAP beïnvloedt. En je ziet die invloed ook bij de huidige SOAP implementatie in Apache AXIS. Wanneer je je webservice op Access deployt, kun je je arguments versturen door ze in een SOAP-message te verpakken, maar ook rechtstreeks als HTTP request parameters. Je distributie-protocol wordt zo heel lightweight en nog steeds loosely coupled. Maar ja, het is geen RMI, dat moet ik toegeven. Dus ik vraag me af: "Waar gaat het naartoe?" En jij zegt eigenlijk: dat wordt bepaald door de spelers, niet door de consumenten.'

DRIEHOEK Marinescu: 'We weten allemaal dat Microsoft SOA zwaar pusht, en dat is gebaseerd op webservices. Ik was dit jaar op de PDC aanwezig bij een ronde tafel discussie. Martin Fowler was er ook. Net als Sun probeert Microsoft SOA te gebruiken voor zowel integratie als voor inter-tier communicatie. Fowler verzette zich daar hevig tegen. Hij zei dat webservices bedoeld zijn voor integratie, niet voor communicatie binnen een enterprise applicatie zelf. Daar moet ernstig getwijfeld worden aan het nut ervan.'

Longshaw: 'Dat is erg angstaanjagend: ik werkte aan een .NET project voor een heel grote UK retailer, en een van de mensen met wie ik samenwerkte ging naar een Microsoft-dag. Die vroeg Don Box of hij dacht dat webservices voor inter-tier communicatie gebruikt zouden moeten worden, en hij zei "Ja.". Die man zei: "Nee, dat is een ramp." Kijk, het is net een driehoek. De basis van de driehoek is je datasources en je integratie, dan heb je je business in het midden en je user interface vooraan, en die driehoek stelt typisch een webapplicatie voor. De user interface vooraan genereert HTML. Als je een webservice nodig hebt, heb je een webservice interface op hetzelfde niveau als de HTML-interface en de rest van de applicatie. De interne communicatie is helemaal de zorg van die applicatie en als je gedistribueerd gaat, is dan moet je aan de applicatie overlaten of het RMI gebruikt of .NET remoting of DCOM, of Corba, of webservice protocollen. Maar het heeft geen zin om webservice protocollen te gebruiken als het geen webservice is.'

Portier: 'Je bedoelt, het zou niet publiek gemaakt moeten worden?'

Longshaw: 'Nee, het punt is dat je geen webservices protocollen zou moeten gebruiken voor intertier communicatie. Ze zijn ontwikkeld voor iets groters, voor integratie.'

Portier: 'Gaat het om iets groters dan om de discussie over het protocol dat over de draad gaat? Of over de filosofie wat publiek is en wat niet? En hoe we integratie moeten realiseren, want in de echte wereld weet je niet waar je grenzen zijn. Vandaag is mijn grens hier, maar morgen fuseert mijn bedrijf met een ander bedrijf. En dan wordt het van "Toen waren we nog gewend om webservices te gebruiken en nu is het intern en kunnen we het niet meer gebruiken?" Het komt neer op het Jini verhaal. Waarom geen distributie op componentniveau, waar je inderdaad met elke component in het systeem dan kunt praten? Natuurlijk komt het dan neer op de vraag, wil je echt dat het performant is?'

Portier: 'Een nieuw onderwerp: we hebben hier drie pattern deskundigen, ik heb twee vragen over het gebruik van patterns. Waar denk je dat ze in passen? Het woord monkey-developer viel al ...'

Longshaw: 'Monkeypatterns, ha ha ha!'

Maassen: 'Als we het toch over patterns en humor hebben: een bekende van mij, Solveig Haugland schreef "Dating Design Patterns". Dat is het originele manuscript van de Gang of Four patterns, allemaal over patterns toegepast op dating. Ze is echt technisch, dus het is een poging om patterns echt te verklaren vanuit dating. Zo is een security proxy ...'

Longshaw: 'een vriend die je stuurt om ...'

Portier: '...te gluren voor achter-de-schermen informatie.'

(deze discussie leidt tot algehele vrolijkheid bij de deelnemers, red.)

Maassen: 'Sorry hiervoor.'

