Voorspelbaar en efficiënt software ontwerpen

Microsoft Application Architecture Guide, 2nd Edition

Edward Bakker en Clemens Reijnen

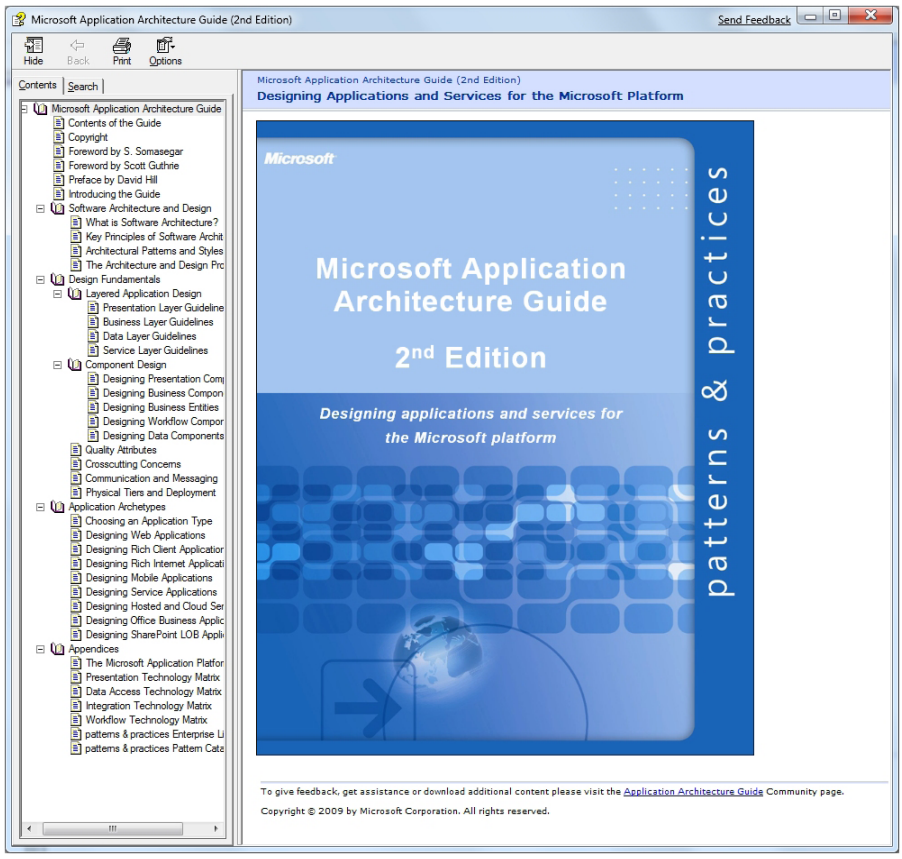
Het beheersen van de grammatica van een taal wil nog niet zeggen dat je een boek kunt schrijven. Zo is het voor de leesbaarheid en het begrip belangrijk om een boek logisch te ordenen en dat het aansluit bij de doelgroep. Paragrafen dienen zich te richten op één onderwerp en er dient een duidelijke overgang tussen paragrafen aanwezig te zijn. Dat bevordert de leesbaarheid. Dit is echter een andere vaardigheid dan het kennen van de grammatica.

Dit geldt niet alleen voor het schrijven van een boek maar ook voor het schrijven van software. Goede kennis van de grammatica van C# of Visual Basic. NET is nog geen garantie voor goed ontworpen software. Net als bij het schrijven van een boek of artikel heb je bij het schrijven van software structuur nodig. Het is de structuur die ons in staat stelt aan de gestelde kwaliteitseisen te voldoen en die daarnaast de oplossing leesbaar en onderhoudbaar maakt.

Architectuur

Door het gebruik van principes en richtlijnen bij softwareontwikkeling zijn we, vergelijkbaar met het schrijven van een boek, in staat software op een efficiëntere en voorspelbare manier te ontwikkelen. Binnen de softwareontwikkeling noemen we het creëren van de structuur van de applicatie ook wel architectuur, met andere woorden: het ontwikkelen van software op basis van architectuur.

Een goede architectuur is niet altijd even eenvoudig te realiseren. Gebruikers, maar ook architecten,ontwikkelaars en beheerders, hebben vele wensen en er zijn minstens zoveel verschillende oplossingen. Net als bij boeken zijn de applicaties die we ontwikkelen onder te verdelen in genres en kunnen we ze ontwerpen volgens verschillende architectuurstijlen. Uiteraard heeft iedere stijl zijn specifieke aandachtspunten waar we bij het ontwerp van de software rekening mee dienen te houden.



