whitehorses

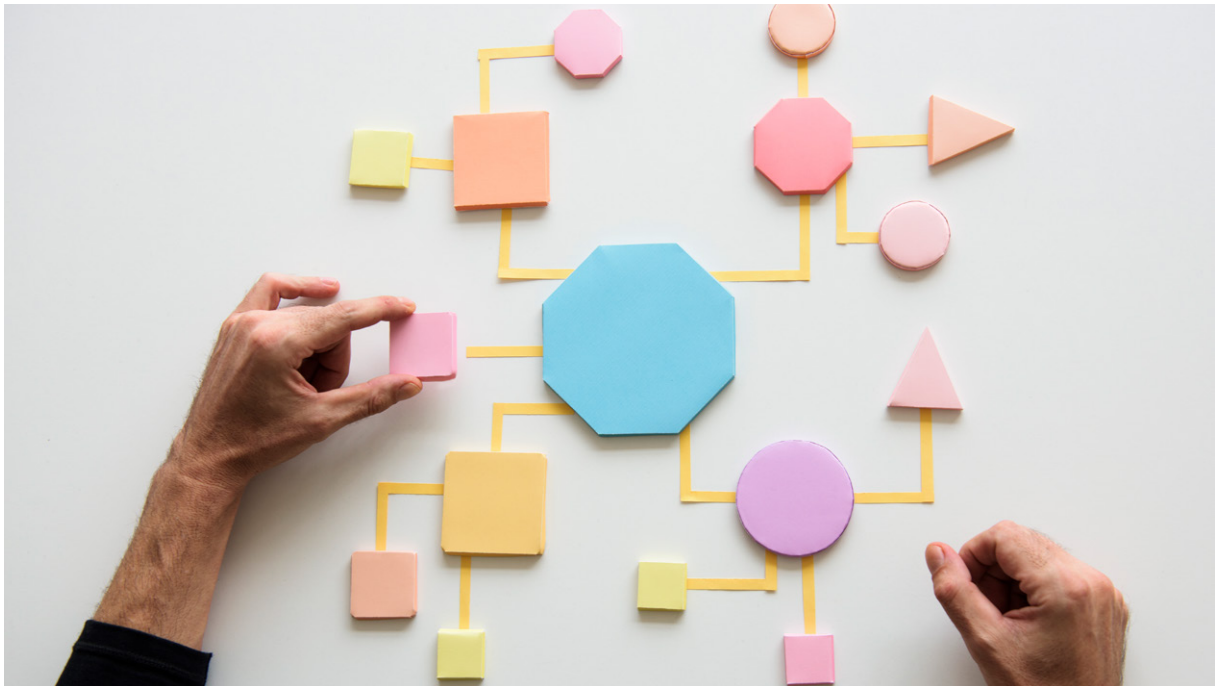
member of the human network

WHITEBOOK

Integratie van OutSystems met TOGAF

Januari 2020

Auteur:
Monique de Vos
INTEGRATIESPECIALIST



Inleiding

OutSystems begon met de gedachte, ‘Wat als we veranderingen tijdens een software ontwikkelingsproject goedkoop en snel maken, in plaats van alles juist te moeten hebben voordat we beginnen?’. Zodoende hebben mensen bij OutSystems de anatomie van verandering bestudeerd en op die kennis het OutSystems Platform gebouwd, met de visie applicaties snel, robuust en goedkoop te houden op elk punt in de applicatie lifecycle. Hoe? De abstractie die het OutSystems platform biedt, zowel in de ontwikkelprocessen als in variabelen als platforms, integratie en lifecycle management, maakt het platform zeer stabiel waardoor applicaties eerder kunnen worden opgeleverd en integraal kunnen worden beheerd. De visie van OutSystems lijkt te werken; in een enquête onder 200 OutSystems gebruikers levert 75% applicaties op in minder dan drie maanden.

Met een complete visie als die van OutSystems is de mate van naleving ervan erg belangrijk om daadwerkelijk de resultaten te behalen die de leverancier voorschotelt. De succesvolle en geteste principes van OutSystems verwateren snel met de introductie van andere frameworks of bestaande organisatie-specifieke principes. Toch is dit onvermijdelijk; tenzij een compleet nieuwe IT-tak (of compleet nieuw bedrijf) wordt opgericht met het specifieke doel om met OutSystems te gaan werken, zullen er altijd andere bestaande werkwijzen en principes heersen in de bedrijfscultuur waar OutSystems mee zal moeten samenwerken. Een

