Diverse organisatie hebben een eigen .NET ontwikkelstraat. Software Release Magazine. Focus was benieuwd naar hoe die er uitzien. Aanvankelijk waren we van plan op basis van een vragenlijst de ontwikkelstraten van een aantal bekende .NET-partijen op een rij te zetten. Niet iedereen vond het even gemakkelijk die lijst te beantwoorden. Zo ontstond de serie: *De ontwikkelstraat van ...* In dit nummer de eerste twee afleveringen. Volgend nummer komt waarschijnlijk Capgemini aan bod.

De ontwikkelstraat van Class A

Class A richt zich op kennisoverdracht, opleiding, advisering en coaching op het gebied van het Microsoft applicatieplatform. Het bedrijf is opgericht door Anko Duizer en Astrid Hackenberg.

1. Welke O/R mapper wordt gebruikt? Er is niet één vaste O/R mapper die wordt gebruikt. Wanneer het project qua omvang er om vraagt wordt een O/R mapper ingezet. We werken met Business Object Broker (BOB) van BAI of LLBGENPRO (zoals gebruikt in de RAD Race). Wanneer het een qua omvang 'klein' project betreft maken we gebruiken van de standaard Microsoft technologieën. We maken dan gebruik van typed datasets.

2. In hoeverre houdt u rekening met de ontwikkelingen bij Microsoft op dit gebied? We volgen de ontwikkelingen van Microsoft op de voet. We gaan er van uit dat Microsoft een O/R mapper gaat leveren op termijn. Met name het LINQ project wordt met veel interesse gevolgd.

3. Is er een standaard framework met klassen waarop iedere applicatie is gebaseerd? Dit hangt sterk af van de klant. De meeste enterprise klanten hebben een eigen framework met klassen. Wanneer de klant het wenst maken we gebruik van het standaard framework van de klant. Als de klant geen standaard framework heeft dan gebruiken we onze eigen base klassen.

4. Worden kant en klare modules gebruikt zo ja waarvoor/ welke? Ja, we maken met name gebruik van de standaard Microsoft EntLib 2.0 modules. Deze proberen we zo min mogelijk aan te passen. Op deze manier gebeurt de evolutie en het onderhoud door Microsoft. We gebruiken deze modules voor het volgende:

- Exception management
- Logging
- User Interface (implementatie van het MVC pattern)
- Authenticatie
- Autorisatie
- Data acces

5. Wordt er code gegenereerd en zo ja hoe? Afhankelijk van het gebruikte framework wordt er code gegenereerd. De code generatie beperkt zich tot data access en data representatie objecten. De code wordt in het geval van BOB gegenereerd op basis van een klasse modellen. In het geval van LLBGENPRO is de database het uitgangspunt.

6. Maakt u gebruik van modellering en is er een direct verband tussen de uiteindelijke code en het model? Ja, we maken gebruik van modellering. Op dit moment maken we gebruik van Visual Studio 2005 als ontwikkelgereedschap. In VS 2005 zijn de class modellen de code en omgekeerd. Er is dus zeker een direct verband.

7. Werkt u ook omgekeerd? Wanneer het bestaande code betreft die aangepast moet worden, dan kunnen we ook omgekeerd werken.

8. Gebruikt u DSL's? We hebben op dit moment geen speciale aandacht op het gebruik van Domain

Specific Languages. Wel gebruiken we de DSL's zoals Microsoft deze aanlevert met Visual Studio.NET 2005.

9. Zo nee, bent u van plan deze te gaan gebruiken? Wanneer? We willen zeker meer gebruik maken van DSL's. Binnen de ontwikkeling van Microsoft is het duidelijk een belangrijk onderwerp. Class-A heeft contacten met Avanade om deze ontwikkeling bij te houden. We willen gezamenlijk workshops gaan leveren en whitepapers schrijven rondom dit onderwerp. Hierdoor gaan DSL's ook meer aandacht krijgen binnen de softwareontwikkelstraat.