FIGUUR1. MICROSOFT APPLICATION ARCHITECTURE GUIDE 2ND EDITION.

In dit artikel besteden we aandacht aan de 'Application Architecture Guide 2nd Edition' van Microsoft Patterns & Practices. Deze guide beschrijft de verschillende stijlen en bespreekt de bijbehorende aandachtsgebieden en principes. Architecten en lead developers worden hierdoor ondersteund bij het ontwerpen van een gedegen architectuur voor applicaties voor het .NET platform. De guide helpt vragen te beantwoorden die architecten bezig houden. Vragen als: Met welke technologie kunnen we voldoen aan de eisen die gesteld worden aan de te ontwikkelen soft-

ware? Hoe passen we de technologie op de juiste manier toe? Maken we een Windows-applicatie? Voldoet een Service Oriënted Architecture? Kiezen we voor Silverlight of toch maar 'gewoon' voor Windows Presentation Foundation?
Tegenover de diversiteit aan technologieën en oplossingen vraagt de markt / klant steeds meer naar standaardisatie. Standaardisatie van softwareontwikkeling en softwarearchitectuur. Veelal om kennisoverdracht eenvoudiger te maken en de kwaliteit van het opgeleverde product te verhogen. Zeker in deze tijden is er minder tijd om te experimenteren en zijn er goedkopere oplossingen nodig. Het is niet voor niets dat Application Lifecycle Management steeds populairder wordt. Naast een voorspelbaar ontwikkelproces dienen we ervoor te zorgen dat onze software zo veel mogelijk waarde toevoegt voor de eindgebruiker, mee kan gaan met veranderende businessbehoeften en dat alles tegen zo laag mogelijke operationele kosten.
Als architect moeten we dus continue zoeken naar de balans tussen requirements en behoeften van de business aan de ene kant en de mogelijkheden van de infrastructuur, technologie en tools aan de andere kant. Dit is zeker geen eenvoudige klus en daarom is het goed dat de Microsoft Application Architecture Guide 2nd Edition ons helpt bij het kiezen van de juiste technologie, architectuur, patronen en gangbare architectuur principes.

Application Architecture Guide

De eerste versie van de 'Application Architecture Guidance' die Microsoft Patterns & Practices opleverde, stamt uit 2002. Het stenentijdperk als we in ICT jaren denken. Het .NET Framework was nog in versie 1 en het doelplatform van de guidance was Windows 2000. Er was nog geen sprake van Windows Communication Foundation, Windows Workflow, Windows Presentation Foundation, Silverlight, Composite Applications, Rich Internet Applications, Web 2.0, Cloud Computing, SaaS en alles wat er nog meer is uitgevonden in de afgelopen jaren. Het boeiende is echter, dat er voor wat betreft ontwerp- en architectuurprincipes niet veel veranderd is ten opzichte van 2002. Eigenlijk kun je zeggen dat de richtlijnen uit 2002 zich in de afgelopen jaren meerdere malen bewezen hebben en in de periode daarna alleen nog maar verder verfijnd zijn.

De architectuur van een Rich Client applicatie ziet er anders uit dan die van een Mobile Web applicatie.

De 2nd Edition van de guide heeft een brede focus. In vergelijking tot zijn voorganger bespreekt de guide meer applicatietypes, technologieën, genres en architectuurstijlen. Naast deze stijlen worden ook de niet-functionele eisen (de zogenaamde 'quality attributes' en 'cross cutting concerns'), die mede invloed hebben op de architectuur van de applicatie, besproken.
Eén van de onderwerpen die uitgebreid behandeld wordt in de guide zijn architectuurstijlen.

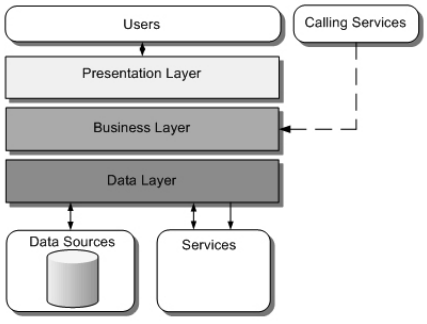
De bekendste daarvan zijn:
- 3-Tier / n-Tier
- Client-Server
- Component-Based Architecture
- Domain Model
- Layered Architecture
- Message-Object-Oriented
- Separated Presentation
- Service-Oriented Architecture (SOA)

De guide behandelt van elke architectuur stijl de meest kenmerkende patronen en beschrijft de bijbehorende applicatietypes (mobile applicatie, services, enzovoorts.)