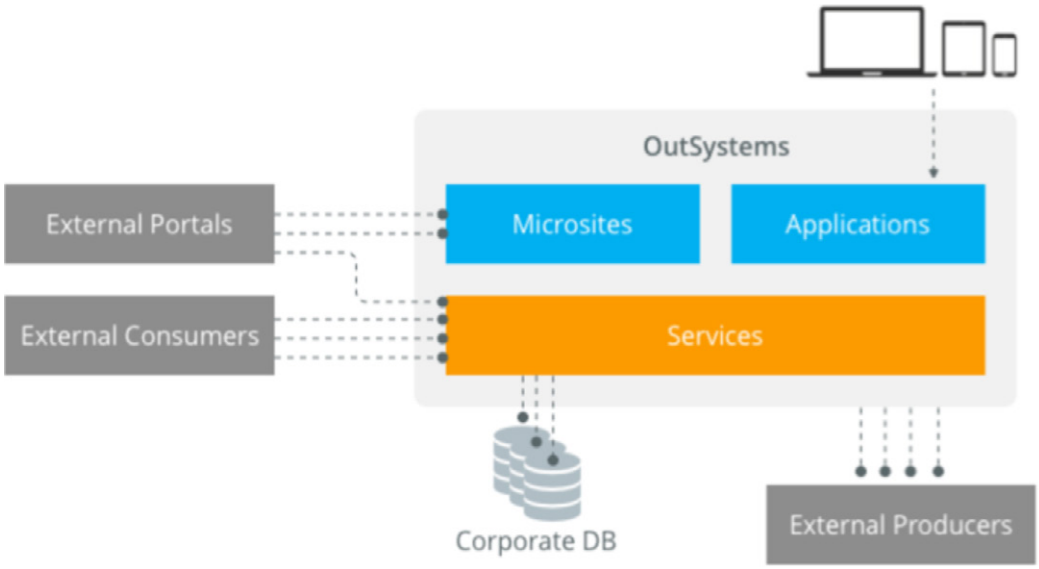
van deze werkwijzen is TOGAF, een zeer populaire methode voor het ontwikkelen en beheren van de Enterprise architectuur.

Dit Whitebook is een theoretische uiteenzetting over de benadering van OutSystems en de integratie van OutSystems met TOGAF.

Een introductie van de OutSystems-benadering van architectuur

Hoewel het hier ook voor wordt gebruikt, is OutSystems niet per sé een voorstander van het compleet vervangen van bestaande bedrijfsapplicaties. Wanneer je OutSystems vraagt wat je dan wél moet vervangen, krijg je de wedervraag, 'Welk deel van je bestaande systemen verandert veel?'. Hoe meer de *requirements* van een systeem aan verandering onderhevig zijn, zij het door wisselende *business needs*, snelle innovatieprocessen, veranderende regelgeving of nieuwe technologie, hoe groter de business case wordt om deze te vervangen door OutSystems. OutSystems is zich bewust van de kosten en potentiële risico's die complete vervanging van een systeem met zich meebrengt, en door de uitgebreide integratiemogelijkheden van het platform is het geen enkel probleem om op oudere systemen aan te sluiten.

Als gevolg hiervan ziet een standaard OutSystems-applicatie er zo uit in relatie met het applicatie-landschap:

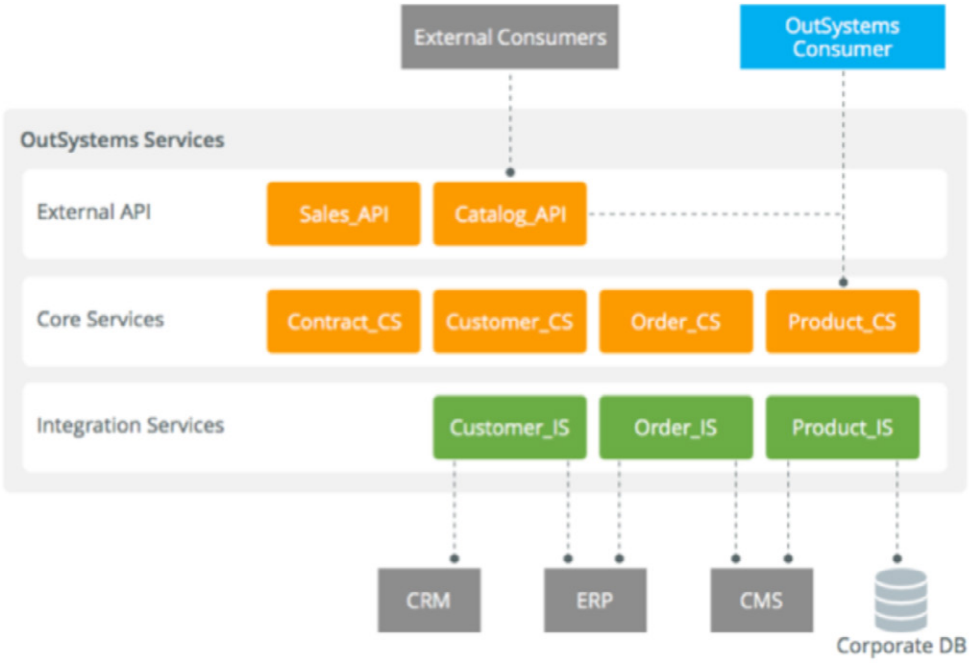


I. De typische positie van een OutSystems applicatie in een enterprise ecosysteem. ©OutSystems, 2019

OutSystems applicaties sluiten aan op andere systemen via APIs (bijv. REST/SOAP) of ingebouwde (database-)drivers (bijv. Oracle, MSSQL). Ook kunnen ontwikkelaars eenvoudig hun eigen/oude (C#, Java, HTML, CSS, JavaScript) code implementeren in de nieuwe applicatie voor specifieke of exotische functies, waardoor de nieuwe implementatie eenvoudig aansluit op de voorgaande iteraties.

OutSystems heeft een sterke voorkeur voor SOA (Service Oriented Architecture) en biedt een framework voor de succesvolle implementatie ervan. De voorgestelde SOA promoot service distribution over drie niveaus: externe APIs, kernservices, en integratieservices. Deze services kunnen *tightly coupled* zijn om het volledige potentieel van OutSystems te benutten in situaties waarin totale onafhankelijkheid van services geen voorwaarde is, of *loosely coupled*/volledig *decoupled* voor het hanteren van een microservices-strategie en het bevorderen van CI/CD. De externe APIs kunnen gezien worden als technische wrapper om een kernservice. De integratieservices zijn technische wrappers om externe data te abstraheren en te normaliseren voor gebruik door de kernservices. Specifieke integratie patronen zijn beschikbaar om het ontwerp ervan te ondersteunen[2].

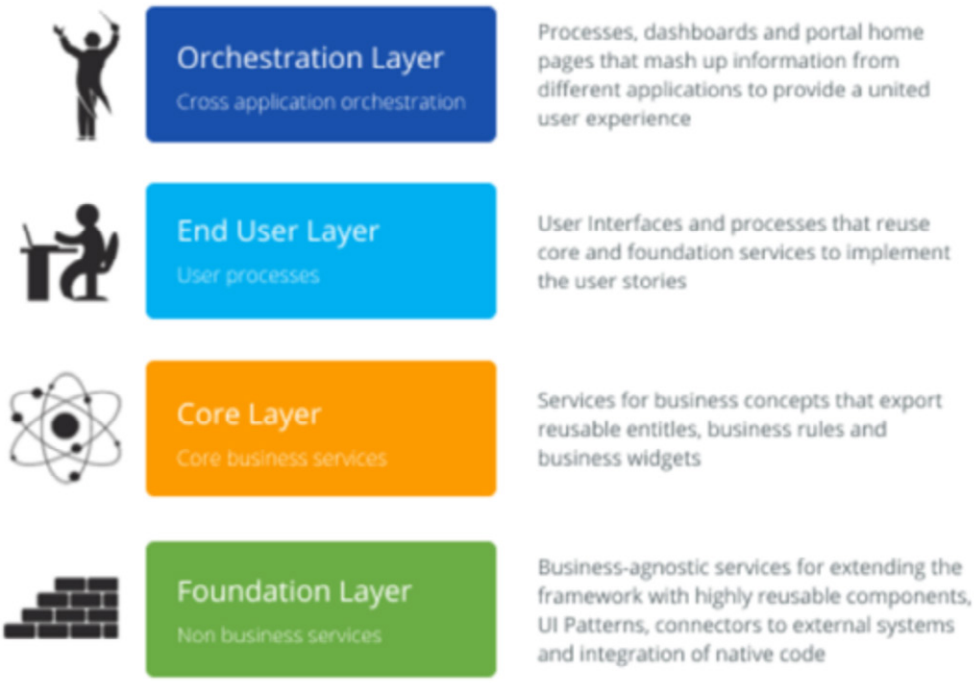
De services worden geplaatst in modules per business concept om business-specifieke lifecycles te kunnen hanteren en de kwestie van eigenaarschap te vergemakkelijken.
De blokken in onderstaand diagram vertegenwoordigen deze gegroepeerde services als modules.



II - Drie niveaus van service distributie: Externe APIs, kernservices, en integratieservices. ©OutSystems, 2019

Om de ontwikkeling van SOA makkelijk te maken op het niveau van een Solution Architecture, heeft OutSystems het Four Layer Canvas ontwikkeld (4LC). Hierin worden de correcte abstractieniveaus van herbruikbare services en de correcte isolatie van functionele modules gedefinieerd en zowel visueel als geautomatiseerd controleerbaar gemaakt. Ontwikkelaars dienen kennis te hebben van het 4LC en kunnen dit inzetten in een project vanaf het Professional certificatieniveau. OutSystems Tech Leads hebben kennis van het complete OutSystems framework inclusief het 4LC en kunnen inspringen als Solution Architect wanneer nodig.