10. Welke IDE('s) wordt/ worden er gebruikt? Microsoft Visual Studio 2005 Team System. VS 2005 is een beter uitbreidbare omgeving. De Class-A softwareontwikkelstraat is geheel terug te vinden in Visual Studio Team System. Wanneer een nieuw project wordt aangemaakt dan kan de Class-A template worden gekozen. Dit zegt iets over de volgende onderwerpen:

- Team/ rollen
- Proces
- Op te leveren document
- Achtergrondinformatie
- Code templates
- UML templates
- Risk management proces
- Bug management proces
- Code standaarden
- Source control
- Dagelijkse/ continuous build
- Check in rules

11. Hoe lang bestaat de ontwikkelstraat in deze vorm en verwacht u binnenkort veranderingen? De softwareontwikkelstraat is een evolutionair proces. Er worden regelmatig verbeteringen aangebracht. De nieuwe Microsoft technologie heeft ook veel invloed op de invloed van de softwareontwikkelstraat. Recent heeft de introductie van Team Foundation Server veel invloed gehad op de invulling van de softwareontwikkelstraat. Daarnaast proberen we ervaringen bij klanten en project direct terug te laten komen in de softwareontwikkelstraat. We verwachten dus zeker veranderingen binnenkort. De oorspronkelijke opzet van de softwareontwikkelstraat stamt uit 2002.

12. Op welke manier wordt getest en wanneer? Er wordt op verschillende manieren getest. Voor de ontwikkelaar is het voornamelijk unit testing. We maken veel gebruik van unit testing om de kwaliteit en stabiliteit van de code te meten. Met de komst van Visual Studio 2005 is dit een integraal onderdeel van de ontwikkelomgeving. Verder worden er stress-testen, functionele testen, integratie- en acceptatie-testen uitgevoerd. In ieder ontwikkelteam is er een rol "Test" aanwezig. Afhankelijk van de omvang wordt de rol door één of meerdere personen ingevuld.

13. Welke talen gebruikt u (verdeling)?

- C# (50%)
- VB.NET (50%) (Toelichting: VB.Net veelal op verzoek.)

14. Hoe dwingend is het gebruik van de straat binnen uw organisatie? Bij ieder project is de softwareontwikkelstraat het uitgangspunt. De straat is wel flexi-

Tabel 1. Omschrijving elementen van een software ontwikkelstraat.

Element	Beschrijving
Team	De basis voor softwareontwikkeling is een goed ingewerkt en helder communicerend team. Hiervoor is een duidelijke rolverdeling een vereiste.
Infrastructuur	De software moet draaien op een gegeven infrastructuur, indien mogelijk moet de infrastructuur zo volledig mogelijk worden benut. Aspecten zijn schaalbaarheid, beschikbaarheid en beveiliging.
Proces/ Werkwijze	De wijze waarop software wordt ontwikkeld. Welke stappen worden genomen? Welke documentatie is noodzakelijk? Wat is de rol van de gebruiker?
Architectuur	De blauwdruk voor de softwareontwikkeling. Hoeveel lagen worden er gebruikt? Wat zijn de patterns? Hoe wordt omgegaan met beveiliging? Op wat voor manier vindt integratie plaats?
Tools	De gereedschappen die worden gebruikt. Bijvoorbeeld Visual Studio.NET voor de code ontwikkeling. Maar ook tools voor source beheer en modellering dienen gedefinieerd te worden.
Templates	Een set aan templates voor ontwerpdocument, requirements, test rapportage, etc.
Standaarden	Een beschrijving van code standaarden, naamgeving richtlijnen, modellering standaarden.

bel en geen harnas. Op basis van de behoeften van de klant of de situatie kan de straat deels worden toegepast. Er kunnen concessies worden gedaan op het gebied van te gebruiken technologie of hulpmiddelen.