Olav Maassen

KOOKBOEK Portier: 'Terug naar de twee vragen. Waar gebruik je patterns? Er zijn veel mensen die EJB's moeilijk bereiken, veel mensen coderen gewoon Java. Waar passen patterns, is het classified information? Is het niet de bedoeling dat sommige mensen het gebruiken? Als je een cursus geeft aan een Cobol programmeur om hem op te leiden tot EJB programmeur, wanneer praat je dan over patterns?'

Maassen: 'Een probleem is op dit moment dat patterns misbruikt worden. In de eerste plaats is er het woord pattern, dat op ieder nieuw gepubliceerd boek wordt geplakt. Dat is één ding. Het andere is dat mensen het als een rookgordijn kunnen gebruiken. Programmeurs zouden het wel moeten leren, maar het is niet de oplossing voor alles. Het is één van de middelen die je bij je zou moeten hebben als communicatieprotocol. Mensen zouden het moeten herkennen als een goede oplossing, voor sommige situaties, maar niet als doel op zichzelf.'

Marinescu: 'Als je een beginner bent, op het gebied van Java of wat dan ook, dan kun je theorie leren, zoals de specs lezen, maar daar heb je weinig aan. Je moet de applicatie leren, en dat is dan in dit geval patterns. Ik denk dat je patterns zo vroeg mogelijk moet leren en je eraan moet wijden. Wanneer je aan projecten gaat werken, heb je dan de ervaring en de voordelen hebben van duizenden anderen die de patterns eerder gebruikt hebben.'

Portier: 'Ik heb hetzelfde gevoel.'

Marinescu: 'Een analogie die ik zou willen geven; een recept voor een gerecht als een pattern: een gedocumenteerde beste oplossing voor iets. Zou je willen beginnen met koken zonder kookboek? Als ik dat doe, is resultaat niet goed.'

Portier: 'De analogie die ik gehoord heb, was die met het schaakspel. Als je naar de schaakclub gaat, leer je eerst waar alle stukken toe dienen, de syntax, en dan leer je de patronen. De tweede les is meteen, nu je de regels kent: hier zijn de patronen, hier zie je hoe het eerder gedaan.'

Maassen: 'Terug naar de kookanalogie; een recept is een pattern dat je niet hoeft te volgen. Binnen de pattern wereld zie je een hoop discussie in de trant van: je doet het verkeerd, je volgt het recept niet precies zoals het opgeschreven is.'

Portier: 'Je wilt een chefkok, om het aan te passen, zodat je niet iets krijgt als McDonalds, wat overal ter wereld hetzelfde smaakt.'

Maassen: 'Ik zou patterns niet willen vergelijken met McDonalds. Het gaat niet om de pattern beweging. Het is geen religie, maar een recept dat je aan je behoeften kunt aanpassen.'

Longshaw: 'Ik maak me een beetje zorgen over de J2EE focus pattern, omdat ze een handige gids zijn om nieuwe en bestaande patterns te implementeren in een J2EE context, maar het betekent niet dat je de code uit de boeken haalt en in je applicatie kunt knippen en plakken.'

Maassen: 'Maar dat gebeurt.'

McDonalds Portier: 'Misschien was het alleen tussen EJB 1.0 en 2.0, maar waren de EJB patterns niet nodig om alle fouten in EJB 1.0 af te dekken? En nu de specificatie er is hoeven een heleboel ervan niet meer gebruikt te worden?'

Marinescu: 'Er zijn patterns die specifiek geduplicceerd zijn door EJB 2.0, één ervan is de notie van value objects, terwijl je daarvoor value objects op de entity bean gecreëerd werden, in een get value object method. Dat is vreselijk, want value objects zijn werkelijk data transfer objects, een network issue. Dat is zeker geduplicceerd. Je zou zelfs kunnen zeggen dat Session Façade een van de belangrijkste value propositions is. Het is niet meer zo handig als vroeger, omdat je niet meer met het gedistribueerde issue rekening hoeft te houden, maar het is nog steeds een bruikbaar framework om je logica te organiseren.'