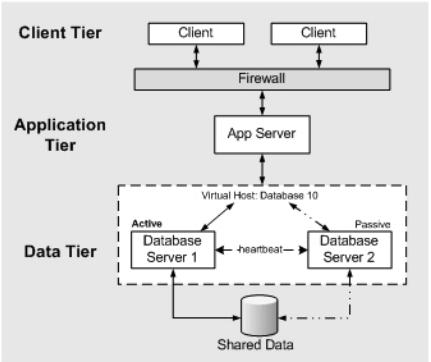
Layers, Components en Tiers

Als we starten met het ontwerpen van onze applicatie beginnen we op het hoogste niveau. Veel voorkomende vragen hierbij zijn: hoe groepeer ik de functionaliteit en hoe verdeel ik dit over de verschillende lagen? Met andere woorden; Hoe voldoe ik aan de 'key design principles'?
De bekendste onderverdeling in lagen is wel het drie lagen model van figuur 2.

Hoewel de in figuur 2 getoonde applicatiestijl een veel gebruikte is, zijn er nog vele andere applicatiestijlen (archetypes) waarvoor een andere optimale verdeling van functionaliteit over lagen geldt. Zo ziet de architectuur van een Rich Client applicatie er bijvoorbeeld anders uit dan die van een Mobile Web applicatie. De guide beschrijft voor ieder van de applicatiestijlen de meest optimale verdeling van de functionaliteit en verantwoordelijkheden over de verschillende lagen. Zoals uit onderstaande citaat uit de guide blijkt, wordt er niet alleen een 'academische' opsplitsing geven maar wordt ook duidelijk aangeven welke overwegingen er zijn om nuances aan te brengen in je architectuur ontwerp.
In some cases, the presentation layer may communicate with the business layer through the services layer. However, this is not an absolute requirement. If the physical deployment of the application locates the presentation layer and the business layer in the same tier, they may communicate directly.
Naast de logische beschrijving van de applicatie in lagen van functionaliteit dient er natuurlijk ook een fysieke beschrijving gemaakt te worden. Veelal wordt hier het component diagram voor gebruikt. Hierbij wordt de daadwerkelijke onderverdeling van de componenten binnen de lagen beschreven met de daarbij behorende interfaces tussen deze componenten. De Application Architecture Guide besteedt uitgebreid aandacht aan de structuur en ontwerp van de applicatie in componenten. Hierbij natuurlijk rekeninghoudend met de verschillende fysieke 'tiers' waarop de verschillende lagen draaien. In figuur 3 zien we een mogelijke opdeling in tiers



FIGUUR2. LAYER DIAGRAM.



FIGUUR3. TIERS.

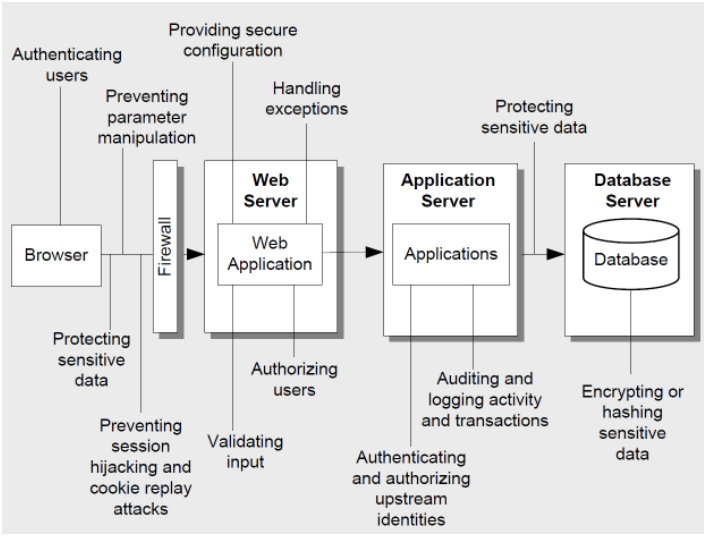
zoals deze in de guide in detail beschreven wordt.