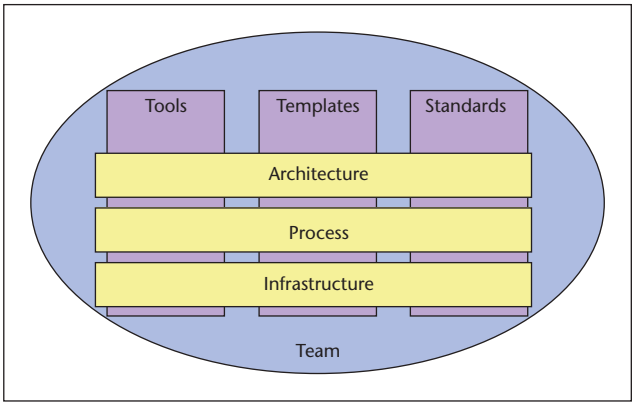
We doen echter nooit concessies aan het teammodel (MSF) en het procesmodel (MSF).

15. Gebruikt u een bepaalde methodologie? Microsoft Solutions Framework (version 4) for Agile. Een ontwikkelstraat bestaat niet alleen uit tools en software, maar ook uit processen, methodes en mensen. Daarom hebben we aan het eind van de vragenlijst wat ruimte overgelaten voor een korte toelichting daarop.

Achtergrondinformatie In onderstaande lijst staan de belangrijkste redenen om een software-ontwikkelstraat te implementeren:

- Uniformiteit
- Kwaliteit
- Onderhoudbaarheid
- Goede samenwerking
- Voorspelbaarheid (tijd, kosten)
- Efficiency
- Controleerbaarheid

Elementen In figuur 1 zijn de elementen van een software ontwikkelstraat weergegeven:



Figuur 1: Software ontwikkelstraat elementen.

Voor een succesvolle implementatie moeten alle elementen zoals beschreven in figuur 1 in meer of mindere mate worden ingevuld. In de tabel staat een globale beschrijving per onderdeel.

Regels Naast deze elementen zijn er voor een goede ontwikkelstraat een aantal gouden regels te onderkennen. Deze zijn niet persé gekoppeld aan een element maar kunnen over elementen heen gaan. Onderstaand de toptien van belangrijkste regels:

Anko Duizer en Astrid Hackenberg van Class A.

1. "Build" dagelijks
2. Definieer duidelijke rollen en verantwoordelijkheden, communiceer!
3. Draag zorg voor goed ingericht versiecontrole en -beheer
4. Motiveer het team dagelijks
5. Stabiliseer iedere twee maanden minimaal een week
6. Testen is de eerste stap in het ontwikkelproces
7. Vertrouw nooit uitsluitend op de tools
8. Denk aan de infrastructuur
9. 'Componentize' de applicatie
10. Onderhoud de architectuur dagelijks

Class-A gebruikt de volgende technieken en technologie in de softwareontwikkelstraat:

- Microsoft .NET 1.1 en 2.0
- Visual Studio Team System
- Team Foundation Server
- Microsoft SQL Server
- EntLib 2.0
- Enterprise Architect
- UML
- BOB
- LLBGENPRO
- Sharepoint Portal Server

De ontwikkelstraat van Info Support

1. Is er een standaard framework met klassen waarop iedere applicatie gebaseerd is?

- *Endeavour* levert een logische en technische referentie-architectuur. In de logische architectuur wordt de structuur en opdeling van applicatie en services beschreven. Zo wordt er onderscheid gemaakt in de business services, processen, front-end en platform services.
- *Technische Referentie Architectuur:* Een referentie-architectuur waarin de technische basisarchitectuur van een enterprise applicatie beschreven wordt. De technische referentie architectuur (TRA) biedt een technische invulling van de logische referentie architectuur (LRA). Onder meer op het gebied van transactiemanagement en error-afhandeling worden technische keuzes gemaakt. Tevens wordt een technische invulling gegeven aan de verschillende typen logische onderdelen die genoemd worden in de LRA. De TRA dient als basis voor de Technische Solution Architectuur (TSA) die in de architectuurfase gemaakt wordt. In deze projectspecifieke architectuur worden de afwijkingen ten opzichte van de referentie-architectuur gedocumenteerd.