Marc Portier

Longshaw: 'Een deel ervan is het stoppen van gaten, maar een ander deel is business infrastructuur. Session Façade is een plaats voor je infrastructuur: als je het in plaats van remoting stopt, dan is het infrastructuur spul. Wanneer je een laag voor je neer wilt zetten, een servicelaag vóór je domain objects om business redenen, dan is het gemakkelijker om de code te bewerken. Het maakt een beter gelaagde applicatie. Of dat al dan niet ook een network te maken heeft, is niet de vraag. Ik denk dat in de vroege dagen van patterns het allemaal bij elkaar zat, nu wordt het meer gescheiden. De meer recente versies zeggen: oké, dat zijn twee verschillende problemen, laten we ze gescheiden aanpakken. Ja, we willen een applicatieservice maar we willen geen Session Façade ervoor, omdat we helemaal in een server container werken en we geen distributie nodig hebben. Eén van de dingen bij het dupliceren van patterns, is dat bij composed entity ...'

Marinescu: 'Ja, dat vergat ik te melden, dat is zeker geduplicceerd.'

Longshaw: '...het pattern op zich staat, het leert mensen een hoop bruikbare zaken, over objecten en dependency's en dependency trees van objecten, persistentie van EJB's en entity beans enzovoorts. Ook nu is het zinnig om het te lezen, want uiteindelijk zullen ze een hoop van het issue begrijpen ook al is de oplossing container managed persistentie, container managed relationships, tik tik tik, daar ga je. Het leren van het pattern komt neer op het leren van het principe van dependant objecten en van meer course grained objecten die de eigenaar ervan zijn. De oplossing is toevallig ingebed in het platform.'

Sweet spot Portier: 'Hoe denken jullie over tools als Optimal J, Sygel's WMEE en meer in het algemeen de MDA beweging?'

Marinescu: 'De middleware company waarvoor ik werk en die de eigenaar is van de Server Side, heeft een productiviteitsstudie met Optimal J gedaan. We maakten gebruik van twee ervaren teams waarvan de één een MDA tool (OptimalJ) gebruikte en de ander een populaire IDE. We hebben ze getest met het implementeren van een systeem dat we van tevoren hadden gespecificeerd. Aan de ene kant was er een team dat bijgestaan werd door een OptimalJ man, en aan de andere kant was er een patterns based code centric aanpak, en het resultaat was dat het OptimalJ team vijfendertig procent eerder klaar was, terwijl ze zelfs een leercurve had-

den om de tool te leren. Dat staat in een studie, die op de middleware site staat. Er is dus wel degelijk empirisch bewijs dat suggereert dat de MDA-aanpak helpt in concrete projecten.'

Longshaw: 'Het helpt bij alle scenario's, want alle toolboxen hebben sweet spots. Het kan je dus helpen bij die soort van applicatie, maar bij anderen niet of misschien is het wel nadelig.'

Maassen: 'Ik ben afgelopen vrijdag bij de Optimal J-labs geweest, en het heeft me beïnvloed.'

Portier: 'Was je onder de indruk of ben je beïnvloed?'

Maassen: 'Ik was er eerst heel sceptisch over, maar ik geloof nu dat ze een goede kans maken om te slagen. Ik ben sceptisch over het hergebruik van domainmodellen. Ik denk niet dat je een J2EE-applicatie snel verhuist naar een andere container, ik denk niet dat je dat vaak doet. Je hebt wel een zeer interessante productiviteitswinst. Misschien is er een sweet spot, maar ik denk dat het toegepast kan worden op veel domeinen. Maar er zullen altijd applicaties zijn die je sneller handmatig kunt bouwen.'

Portier: 'Dus ik hoor wat scepsis. In de zin van: ja, maar het helpt alleen bij het schrijven van saaie code die J2EE je verplicht te schrijven.'

Maassen: 'Ik gebruik een heel pragmatische aanpak: MDA gaat over het genereren van code, dat is alles waar het om draait.'

Portier: 'De test bij de middleware company zou ook een derde groep hebben moeten omvatten die X-doclet of iets dergelijks zou gebruiken, dan zou de uitkomst misschien anders geweest zijn.'

XP-ADEPT Marinescu: 'De misvatting bestaat dat MDA-tools alleen maar code kunnen genereren. Het lijkt erop dat er een beweging is om code te genereren, maar MDA gaat daar niet over. MDA gaat over het creëren van een applicatie waarbij je model altijd synchroon loopt met je code. Je kunt veranderingen aanbrengen op verschillende abstractieniveaus, en ze zullen allemaal synchroon gehouden worden. MDA probeert dus ook de last van het onderhoud te simplificeren, dus je kunt een jaar later bij een applicatie aan het model gaan sleutelen. Ik weet niet hoe succesvol ze zijn, een jaar geleden was OptimalJ nog niet tot alles in staat, maar ze zeiden dat ze hard aan het werk waren met nieuwe oplossingen. Hoe dan ook, MDA gaat over veel meer dan over codegeneratie.'