De Application Architecture Guide behandelt decross cutting concerns, de quality attributes , de lagen, componenten en tiers en geeft daarmee een compleet beeld van de aandachtsgebieden die bestudeerd dienen te worden bij het ontwerpen van applicaties.

Architectuur aanpak

Het ontwerpen van een geschikte architectuur is niet eenvoudig. Dit komt enerzijds door de overvloed aan technologieën, maar wordt anderzijds veroorzaakt door het ontbreken van een eenduidige aanpak voor het ontwerpen van een architectuur. Om deze reden beschrijft de Application Architecture Guide 2nd Edition een architectuur aanpak die ons richting geeft bij de stappen die we moeten uitvoeren om tot een geschikte architectuur te komen. Zoals we in figuur 2 kunnen zien bestaat deze aanpak uit een vijftal stappen. In dit artikel hebben we te weinig ruimte om alle stappen in detail te behandelen maar toch willen we even kort de door de stappen heen lopen.

1. Bepalen van architectuur doelstellingen.
In deze stap van de architectuuraanpak bepalen we het doel van de architectuur die we gaan ontwerpen. Met andere woorden, wat willen we met de architectuur bereiken? Hoewel de activiteiten binnen deze stap in feite niets te maken hebben met 'architectuurontwerp' is de uitkomst van deze stap van essentieel belang voor de vervolgstappen in de architectuuraanpak. Zo bepalen we in deze stap bijvoorbeeld of we de architectuurontwerpen voor een prototype of een productiesysteem. Welke technologische middelen we tot onze beschikking hebben en welke deploymenteisen er aan de architectuur gesteld worden. Je zult begrijpen dat antwoorden op dergelijke vragen impact hebben op de tijd die je besteedt in de vervolgstappen van de architectuuraanpak en de diepgang waarin je de architectuur uitwerkt en test.



FIGUUR 5. OVERZICHT VAN EEN AANTAL VEELVOORKOMENDE 'HOTSPOTS'.

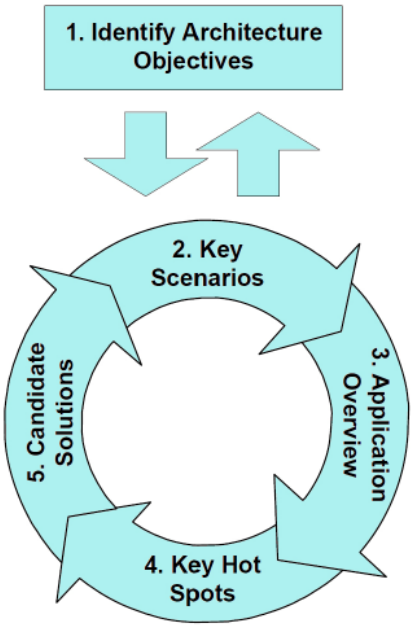
2. Bepalen van de key scenarios.
In deze stap van de architectuuraanpak bepalen we de belangrijkste scenario's voor de applicatie die we moeten ontwerpen. Een scenario kan belangrijk zijn vanuit het oogpunt van architectuur, maar ook zeker vanuit het oogpunt van gebruikersfunctionaliteit. Denk hierbij bijvoorbeeld aan de manier waarop security werkt en de manier waarop de gebruiker hiermee geconfronteerd wordt.
Het bepalen van deze key scenario's helpt ons bij het identificeren van belangrijke eisen die gesteld worden aan de architectuur die we ontwerpen en stelt ons, in een latere stap, in staat de ontworpen architectuur te valideren aan de hand van deze scenario's.

3. Creëer een overview van de applicatie
In deze stap proberen we een zo goed mogelijk beeld te krijgen van de applicatie die we gaan ontwikkelen. Dit helpt ons ervoor te zorgen dat de architectuur die we gaan ontwerpen ook daadwerkelijk aansluit bij de te ontwikkelen applicatie. Vragen waarop we in deze stap van de architectuuraanpak antwoorden proberen te krijgen zijn bijvoorbeeld: Wat voor soort applicatie bouwen we? Bouwen we een mobile applicatie of een service? Op welke infrastructuur moeten we straks deployen? Welke eisen worden er gesteld aan security, Welke technologie gebruiken we?