2. Worden kant en klare modules gebruikt, zo ja, waarvoor/welke? Voor het abstraheren van veel gebruikte mechanismen zijn kant-en-klare building blocks gemaakt:

- *DataAccess:* Building block dat de communicatie met de database verzorgt. DataAccess implementeert de standaardmethode van databasetoegang conform de technische referentie architectuur. Het DataAccess building block biedt een eenvoudiger te gebruiken abstractielaag bovenop ADO.NET.
- *Logging:* Building block waarmee informatie over de toestand van een applicatie wordt vastgelegd conform de Technische Referentie Architectuur. Onder het Logging building block vallen zowel technische fouten, oftewel excepties, die in de applicatie zijn opgetreden als ook het run-time gedrag van de applicatie.
- *Exception Management:* Building block dat opgetreden fouten conform de technische referentie architectuur afhandelt. Hierbij kunnen technische en functionele fouten op een verschillende wijze worden afgehandeld
- *WebFramework:* Building block dat de navigatie en presentatielaag implementeert uit de technische referentie architectuur. Door een strikte scheiding tussen de data (model), het tonen van de data (view) en de navigatie (controller) wordt de herbruikbaarheid van schermen sterk vereenvoudigd. Standaardzaken als parameteroverdracht tussen webpagina's en het tonen en afhandelen van functionele fouten zijn ingebouwd. Verder is de vertaling tussen UML activity diagrammen en de uiteindelijke navigatie eenduidig vastgelegd zodat ontwerp en implementatie nauw met elkaar zijn verbonden. Het gebruik van het webframework zal de time-to-market voor webapplicaties sterk verkorten.
- *WinFramework:* Building block dat de navigatie en presentatielaag implementeert uit de technische referentie-architectuur. Door een strikte scheiding tussen de data (model), het tonen van de data (view) en de navigatie (controller) wordt de herbruikbaarheid van schermen sterk vereenvoudigd. De vertaling tussen UML activity-diagrammen en de uiteindelijke navigatie is op een eenduidige manier vastgelegd zodat ontwerp en implementatie nauw met elkaar zijn verbonden. Het gebruik van het winframework zal de time-to-market voor winforms-applicaties sterk verkorten.

> De technische referentie architectuur (TRA) biedt een technische invulling van de logische referentie architectuur (LRA)

Daarnaast zijn er een tal van tools die het ontwikkelen ondersteunen, zoals:

- *AutoBuild:* Een tool die het dagelijkse buildproces verzorgt. AutoBuild voert alle taken uit die nodig zijn om een 'compleet' project te builden. Hierdoor kan dagelijks vanuit een laatste en consistente build worden gewerkt en getest.
- *AutoCheck:* Een tool waarmee geautomatiseerd kan worden gecontroleerd of, en in welke mate, ontwik-

kelde code voldoet aan standaards en richtlijnen van leveranciers, aangevuld met Info Support standaards en richtlijnen.