Het 4LC definieert vier lagen met verschillende functies:



III - De niveaus van het Four Layer Canvas (4LC). ©OutSystems, 2019

De onderste laag in het 4LC is de Funderingslaag (Foundation Layer). De Foundation Layer bestaat uit zeer specifieke elementen, non-functional en integration requirements, die herbruikbaar zijn voor alle applicaties; connectoren naar externe systemen, integratie van externe (native) code en zeer generieke UI-patronen. De laag er boven is de Kernlaag (Core Layer). De Core Layer focust zich voornamelijk op herbruikbare businessconcepten, zoals businessprocessen, business rules en business-specifieke UI widgets. Om deze processen te realiseren maakt de Core Layer gebruik van de modules uit de Foundation Layer en voegt daar zijn eigen logica aan toe. Sommige Core Modules staan volledig op zichzelf, bijvoorbeeld wanneer Core Modules hun eigen, OutSystems-native data-entiteiten onderhouden.

In de Eindgebruikerslaag (End User Layer) komt alles samen; hier vormt zich de applicatie zoals de eindgebruiker hem ziet. Hier worden applicatie-specifieke User Interfaces en processen gebouwd die gebruikmaken van de Core Services om de User Stories te implementeren. Uiteindelijk trekt de bovenste laag, de Orchestratielaag (Orchestration Layer), informatie uit verschillende End User applicaties en voegt deze samen voor gebruik door portalen, dashboards en cross-applicatie processen.

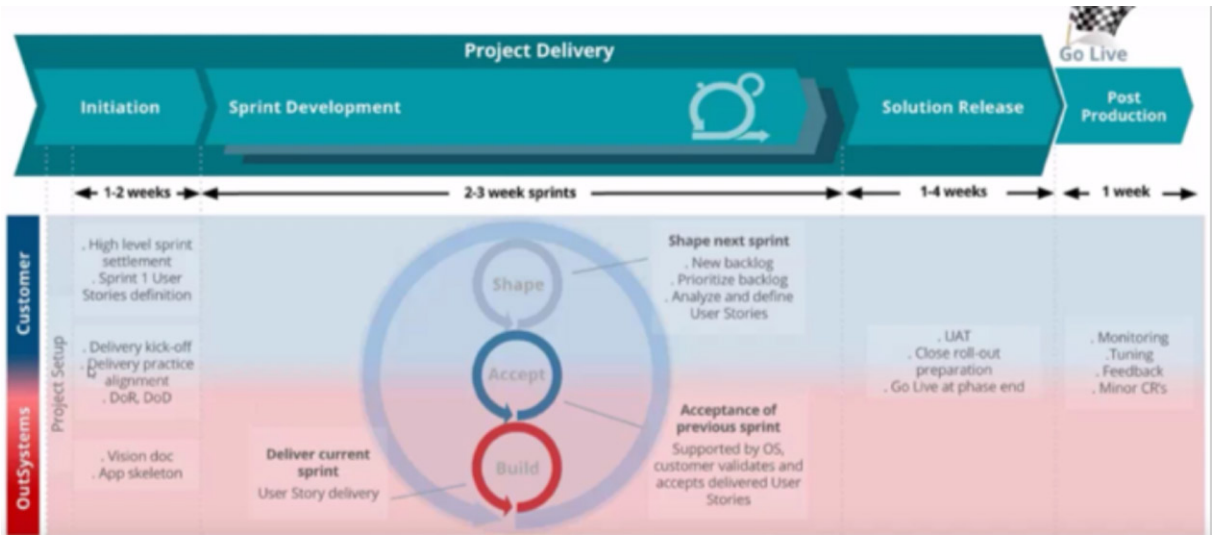
Binnen deze lagen blijven de modules gescheiden per concept. Na het identificeren van de verschillende concepten, worden de bijbehorende modules gedefinieerd om de concepten te implementeren. Een voorbeeld van het identificeren van concepten is weergegeven in het diagram hieronder.

IV - Het identificeren en scheiden van concepten in het Four Layer Canvas (4LC). ©OutSystems, 2019

OutSystems, Emergent Design en Agile

Voor het identificeren en definiëren van concepten is OutSystems fan van *emergent design*. *Emergent design* is het uitkijken naar opkomende concepten tijdens het bouwen van een applicatie en deze direct verwerken in het design van de applicatie wanneer ze worden gezien. Een goed voorbeeld van *emergent design* is het leggen van natuurlijke paden tussen gebouwen; lever een universiteitscampus op zonder aangelegde paden, kijk naar de natuurlijke paden die men loopt (die vaak de vorm krijgen van een zandpad over tijd) en asfalteer die. De onderliggende gedachte is dat wij nooit perfect vooraf kunnen bedenken hoe een gebruiker zich gedraagt, of wat de echte behoeften zijn van de business, et cetera, en daarom beter stapsgewijs en *just-in-time* kunnen ontwerpen in plaats van het hele ontwerp vooraf op te leveren. Zo is er altijd voldoende ruimte en flexibiliteit voor grote veranderingen in het ontwerp.

