Methodology

Het gebeurt nog steeds regelmatig dat een applicatie wordt opgeleverd, zonder dat er in een vroeg stadium de gelegenheid is geweest invloed uit te oefenen op het uitrolschema. Meestal is er helemaal niet aan uitrollen gedacht, met onaangename verrassingen als gevolg. Dat moet dus anders, en dat kan ook. Met een helder uitrolschema kunnen die verrassingen worden voorkomen.

Maak een requirement van het uitrolschema

Voorkom onaangename verrassingen

De uitrolmogelijkheden van een applicatie hangen nauw samen met de architectuur. Het maken van een verkeerde of géén keuze kan desastreuze gevolgen hebben. Hoe ver ook eventuele uitrolperikelen bij aanvang van een project in de toekomst lijken te liggen, het verdient zeer de aanbeveling om daar al in een vroeg stadium voldoende aandacht aan te besteden. Over uitrollen moet daarom steeds in termen van requirements worden nagedacht.

Nog steeds gebeurt het regelmatig dat bij oplevering van een applicatie of een website nauwelijks over een uitrolschema is nagedacht. Het uitrollen gebeurt dan *à l'improviste*. In het voortraject is er nooit duidelijk van een uitrolschema - een document waarin wordt gespecificeerd welke delen van een applicatie op welke manier en met welke frequentie kunnen worden uitgerold - sprake geweest. Pas gaandeweg, als de applicatie al live is en als er al beheer moet worden gepleegd – beginnen de effecten van deze leemte voelbaar te worden. De functionele eigenaar zit ijverig werklijsten te maken van wat er allemaal gefixt en aangepast moet worden. Maar van de IT-ers krijgt hij steeds te horen wat er op het gebied van uitrollen allemaal kán – dat wil zeggen, wat er sóms kan, wat er allemaal móeilijk kan en wat er misschien wel helemáál niet kan. En wie had hier om gevraagd? Inderdaad, helemaal niemand en dat is precies het punt: er is nooit een uitrolschema gespecificeerd geweest.

Vijf release-cycli

Stilzwijgend is de applicatie-eigenaar ervan uitgegaan dat uitrollen helemaal vanzelf zou gaan. Bij kleine, relatief eenvoudige sites, waar de oorspronkelijke ontwikkelaars ook het beheer hebben overgenomen, kan dit misschien nog steeds het geval zijn. Maar bij grote, complexe sites, waarbij downtime ook nog eens rechtstreeks kan worden vertaald in omzetderving of imagoschade, is het verstandig al helemaal in het beginstadium van de ontwikkeling van een applicatie een uitrolschema te formuleren, waar vervolgens de architectuur op afgestemd kan worden.

De basiselementen van een uitrolschema zijn vijf verschillende release-cycli:

1) Een nieuwe versie

Een nieuwe versie is de meest grootschalige wijziging op het systeem. We denken aan een echte 2.0 versie waarbij ook wijzigingen in de hardware of het netwerk mogelijk zijn. Het spreekt voor zich dat een nieuwe versie grondig doorgetest moet worden. Het opleveren van een nieuwe versie is bijna te vergelijken met het opleveren van een geheel nieuwe applicatie, behalve dan dat de site bij de gebruikers bekend is en vanaf de eerste dag met het volle aantal bezoekers te maken zal krijgen. Het tempo waarin nieuwe versies opgeleverd worden ligt ergens tussen de 2 en 8 jaar

2) Major release

'Major' noemen we de release waarin significante wijzigingen of uitbreidingen van de functionaliteit zijn ondergebracht. Weliswaar hebben we het niet over een 2.0, maar altijd nog over een 1.1. In de meeste gevallen zal er een ontwikkelteam voor

Berry Vorstenbosch
is Release Manager bij Auto Trader.

zijn samengesteld en zal de leiding in handen zijn van een projectmanager. Ook in het geval van een major release zal er grondig getest moeten worden. Niet alleen de nieuwe of gewijzigde functionaliteit, maar alles wat in de site door de wijzigingen beïnvloed kan worden, moet getest worden (regressietest). Over het ontwikkelen van een major release gaan meestal enkele maanden heen.