- *AutoDoc:* Een tool die automatisch technische documentatie genereert op basis van het commentaar dat in de source-code is opgenomen. De structuur van de documentatie die wordt gegenereerd is vergelijkbaar met structuur van de documentatie van de Microsoft MSDN library en is beschikbaar in HTML-formaat en als compiled help file.
- *UnitTest:* Een tool waarmee een applicatie geautomatiseerd aan testen onderworpen kan worden. UnitTest adopteert het unittest concept, waarin de kleinst compileerbare eenheid (meestal een class) door middel van scenario's getest wordt op eisen die zijn gesteld in het ontwerp. Bij het testen wordt uitgegaan van een test-first-principe, waarbij alvorens te implementeren eerst de testen worden geschreven. Het gebruik van UnitTest zorgt ervoor dat de constante werking van een systeem dat in bouw of onderhoud zit kan worden gegarandeerd. Op ieder moment in het proces kunnen alle testen geautomatiseerd worden uitgevoerd en kan inzicht worden verkregen in de gevolgen van een refactoring slag of bugfix.
- *UnitTest DW:* Een tool waarmee inzicht wordt verkregen in het testproces. UnitTest DW gebruikt de testresultaten uit de UnitTest ten behoeve van de analyse van het testproces. Grafisch wordt inzichtelijk gemaakt of de code stabiel is, of de aantallen features stabiel zijn en welke functionaliteit een performance bottleneck kan zijn.
- *Project Portal:* Een tool waarmee dynamische projectinformatie wordt getoond. Een aantal Endeavour-producten levert specifieke informatie. Hieronder vallen UnitTest, AutoBuild en AutoCheck, waarvan de gegevens in de Project Portal zichtbaar worden gemaakt.
- *Digital Coach:* De tool Endeavour Digital Coach biedt voor iedere afzonderlijke deelnemer een softwareproject en een website aan met daarin de procesbeschrijving ondersteund met belangrijke informatie, ervaringskennis, checklists, links en trainingen.

We maken veel gebruik van unit testing om de kwaliteit en de stabiliteit van de code te meten

3. Wordt er code gegenereerd en zo ja hoe? Omdat gegenereerde code ook onderhouden moet worden, zoekt Info Support altijd eerst naar andere mogelijkheden, zoals generieke oplossingen, meta-gegevens, etc. Op deze manier ontstaat er minder te onderhouden source-code. Voor sommige zaken is genereren wel de beste optie, zo worden de database en standard stored procedures gegenereerd op basis van aanpasbare templates.

4. Maakt u gebruik van modellering en is er een direct verband tussen de uiteindelijke code en het model? Er wordt gebruik gemaakt van modellering door middel van UML 2.0, maar primair voor analyse doeleinden. Er vindt dan ook geen volledige code-generatie plaats. Voor sommige dingen, zoals het datamodel wordt wel code gegenereerd.

5. Werkt dat ook omgekeerd? Voor het datamodel geldt dat er geen roundtrip engineering benodigd is en dat wordt ook niet toegepast.

6. Gebruikt u DSL's? Geen extra DSL's ten opzichte van wat Microsoft standaard aanbiedt.

7. Zo nee, bent u van plan deze te gaan gebruiken? Wanneer? Ja, zo gauw Microsoft een goede ondersteuning levert voor het bouwen van eigen DSL's (met volledige designers), dit is nu nog steeds in een bèta-stadium.

8. Welke IDE('s) wordt/worden er gebruikt? Visual Studio 2003, Visual Studio 2005 Professional & Visual Studio 2005 Team System.

9. Hoe lang bestaat de ontwikkelstraat in deze vorm en verwacht u binnenkort veranderingen? De ontwikkelstraat in deze vorm bestaat ruim drie jaar. Er wordt continu doorontwikkeld aan de ontwikkelstraat. Zo is er een vast team van architecten, process engineers en ontwikkelaars. Ook nemen we deel aan Technical Adaption Programs van Microsoft. Zo zijn wij twee jaar geleden, als enige Nederlandse partij, betrokken bij de ontwikkeling van Team System. We zijn daar betrokken bij Software Design Reviews, Planning sessie et cetera. Hierdoor kunnen wij onze klanten garanties bieden voor een toekomstvaste oplossing. Door de directe contacten die we hebben met de Development Teams van Microsoft, worden wij ook veel gevraagd input te leveren aan de toekomstige ontwikkelingen bij Microsoft.

10. Op welke manier wordt getest en wanneer? Endeavour biedt een uitgebreide invulling voor de testdiscipline. Niet alleen wordt ondersteuning geboden voor unit-testen, maar ook voor systeem en functionele testen. Het afleiden van testcases en het gebruik van testtechnieken wordt uitvoerig ondersteund. Het testen start vanaf het begin van het project. In de inception fase wordt een testplan opgesteld en vanaf het moment

dat de eerste use-cases zijn opgesteld, worden ook de eerste testcases afgeleid.