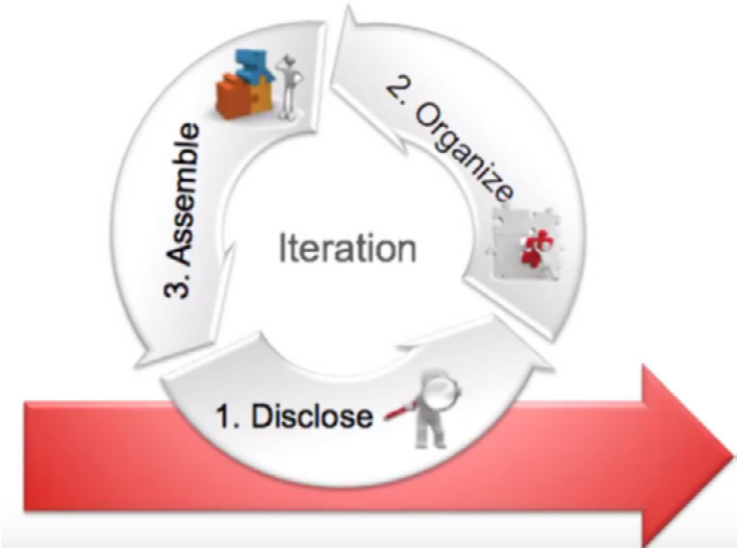
Om een succesvolle *emergent design* strategie te kunnen uitvoeren heeft OutSystems een eigen vorm van Agile gecreëerd. Gezien het platform en de Agile methodologie in hetzelfde jaar zijn ontstaan, heeft OutSystems de ontwikkelingen in Agile weliswaar op de voet gevolgd, maar heeft het bedrijf niet kunnen profiteren van een adaptie van de nieuwere varianten van Agile, zoals LeSS of SAFe. Dit resulteert in een originele en redelijk solide toepassing van Agile die niet één op één te vergelijken valt met een ander Agile framework, maar die wel heel compatibel is met Agile-gebaseerde frameworks.



V - OutSystems Agile project planning. ©OutSystems, 2019

Een project start met een *Initiation Workshop* gedurende een hele week, waar alle (alle!) mensen die aan het project werken aan meedoen, van de Project Manager tot de Key User. Binnen de workshop wordt gezorgd dat het project soepel kan beginnen; alle briefings worden gedaan, de visie en business context worden verfijnd en gedeeld, de User Stories en Epics die al bekend zijn worden op de backlog geplaatst, de eerste UI mockups worden gepresenteerd, data wordt gemigreerd en de juiste personen krijgen toegang tot de juiste omgevingen. Daarna start de ontwikkelfase in sprints van twee à drie weken. Binnen een sprint zijn drie 'sub-sprints' aan de gang; *Shape*, *Accept* en *Build* (Vorm, Accepteer en Bouw). De Accepteer- en Bouw-sprintcycli houden zich bezig met het testen en ontwikkelen van user stories. De Vorm-sprint houdt zich bezig met het definiëren van User Stories en het steeds opnieuw prioriteren van de backlog. Hier is het principe van *emergent design* dominant aanwezig; User Stories worden geschreven (en herschreven) naast de 'standaard'-sprints in plaats van ervan uit te gaan dat vooraf alle User Stories definitief bekend zijn.

Voor architectuur binnen OutSystems geldt hetzelfde principe; we weten nog niet hoe het eindproduct er uit gaat zien voordat we grondig hebben getest met onze key users of feedback hebben ontvangen naar aanleiding van de oplevering van ons *Minimum Viable* Product (MVP), dus de architectuur van de met OutSystems gebouwde oplossing is ook tijdens de ontwikkelfase aan verandering onderhevig. Ook architectuurontwerp wordt iteratief ingestoken in drie stappen:



VI - Drie stappen voor iteratieve architectuur. ©OutSystems, 2019

Disclose (Ontdek) de nieuwe concepten die toegevoegd moeten worden en de veranderingen die gemaakt moeten worden; *Organize* (Organiseer) de nieuwe veranderingen per concept op het canvas; *Assemble* (Samenbrengen) van verschillende concepten - of juist het isoleren van herbruikbare logica. Waar precies deze iteratie moet plaatsvinden wordt niet verteld, maar het is aannemelijk dat het de bedoeling is dat ook deze stappen per sprint doorlopen worden door de Tech Lead, de architect of een senior ontwikkelaar van het ontwikkelteam.