3) Minor release

Naast een major release kan er ook een minor release worden gedefinieerd, een 1.0.1. Hierin zijn kleine wijzigingen en bugfixes opgenomen. Eén van de belangrijkste taken van releasemanagement is om te bepalen welke veranderingen thuishoren in een major en welke in een minor release. Bij een minor release zijn de wijzigingen op het systeem minder ingrijpend, wat betekent dat de testinspanningen kleiner zijn en de risico's bij uitrol beduidend lager zijn. In de regel zullen minor releases niet door projectteams worden gedaan, maar door applicatiebeheer. Minor release worden eerder binnen weken maanden uitgerold.

4) Gestructureerde data

Veel informatieverschaffing is zelf aan een cyclus onderhevig – denk aan nieuwsbrieven, wijzigingen van het assortiment in een webwinkel, het menu in een restaurant, of gewijzigde reistijden wegens werkzaamheden. Het kan verstandig zijn om een gedeelte van de informatieverschaffing in een week- of maandcyclus onder te brengen. Naar binnen toe kunnen organisaties daar de bedrijfsprocessen op inrichten en naar buiten toe is de informatieverschaffing consistent en transparant.
Dit betekent dat er binnen deze cyclus zonder al te veel inspanning een brok data op de site geplaatst moet kunnen worden en dat de architectuur daarop ingericht dient te zijn. Technisch zijn er talloze mechanismen denkbaar die zo'n cyclus verzorgen en de mogelijkheid bestaat diverse stappen daarin te automatiseren.

5) Ad hoc data

Bij ad hoc informatie kan bijvoorbeeld gedacht worden aan nieuws of aan informatie over storingen. Het is informatie die snel, dat wil zeggen, binnen een dag, binnen een uur, of zelfs on the fly beschikbaar moet zijn. In verreweg de meeste gevallen zal voor deze cyclus een CMS (Content Management Systeem) worden ingezet.