Maassen: 'Daar ben ik het mee eens, maar uiteindelijk komt het erop neer dat je code moet leveren. Dat is het hele eieren eten.'

Portier: 'Het lijkt erop dat we een XP-adept in ons midden hebben.'

Maassen: 'Niet alleen XP, agile software development. Ik denk dat het absoluut helpt, ik heb ook gezien dat je het model aanpast en die modificaties in je applicatie verschijnen. Niettemin: uiteindelijk zal er natuurlijk code moeten zijn.'

Longshaw: 'Het helpt zeker aan de buitenkant, maar of het ook bij de kern helpt is de vraag. Wanneer je MDA in de kern van de zaak beschouwt, zie je dat er voorlopers zijn in UML. Merkwaardig eigenlijk, je bent in feite een andere taal aan het uitvinden, terwijl er al twintig talen zijn iets dergelijks doen. Maar ik geef toe, waarschijnlijk is dat hetzelfde wat chip programmeurs vijftien jaar geleden zeiden.'

Portier: 'Ik deel wel iets van de scepsis, als je zegt: uiteindelijk heb je natuurlijk code nodig. MDA maakt UML tot een nieuwe programmeertaal, wat het niet is, en opeens moet je al die nuances weten, niet zozeer van UML maar van hoe de code generatie geïnterpreteerd zal worden door UML, om de generatie te laten verlopen zoals ik dat zou willen.'

Marinescu: 'Ik denk dat dat een probleem is, dat mettertijd opgelost zal worden. De vraag is: hoeveel doe je in het model? Ik heb een MDA-debat gevoerd met Compuware mensen op JavaOne en al die mensen in het publiek vroegen: hoe doe je dit en hoe dat. Ze spraken over hoe de volgende versie van UML daar voorzieningen voor zou hebben. Toen zei iemand inderdaad: is dit niet een volgende taal aan het worden? Waarvoor hebben we die nodig?'

> "Daar schrik ik van: dat is de 'draggy, droppy, pointy, clicky' benadering van patterns"

Maassen: 'Het wordt een visuele programmeertaal.'

Marinescu: 'Maar om even de vraag aan te snijden, hoeveel code je kunt genereren met MDA tools: er zijn al tools die J2EE design patterns in hun tool aan het integreren zijn, zodat het niet alleen code generatie is, maar ook een hoop van het pattern werk dat je zelf sowieso gedaan zou hebben.'

Maassen: 'Het helpt wanneer het pattern het voor je doet.'

Marinescu: 'Je kunt ook aan het pattern denken. Je kunt het tekenen op een abstract niveau.'

Longshaw: 'Daar schrik ik ook van, want dat is de 'draggy, droppy, pointy, clicky' benadering van patterns. Ik pak mijn patterns, zet ze bij elkaar als delen van een legpuzzel en hup daar is mijn oplossing. Het gaat vooral om de stijl, ik heb gewoon het gevoel dat het niet goed kan zijn om op een knop te drukken en dan de patterns de code te laten genereren.'

Maassen: 'Als je door gaat naar de vergelijking met huizen: de vraag is of je in een nieuwbouwgebied prefab

datingdesignpatterns

'I ran the entire program today, Saturday April 5th. Got up and got dressed in the Decorator suit, then hit the antique stores (Exposed Accelerated Collector). Had brunch with Lucy and Cindy and got Cindy's number, though Lucy called later and asked me to teach her to ski (Big Fat Opening). Did the volunteer work for Humane Society 2-4 and got to know Leslie; am working on her from the Trojan Facade angle and am planning for her to invite me camping over Memorial Day. Finally, implemented Interested Listener on date with Sonja with spectacular results. Am discovering more patterns and strategies every day but think IL is most effective. More later; she just came out of the bathroom.'

Christopher Alexander

huizen wilt hebben of unieke huizen. Voor veel mensen zijn die prefab huizen prima, en zo is het voor veel bedrijven met prefab applicaties ook.'

Longshaw: 'Waar het op neerkomt is wat de sweet spot doet, als het geen 3 tier enterprise applicaties voor webbased shopping zijn. Als je context anders is, zou het wel eens niet kunnen werken. Je moet dus de context aan je tool aanpassen, want niet alles is als de Pet Store.'