Met dit framework weet OutSystems een stabiel proces op te zetten op projectteam-niveau. Een hoop projecten worden ook daadwerkelijk op dit niveau uitgevoerd; bijna onafhankelijke projecten die rond de drie maanden looptijd hebben in totaal. OutSystems als platform is echter ook zeer geschikt voor grote bedrijven met meerdere applicaties in het portfolio. Het inbrengen van een uitgebreide visie op projectvoering als die van OutSystems in een bestaand bedrijf kan echter lastig zijn en het niet volgen van de richtlijnen van OutSystems kan ten koste gaan van de optimale snelheid die het platform belooft in zowel de performance als de ontwikkelduur van de applicaties.

TOGAF

TOGAF is een framework voor Enterprise architectuur. Het is een van de antwoorden op de vraag, ‘als we consistent en succesvol architectuur willen gebruiken in een potentieel groot bedrijf met een mogelijk breed portfolio aan software, hoe pakken we dat aan?’ Naast TOGAF bestaan er nog andere Enterprise architectuur frameworks, bijvoorbeeld Zachman. TOGAF wordt gebruikt in meer dan 80% van de grote internationale bedrijven, en kennis van TOGAF is dan ook essentieel voor degenen die zich verder willen verdiepen in (Enterprise) architectuur.

TOGAF is eerder een aanpasbare gereedschapskist van breed inzetbare *tools* dan een nauw op te volgen methode. Het geeft gestructureerd advies gebaseerd op de ervaring van zo’n 200 Enterprise-architecten, en hoewel dat advies het best serieus genomen kan worden, weten deze architecten ook heel goed dat een ‘*one-size-fits-all solution*’ zonder afwijking toepassen in de dynamische bedrijfswereld niet leidt tot het doel van TOGAF, namelijk consistent succesvol architectuur voeren. De invoering van TOGAF in een bedrijf begint dan ook met de vraag, ‘hoe gaan we het zo aanpakken dat onze methode past bij dit bedrijf?’. Pas na die vraag beantwoord te hebben begint het project; dit heet dan ook de *Preliminary Phase*.

TOGAF structureert zichzelf rond een iteratief proces, de *Architecture Development Method* (ADM). Dit proces gaat door fases, gelabeld met de letters A tot en met H, waarin de architectuur van een of meerdere oplossingen vorm krijgen, worden uitgevoerd en worden onderhouden. Fase A begint het project door zich af te vragen wat er gedaan moet worden, wie de stakeholders zijn (en hoe er het best contact met ze kan worden gehouden) en hoe de ideale oplossing er uit zou moeten zien in grote lijn. Aan het eind van fase A wordt een contract getekend met de sponsor en in fase B tot en met F wordt dit plan tot in de details uitgewerkt. Daarna wordt een nieuw contract getekend, dit keer met de ontwikkelaars erbij die het plan gaan uitvoeren in fase G, en in fase H wordt het product onderhouden. In fases G en H wordt controle gehouden over de uitvoering (als de uitvoering het zorgvuldig goedgekeurde plan niet volgt heeft iedereen een probleem) en worden kleine aanpassingen gemaakt waar nodig. Dit betekent dat er een significant deel van het plan klaarligt wanneer men begint met bouwen. Is er een aanpassing nodig in een van de 8 fases, dan kan er teruggegaan worden naar elke fase die men nodig acht om het probleem op te lossen.

Naast een voorstel voor het algemene architectuurproces (de ADM) geeft TOGAF ook een handleiding om op een transparante en verantwoordelijke manier met alle omliggende verantwoordelijkheid en documentatie om te gaan. Er wordt veel nadruk gelegd op de toegevoegde waarde van gestructureerde documentatie, contracten, steun vanuit de stakeholders en hoger management en naleving van de gemaakte plannen. Het proces van het managen van de uitvoering en van de architecten zelf wordt *Governance* genoemd.

TOGAF en OutSystems

Wanneer er binnen een architecturaal volwassen organisatie die werkt volgens het TOGAF framework wordt gekozen voor OutSystems is er soms al een hoop werk gedaan op architecturaal gebied. Veel projecten kiezen er gelukkig voor om de keuze voor een technologie of platform al eerder te maken binnen het proces, waardoor er al eerder in het proces ruimte ontstaat om rekening te houden met een framework als OutSystems.
Een aantal onderdelen van de implementatie van OutSystems in een TOGAF project zijn niet direct duidelijk. Hoe zit het met *emergent design* in TOGAF? OutSystems beschrijft een strenge vorm van Agile, is daar plek voor in het Governance Framework? Welke tools biedt OutSystems die absoluut gebruikt moeten worden voor een succesvol project?