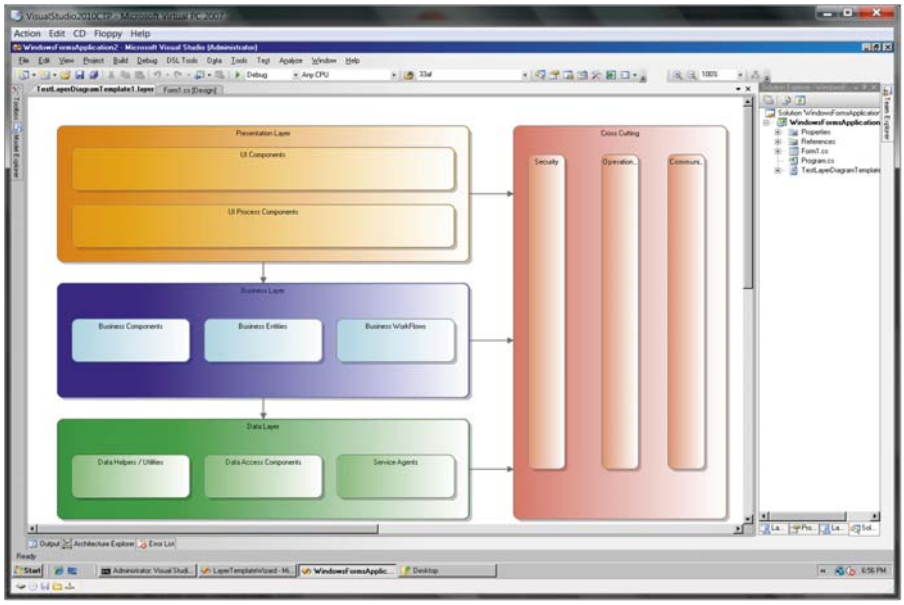
Aan het einde van deze stap hebben we een goed beeld van de applicatie die we willen bouwen, hebben we de belangrijkste eisen die gesteld worden aan de architectuur in kaart gebracht en hebben we de genomen (architectuur) beslissingen vastgelegd.

4. Bepaal de key hotspots
Binnen deze stap bepalen we de plaatsen binnen de applicatie waar we de grootste uitdagingen verwachten. In de Application Architecture Guide worden dit ook wel hotspots genoemd. Vaak zijn dit de aandachtsgebieden waarin kwaliteitseisen gesteld worden zoals: snelheid van gebruik, flexibiliteit en dergelijke. Maar ook uitdagingen, of beter gezegd risico's, op het gebied van de zogenoemde ' cross cutting concerns' die binnen ieder onderdeel van de applicatie gelden. Een veel voorkomende cross cutting concern is bijvoorbeeld: Exception Management. Hoe worden fouten opgevangen en afgewerkt binnen de applicatie en hoe worden ze kenbaar gemaakt aan de beheerder of gebruiker. Andere bekende cross cutting concerns zijn 'logging' en 'instrumentation'. Vragen die hierbij spelen zijn: wat wordt er vastgelegd tijdens het gebruik van de applicatie en hoe wordt de applicatie geconfigureerd? Dit zijn allemaal 'hotspots' die de oplossingsrichting van de applicatie beïnvloeden en zijn daarmee belangrijke onderdelen die vooraf goed over nagedacht dient te worden.

De Application Architecture Guide bevat een overzicht van de meest voorkomende 'hotspots' en de bijbehorende aandachtspunten. Bijvoorbeeld het onderstaande is een uitwerking van de hotspot ' testability':



FIGUUR4. STAPPEN BINNEN DE ARCHITECTUUR AANPAK.



FIGUUR6. LAYER DIAGRAM MET VALIDATIE BINNEN VSTA 2010.

Testability is a measure of how well system or components allow you to create test criteria and execute tests to determine if the criteria are met. Testability allows faults in a system to be isolated in a timely and effective manner.

5. Kandidaat oplossingen (architecturen)

Nu we alle belangrijke scenario's, technische eisen en hotspots boven tafel hebben, kunnen we in deze stap de 'high level architectuur' gaan bepalen en daarna de verdere details in vullen. In deze stap zullen we met behulp van (kleine) prototypen de toekomstige architectuur testen aan de hand van de key scenario's die we in stap 2 van de architectuur aanpak bepaald hebben. Stap 2 tot en met 5 van de architectuuraanpak is een iteratief proces om uiteindelijk tot de geschikte architectuur te komen.