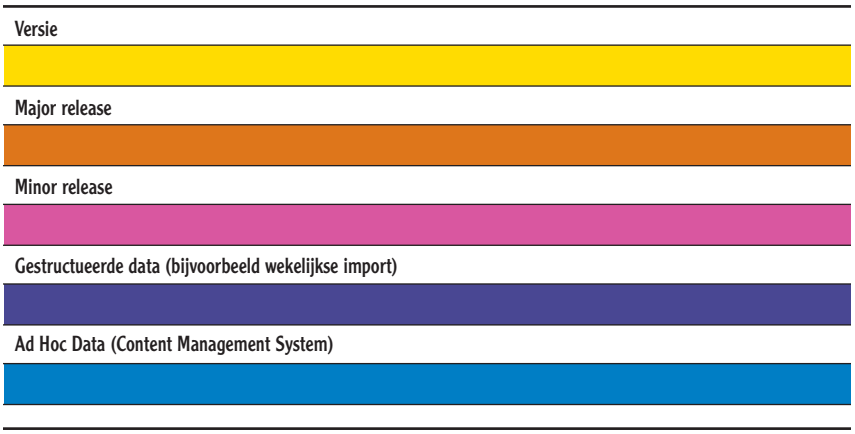
De kracht

Problemen met uitrollen zijn vaak terug te voeren op het plaatsen van elementen in de verkeerde releasecyclus. Een berucht voorbeeld is de 'hard' gecodeerde tekst. Ooit had iemand bedacht dat een bepaalde tekst niet CMS-gestuurd hoefde te zijn, omdat deze toch nooit gewijzigd zou gaan worden. En als de tekst dan tóch gewijzigd moet worden, dan kan dat pas gebeuren in de eerstvolgende minor release die misschien wel veertien dagen op zich laat wachten. En leg dat maar eens uit, zo'n klein tekstje…
Ook hier geldt, voorkomen is beter dan fixen. Hoe meer helderheid over wat waar thuishoort, hoe beter. Is de opdrachtgever in staat de datastromen en de toekomstige wijzigingen onder te verdelen in de vijf hierboven genoemde releasesoorten en kan de opdrachtgever aan iedere releasecyclus een tijdsbestek vastkoppelen? Dan beschikt hij over de belangrijkste elementen om een uitrolschema op te stellen. Met zo'n uitrolschema kan hij aan de bouwers eisen stellen over hoe en hoe vaak verschillende onderdelen van de applicatie uitgerold moeten worden. Of met andere woorden, het uitrolschema fungeert dan als een requirement.
Van daaruit kunnen vragen beantwoord worden als: wat wordt configureerbaar gemaakt en wat niet? Welke onderdelen moeten los van de applicatie-als-geheel kunnen worden uitgerold? Met een helder uitrolschema kunnen onaangename verrassingen worden voorkomen en kan het uitrolproces afgestemd worden op de wensen binnen de organisatie.

Een voorbeeld uit de praktijk

Om een en ander te illustreren nemen we als voorbeeld een technische applicatie bij een verpakkingsfabrikant. Deze applicatie moet via webtechnologie ontsloten worden. De verpakkingsfabrikant heeft een enorme hoeveelheid knowhow verzameld over welke verpakkingsmaterialen je voor welke producten het beste kunt gebruiken. In het laboratorium, of, in meer hedendaagse taal – bij het Research & Development Department – worden voortdurend metingen gedaan, nieuwe materialen onderzocht of eigenschappen van bestaande materialen geijkt. De rekenapplicatie zorgt ervoor dat men scherpe aanbiedingen kan maken.
In een niet al te ver verleden werden de resultaten van al dit onderzoek met een Excel-sheet door de hele organisatie heen gedistribueerd. Deze wijze van verspreiding met losse bestanden gaf sowieso

Voorkomen is beter dan fixen.



Figuur 1: De vijf release soorten.

Er wordt niets voor niets of verkeerd gebouwd.

al de nodige problemen, zoals het in omloop zijn van talloze versies. Nu, met een aantal fusies met buitenlandse bedrijven in het verschiet (waarvan al bekend is dat ze intern andere software gebruiken) wordt voorzien dat deze werkwijze onoverkomelijke problemen zal gaan opleveren.
Voldoende reden om de oude werkwijze op de schop te gooien en te gaan denken aan ontsluiting van gegevens middels webtechnologie. Alle bedrijven beschikken inmiddels over een intranet en op elke kantoor-pc staat een browser. Er wordt een aantal externe partijen in de hand genomen die de bedrijfsprocessen moeten dooranalyseren en de functionaliteit in de bestaande Excel-applicatie in kaart brengen. Samen met de hoofdverantwoordelijken van Research & Development komen ze tot een set requirements waarin ook gekeken is naar een toekomstige situatie waarin meertaligheid van belang zal zijn.

Wekelijks releasen?

Een van de eerste dingen die in de analyse duidelijk wordt, is dat een CMS niet nodig is. Het gaat om een technische applicatie die getallen oplevert die als input gelden voor een offerte. Eventuele nieuwsberichten kunnen via het intranet verspreid worden. Een andere mogelijkheid is het inbouwen van een pop-upje bij het opstarten, een veldje waar een beheerder eventueel een nieuwsberichtje in kwijt kan. Een heel CMS zou overkill zijn.
Verder blijkt uit de analyse dat het ombouwen van de Excel-applicatie naar een volledige beheermodule met een webinterface zowel duur als complex is. Daarom wordt ervoor gekozen verder te blijven werken met Excel als moederbestand. Omdat voorheen de Excel-sheet iedere week werd gedistribueerd, gaat de opdrachtgever de eis stellen om minimaal eens per week te kunnen uitrollen.
Hier tekent zich een belangrijk beslismoment af. In het denken bij de opdrachtgever is nog geen onderscheid gemaakt tussen wat we hierboven gestructureerde data en minor release hebben genoemd. Met het bouwen van een robuuste importmodule kan alles wat er in de sheet zit in de applicatie worden gezet zonder dat er daadwerkelijk gereleased hoeft te worden. De importmodule kan zo gebouwd worden dat de laatste versie van de sheet alleen maar op een bepaalde netwerkshare hoeft te worden neergezet om, in een nachtelijk verwerkingsproces, de webapplicatie met de meest recente data en rekenregels te voeden.