(Solution) Architectuur in de taal van OutSystems

Het Four Layer Canvas (4LC) van OutSystems (zie hoofdstuk '*Een introductie van de OutSystems-benadering van architectuur*') is een tool die niet alleen uitgebreid ondersteund wordt door OutSystems door middel van o.a. geautomatiseerde verificatie en die begrepen wordt door alle OutSystems experts, maar het is ook een tool die op een intuïtieve manier vertaalt naar de optimale manier om OutSystems te gebruiken. Architectuur met OutSystems resulteert in een aantal nieuwe diagrammen op Applicatie-niveau. Hierdoor wordt het wenselijk OutSystems voor te stellen in fase C of opnieuw te kijken naar de gemaakte plannen in face C, het definiëren van de data- en applicatiestructuur.

De Initiation Workshop en de toevoeging van UX

De Initiation Workshop is een één tot twee weken durend multidisciplinair programma waarin gezorgd wordt voor een ongehinderde, vlotte start van een project. Gedurende dit programma wordt de visie vastgelegd en de business context gedeeld, wordt er gezorgd voor de eerste bruikbare user stories en worden high-level *epics* gedefinieerd. De infrastructuur wordt voorbereid, de eerste UI wireframes en hi-fi mockups worden getoond en data wordt gemigreerd waar nodig. Dit proces zorgt niet alleen voor een vlotte start van het project, maar ook voor een algemene consensus tussen alle betrokkenen over wat er uitgevoerd gaat worden en hoe dat wordt gedaan. Daarom is het eind van de Initiation Workshop een perfect moment om een Architectuurcontract op te stellen en te tekenen (mits het een Agile contract betreft – zie '*Aanpassingen aan TOGAF – Governance en de rol van architect*'.)

Tijdens de Initiation Workshop worden ook gebruikersrollen en *Persona's* gecreëerd (of getoond). Het gebruik van persona's is zeer aan te raden, maar hoeft niet te wachten tot de Initiation Workshop in een TOGAF context. Het gebruik van persona's is zeer nuttig in TOGAF ADM Fase A en B, waar de business case wordt gemaakt en uitgediept; persona's verdiepen het begrip van een actor en kunnen dienen als hulpmiddelen om empathie toe te voegen aan business cases en aannames over actoren empirisch te toetsen. Persona's zijn een specialiteit van UX en designers en worden vaak gebruikt om *value propositions* een gezicht te geven of kracht bij te zetten. Gezien de hoeveelheid focus op UX een project vanaf het begin al moet hebben om te slagen in de markt, lijkt het verstandig om UX actief te betrekken vanaf de eerste fases van het (architectuur)project.

Het is ook zeer verstandig UX te betrekken tijdens de uitvoering van het project; het uitvoeren van user tests in een vroeg stadium is een extra veiligheidsmaatregel voor de investering in het project gezien potentieel desastreuze aannamefouten op tijd uit het project worden gehaald en waardevolle feedback op tijd kan worden verwerkt in het project om extensieve refactoring te voorkomen. OutSystems Agile incorporeert het UX-proces in zowel de Initiation Workshop als in de Sprints. Om OutSystems Agile (en de integratie van UX) optimaal tot zijn recht te laten komen is er een significante flexibiliteit nodig in uitoefening van de Enterprise architectuur.

Aanpassingen aan TOGAF - Principes

Om volledig gebruik te kunnen maken van de flexibiliteit van OutSystems' modulaire aanpak is het hebben van een zeer gedetailleerd plan eerder een hinder dan een hulp om te kunnen beginnen met de uitvoering. Het hele plan kan veranderen op elk moment, en als het hele plan verandert is alle extra inzet die in het plan is gestoken vooraf aan de verandering verspilling van tijd en middelen. Het is een kunst op zich om een minimaal uitvoerbaar plan op te leveren en van daaruit verder te ontwikkelen in nauwe samenwerking met de teams. Het principe dat hiermee samenhangt is '*Just in time, just enough*', en de bestaande set principes zal onder de loep moeten worden genomen om dit principe te integreren op een manier die niet in conflict is met de bestaande principes. Ook het aannemen van additionele aan Agile gerelateerde principes die het '*Just in time, just enough*'-principe ondersteunen kan wenselijk zijn.

Naam:	Just in time, just enough
Statement:	Het enige architecturale werk wat gedaan wordt is het werk wat zowel noodzakelijk als mogelijk is op dit moment.
Rationale:	We kunnen alleen goede beslissingen maken als we genoeg aantoonbare kennis hebben om de essentie van een vraagstuk te begrijpen. Die kennis openbaart zich gaandeweg gedurende het proces; vooraf kunnen we niet alles weten.
Implicaties:	We doen alleen het werk waarvan we zeker weten dat het nodig is, om optimaal en zonder bias het aan verandering onderhevige ontwikkelproces te kunnen ondersteunen zonder architecturale middelen te verspillen.