Marinescu: 'Werkelijk?'

Longshaw: '...en er zijn ook geen drie miljoen ondernemingen die allemaal Amazon heten.'

NOTEPAD Portier: 'IDE's, we hebben het er al even over gehad. Wat gebruiken jullie? Kom op, zeg niet dat jullie het niet mogen zeggen.'

Maassen: 'Ik gebruik Eclipse.'

Portier: 'Sinds welke versie?'

Maassen: '2.0'

Marinescu: 'We gebruiken IntelliJ op de serverside sinds versie 2.0'

Maassen: 'Ik zou graag eens werken met IntelliJ'

Marinescu: 'En VI en soms Notepad.'

Allen: gegrinnik.

Longshaw: 'Eclips en ook VI.'

Maassen: 'Er is nog steeds geen IDE die dezelfde kwaliteit biedt als VI. Ik kan zoveel doen met een paar toetsaanslagen.'

Longshaw: 'en dan Intellisense.'

Portier: 'of 3626 parallelle cut & paste buffers, waaruit je kunt kiezen.'

Portier: 'Eclipse is gelukt. Staat Sun nu in de schaduw?'

Marinescu: 'Afgelopen week zei Sun dat wanneer de industrie het eens kon worden over plug ins voor IDE's, dat ze dan Eclipse zouden ondersteunen. Ze willen één api zodat al die plug ins voor Eclipse ook zouden werken in NetBeans. Ze willen nog steeds NetBeans ondersteunen, maar ook aansluiten bij de populariteit die Eclipse heeft. In feite is Oracle ook toegetreden tot de IDE plug ins JSR een paar maanden geleden.'

(Op het moment dat dit gesprek plaatsvond was de Java Tools Community (JTC) nog niet opgericht, red.)

Maassen: 'Kunnen ze niet gewoon de plug-in van Eclipse tot standaard verheffen.'

Marinescu: 'Dat gaat waarschijnlijk ook gebeuren, maar Sun zal zijn gezicht willen redden door het standaardisatieproces in te gaan.'

Portier: 'Is er iemand die kan uitleggen waarom Eclipse geslaagd is?'

Maassen: 'Eclipse is sneller dan NetBeans.'

Portier: 'Is het alleen maar de snelheid?'

Maassen: 'Het is ook meer gebruikersvriendelijk.'

Portier: 'Kom op, iedereen heeft zijn lievelingtheorie waarom het zo is.'

Longshaw: 'Het is gratis te downloaden en het lijkt erg op Visual Age, veel mensen gebruikten al Visual Age en je kon het dus zo downloaden en gebruiken.'

GEMEENSCHAP Terug naar IDE's:

Portier: 'Die zeventig procent adaptatie-graad van Eclipse, waar ligt dat aan?'

Marinescu: 'Dat komt zeker ook omdat het IBM is. De Visual Age mensen hoefden bovendien nauwelijks te migreren, en IBM heeft een grote bestaande klantenkring.'

Longshaw: 'En het is ook geen zevenhonderd dollar of zo. In de Microsoft omgeving betaal ik 2.000 dollar per jaar voor het MSDN-platform.'

Portier: 'Maar velen beschouwen het betalen van licenties in de Microsoft wereld als het betalen van belasting: ik ben ertoe verplicht.'

Longshaw: 'Maar in de Microsoft omgeving is het een heel ander verhaal. Daar kan ik met praktisch iedere klant werken, die ik tegen zou kunnen komen. In de Java wereld moet ik vijf of zes enterprise developer licenties kopen. Ik kan het me niet veroorloven 1.200 dollar per jaar aan licenties uit te geven.'

Portier: 'Ik heb mijn eigen lievelingstheorie. Natuurlijk heb je vergeleken met IntelliJ kostenissues, maar vergeleken met NetBeans is het weer anders. Eclipse is geen IDE, maar heeft zichzelf gepositioneerd als een platform dat gewoon werkt en waarvoor mensen zelf plug-ins schrijven. Dat voert tot mijn eigen theorie over .NET: .NET zal alleen maar slagen als er een gemeenschap is. Misschien ben ik naïef, maar dat is wat ik zie; community's en mensen maken het verschil.'

Dré de Man