Application Architecture Guide in de praktijk

In de huidige versie is de guide bijna 400 pagina's dik. Het ligt **daarom** voor de hand dat niet iedereen de guide in één ruk van begin tot einde uitleest. Gelukkig is de guide zo opgedeeld dat deze ook als naslagwerk gebruikt kan worden. Hiermee wordt het dus mogelijk, slechts die gedeeltes van de guide te lezen die op dat moment relevant zijn. Toch blijft het in de praktijk een uitdaging om er voor te zorgen dat de informatie uit de guide ook daadwerkelijk gebruikt wordt door de architecten en lead developers binnen onze organisatie.

Uit het verleden is gebleken dat veel bedrijven hun eigen '.NET ontwikkeling richtlijnen' schrijven. Veelal zijn deze richtlijnen een selectie uit documenten als de Application Architecture Guide en worden deze richtlijnen aangevuld met wat bedrijfsspecifieke standaarden en variaties. Ook hiervoor geldt dat we overtuigingskracht nodig hebben om dergelijke richtlijnen goed ingebed te krijgen binnen de organisatie.

Controle op de naleving van de richtlijnen gaan meestal niet veel verder dan controles op het gebruik van naamgevingconventies binnen de source code. Controle op naleving van architectuur richtlijnen zien we een stuk minder vaak.

Eén van de mogelijkheden die we hebben om dergelijke 'geschreven' guidance echt van nut te laten zijn, is het inbedden van deze guidance in de tools die architecten, designers en (lead) developers dagelijks gebruiken bij het ontwikkelen van hun software. Gelukkig zijn er een aantal ontwikkelingen in die richting gaande. De volgende versie van Visual Studio Team System [2010] zal bijvoorbeeld UML gaan ondersteunen voor het ontwerpen van applicaties. Naast de meest gangbare UML diagrammen zal er 'out of the box' ook een Layer Diagram beschikbaar zijn. Dit diagram biedt naast modelleerfunctionaliteit ook de mogelijkheid om de implementaties (in code) te valideren tegen vooraf gedefinieerde richtlijnen. Zo kunnen we bijvoorbeeld valideren of er voldaan is aan de functionele opdeling van de applicatie zoals deze bedoeld is door de architect. Ook bieden deze diagrammen, door de rijke extensibility infrastructuur, vele mogelijkheden om guidance zoals deze beschreven wordt in de Application Architecture Guide beschikbaar te stellen binnen de Visual Studio IDE.

Conclusie

Het zal duidelijk zijn dat het in een artikel als dit niet mogelijk is om de complete inhoud van de Application Architecture Guide 2nd Edition te beschrijven. Bovenstaande geeft wel aan dat deze guide de moeite van het bestuderen waard is. De informatie in deze guide en de voorgestelde architectuuraanpak zullen zeker helpen bij het ontwerpen van solide architecturen. Wel dienen we ons te realiseren dat we een manier zullen moeten vinden waarop we de richtlijnen en aanbevelingen uit de guide kunnen integreren in onze softwareontwikkeling activiteiten. De kennis over deze richtlijnen dient gedeeld te worden binnen het team en daar waar mogelijk zullen we de ontworpen architectuur moeten valideren aan deze richtlijnen.

Links

AppArchGuide 2nd: http://www.codeplex.com/AppArchGuide

App Arch Guide 2.0 Knowledge Base: http://apparch.codeplex.com/

Blog Edward Bakker: www.EdwardBakker.nl

Blog Clemens Reijnen: www.ClemensReijnen.nl

De auteurs van dit artikel werken op dit moment aan een MSDN whitepaper waarin een architectuuraanpak beschreven wordt die de richtlijnen uit de Application Architecture Guide 2nd Edition combineert met de mogelijkheden van Visual Studio Team Architect 2010 (VSTA). Dit whitepaper beschrijft ook hoe we de extensibility mogelijkheden van VSTA kunnen gebruiken om bijvoorbeeld validaties uit te voeren op de ontworpen architectuur. Meer informatie over dit onderwerp en publicatiedatum van het whitepaper kun je vinden op de hiernaast genoemde blogs.

Edward Bakker, is werkzaam bij Inter Access, is MVP Solution Architect en is bereikbaar via edward.bakker@interaccess.nl of zijn blog.

Clemens Reijnen, is werkzaam bij Sogeti Nederland en MVP Team System.