Andere Agile principes kunnen worden overwogen, maar het 'Just in time'-principe geeft de benodigde manier van denken over architectuur het meest simpel en accuraat weer.

Aanpassingen aan TOGAF – Governance en de rol van architect

Naast de timing van architecturaal werk moet er ook opnieuw gekeken worden naar de positie van de architect tegenover de business en tegenover de ontwikkelteams. De extra zekerheid die TOGAF biedt aan de business door middel van contracten, transparantie en documentatie blijft een sterk punt van TOGAF. Werken met contracten en werken met Agile sluit elkaar niet uit, het is de inhoud van de contracten en de manier van het doen eerbiedigen van de contracten waar anders op ingestoken moet worden. Van Agile-waardige contracten zijn genoeg voorbeelden te vinden op het internet, Agile Governance is vaak een lastiger punt. Samen met het 'Just in time, just enough'-principe wordt Agile *Governance* echter ineens conceptueel simpeler.

Met het 'Just in time, just enough' principe hoeft een ontwikkelteam zich niet langer exact te houden aan een vooraf opgesteld plan. Nieuwe ontwikkelingen kunnen in overleg met het team omgezet worden in architecturale beslissingen in plaats van er volledig buiten. Een architect is een deel van een of meerdere teams en wordt als expert betrokken in zaken die met architectuur te maken hebben. LeSS (een Scaled Agile Framework) heeft hiervoor een goede methode die goed aansluit op OutSystems Agile; *Architecture Workshops*, waarin het team en de architect werken aan het ontwerp van de volgende stap in de architecturale opbouw van het systeem. De rol van de architect is het zorgen voor consistentie door het hele landschap; er moeten immers geen dubbele databases worden gebouwd, of twee verschillende modules met nagenoeg dezelfde functionaliteit. De architect houdt de architecturale visie in het achterhoofd bij het maken van beslissingen en is samen met de product owner verantwoordelijk voor de communicatie met stakeholders volgens het communicatieplan. Na een Architecture Workshop documenteert de architect de beslissingen, doet ze onderzoek naar punten die op tafel zijn blijven liggen, en houdt ze andere architecten op de hoogte van de veranderingen zodat zij de nieuwe situatie mee kunnen nemen in hun eigen beslissingen. Termen als 'compliant, consistent, irrelevant' zijn bij deze aanpak niet meer aan de orde; architecten bepalen in samenwerking met het team de volgende stap in plaats van een voorschrift aan te leveren. Nalevingsonderzoek richt zich in dit geval voornamelijk op zaken als het naleven van (door de organisatie of de teams bepaalde) *best practices* en testpraktijken.

Aanpassingen aan TOGAF – de OutSystems architectuurcyclus

OutSystems schrijft drie stappen voor iteratieve architectuur voor, maar waar past dit concept binnen het hart van TOGAF, de iteratieve *Architecture Developement Method* (ADM)? De OutSystems architectuurcyclus – *Disclose, Organize, Assemble* – lijkt een miniatuurvariant van fases B tot E. De toepasbaarheid van deze mini-architectuurcyclus is echter voornamelijk tijdens de uitvoering van het project, als vervanging van het proces van de *Change Requests*. In plaats van te kijken hoe de aangevraagde aanpassing past in het grotere geheel van de vooraf vastgestelde architectuur kijkt de mini-cyclus in *alle* gevallen grondig naar de structuur van de applicatie en reorganiseert het daar waar nodig. Deze mini-cyclus gaat voornamelijk over het applicatie- en dataniveau. De applicatie van de OutSystems mini-cyclus kan het best geplaatst worden in fases G en H; tijdens sprints, met een vaste frequentie of wanneer nodig, in een Architecture Workshop.

Conclusie

De hier genoemde aanpassingen aan de TOGAF toolkit ondersteunen de werkwijze die OutSystems aanbeveelt om de optimale snelheid en flexibiliteit van het platform te kunnen benutten. Dit is het minimum om het OutSystems platform en de OutSystems specialisten hun werk te kunnen laten doen zoals OutSystems het voor ogen heeft. Voor optimale ondersteuning moet per organisatie gekeken worden welke verdere aanpassingen (in de geest van Agile) passen bij het bedrijf en de bedrijfscultuur; voor sommige organisaties zullen de genoemde aanpassingen al een zeer grote stap zijn.

OutSystems Sources:

By the (Play) Book: The Agile Practice at OutSystems
Project Delivery Playbook
OutSystems Evaluation Guide: Architecture

The Open Group Sources:

The TOGAF® Standard, Version 9.2