11. Welke talen gebruikt u (graag verdeling)? De verhouding C# ten opzichte van VB.NET is ongeveer 70:30. We zien dit voor enterprise applicaties eerder groeien naar 80:20. De verhouding van de verschillende talen binnen een applicatie is ongeveer C# 60%, SQL 20%, ASP.NET 20%.

12. Hoe dwingend is het gebruik van de straat binnen uw organisatie? Binnen onze organisatie is het gebruik van de ontwikkelstraat verplicht en wordt afgedwongen door het Project Office. Deze bewaakt, ondersteunt en controleert het proces en het gebruik van de straat intern. Daarnaast worden metrieken verzameld over het proces.

13. Gebruikt u een bepaalde methodologie, zo ja welke?

- Het beschreven ontwikkelproces in de Endeavour Digital Coach is gebaseerd op het Unified Proces aangevuld met de best practices van Info Support. De methode is uitvoerig beschreven en is zeer verweven in de ontwikkelstraat door middel van de Digital Coach.
- Voor opstellen en verzamelen van metrieken wordt gebruik gemaakt van ISO 15939 met als PSM (Practical Sofware Measurement) praktische invulling daarvan.
- UML voor het opstellen van use cases.

14. Wordt er een O/R mapper gebruikt en zo ja welke? We volgen hierin de Microsoft-lijn. Dit betekent dat we kijken naar LINQ en DLINQ.

15. In hoeverre houdt u rekening met de ontwikkelingen bij Microsoft op dit gebied? Bij de ontwikkeling van Endeavour houden wij veel rekening met de ontwikkelingen bij Microsoft. Info Support heeft een hechte relatie met Microsoft die zich vertaalt in bijvoorbeeld het meewerken in TAP programma's (Technical Adoption Program) voor onder meer Team System en EntLib. Daarnaast worden regelmatig presentaties aan Redmond gegeven en via Conference Calls worden wij op de hoogte gehouden van de ontwikkelingen bij Microsoft. De relatie is zo goed dat Microsoft een aantal change request op het laatste moment nog heeft doorgevoerd in TFS die wij belangrijk achtten voor onze ontwikkelstraat en onze klanten. We kijken ook heel goed naar LINQ/DLINQ voor OR mapping. We hebben er al eens een presentatie over gegeven op DevDays (Linq en C#) en SDN (XLinq).

Endeavour Een ontwikkelstraat gaat meer dan over alleen technologie, softwareontwikkeling is een complex proces. Er zijn veel zijwegen en alternatieve routes mogelijk. Tijdens het proces dienen steeds opnieuw keuzes gemaakt te worden, bijvoorbeeld over aanpak, functionaliteit, technologie en architectuur. Hoe vaak komt het niet voor dat software-ontwikkelprojecten ondoorgrondelijk zijn geworden? Wat is de status? Waarom loopt het allemaal uit? Waarom kost het veel meer dan gepland? Kan iemand u de kwaliteit garanderen? U heeft het idee dat u de grip verliest.

Volledige transparantie Het Professional Development Center van Info Support beschikt over software ontwikkelstraten die volledige transparantie bieden, zodat u nauwgezet het proces kunt monitoren. Zo'n software ontwikkelstraat kunt u zien als het geheel van hulpmiddelen, processen, procedures en communicatiemiddelen die nodig zijn om de software te ontwikkelen en te onderhouden. Door onze ervaring in het ontwikkelen van software en onze gestandaardiseerde manier van werken in de ontwikkelstraat, kunnen we voortdurend uitspraken doen over de kwaliteit en productiviteit. Hierdoor wordt softwareontwikkeling een voorspelbaar en beheersbaar proces. Dit beperkt zich niet tot de eerste oplevering maar werkt door gedurende de gehele life cycle van een applicatie, dus ook tijdens het beheer en onderhoud.

'Software-ontwikkeling is een complex proces: er zijn veel zijwegen en alternatieve routes mogelijk'