Nu de technische rekenmodule in een snelle releasecyclus is neergezet, kunnen de eisen voor de uitrol van een minor release versoepeld worden. Een minor release is een veel zwaarder en dus duurder traject: er wordt code vervangen (vaak door een externe partij), er zijn grotere risico's en er moet getest worden. In de minor release komen alle wijzigingen terecht die niet direct met de interne rekenmachine te maken hebben – zoals bijvoorbeeld het tonen van resultaten in verschillende eenheden.

Meertaligheid

En dan het uitrollen van de nieuw toegevoegde functionaliteit, in dit voorbeeld met name meertaligheid. Hoe je meertaligheid het beste kan uitrollen hangt helemaal af van het type applicatie, en van de exposure die de teksten geven naar de buitenwereld. Bij een corporate website wil je onmiddellijk in staat zijn ook maar het kleinste spelfoutje per direct te vervangen door de juiste tekst, maar in talloze applicaties zijn dergelijke drastische eisen niet nodig. In ons voorbeeld van een Research & Development afdeling van een verpakkingsfabrikant was sowieso al afgezien van een CMS en bestaat de hele cyclus die we hierboven ad hoc data hebben genoemd domweg niet.
Het lijkt voor de hand te liggen om de vertalingen van alle teksten die op de website voorkomen ook via een kort-cyclische importmodule te regelen. Een comma-separated bestand, een databaseje, of opnieuw een Excelsheet die ergens op een bepaalde plek neer wordt gezet. Echter, wat op het moment van de bouw van de applicatie nog helemaal niet duidelijk is, is hoe het verzamelen van vertalingen in de toekomst zal gaan, welke formaten er gebruikt zullen gaan worden. Andere vragen zijn – om hoeveel teksten en talen gaat het eigenlijk? Hoe erg is het als nieuw aangesloten gebruikers de eerste tijd moeten volstaan met Engels als default taal?
Op grond van dergelijke afwegingen kan worden besloten helemaal geen importmodule te bouwen, maar het toevoegen van een nieuwe taal vooralsnog onderdeel te laten zijn van een minor release. Deze situatie, zo schat men, zal voor de komende twee jaar voldoen. Een minor release zal dan mogelijk iets duurder worden, maar om deze al in een langzame cyclus van 6 weken zit, zullen de kosten op jaarbasis klein zijn, veel kleiner dan de bouw van een aparte importmodule. De toevoeging van een dergelijk module kan dan doorgeschoven worden naar een latere major release.

Spagaat

In het voorbeeld dat we hier hebben geschetst zijn wat het uitrollen betreft steeds bewuste keuzes gemaakt. Het uitrollen van de rekenmodule is helemaal losgetrokken van het uitrollen van andere functionaliteit. Stel je een situatie voor waarin er een fout in de rekenmodule wordt ontdekt die onmiddellijk moet worden hersteld, terwijl die rekenmodule deel uitmaakt van een release die wegens andere defects vertraging oploopt.
De releasemanager komt dan in de beruchte spagaat: kiezen tussen een teveel aan risico of een teveel aan doorlooptijd. Een goed uitrolschema helpt dit te voorkomen. «