Testen en architectuur als basis voor een duurzame informatievoorziening

Combinatie van architectuur, testen en conceptuele modellen is essentieel

AG CONNECT - 18 OKTOBER 2018

Organisaties zijn vaak op zoek naar snelle oplossingen. Kwaliteit krijgt hierdoor niet altijd de aandacht die het verdient. Tegen de tijd dat het gebrek aan kwaliteit opbreekt, is het vaak al te laat. Architectuur en testen zijn twee belangrijke kwaliteitsmaatregelen die hand in hand gaan en essentieel zijn om kwaliteit te laten slagen en daarmee tot een duurzame informatievoorziening te komen. Door testdata, testgevallen en testscripts te bouwen op architectuurkeuzes en modellen, ontstaat een stevig bouwwerk. Danny Greefhorst en René Ceelen geloven in een combinatie van architectuur, testen en conceptuele modellen.

De basis voor zowel architectuur als testen zijn doelstellingen en behoeften. Deze worden vertaald naar meer speci eke keuzes, zoals principes, eisen en ontwerpkeuzes. Hierdoor ontstaat een logische redeneerlijn van kaders naar inrichting. De andere kant van de medaille is het toetsen of aan al dit soort keuzes is voldaan. Dit is de basis voor het algemeen bekende V-model, waarin voor elk van de keuzes een tegenhangend niveau van testen bestaat. Het is logisch architectuur onderdeel te laten zijn van dit V-model. Architectuurkeuzes en de eisen die daaraan ten grondslag liggen, kennen dan ook een eigen vorm van testen. Dat kunnen zowel businessgerelateerde keuzes zijn als IT-speci eke keuzes.

Ontwerpruimte

Architectuurkeuzes zijn de basis voor eisen en ontwerpen; ze geven aan welke ontwerpruimte bestaat. Ze vormen een belangrijke schakel tussen strategie en uitvoering en moeten in het proces en de traceerbaarheid dus een duidelijke plaats krijgen. De hoogst liggende architectuurkeuzes zijn architectuurprincipes die aangeven in welke richting oplossingen moeten worden gezocht. Omdat een belangrijk deel van deze architectuurkeuzes is opgeschreven in natuurlijke taal en het resultaat is van een subjectief proces, zal een test of oplossingen hieraan voldoen voor een belangrijk deel een menselijke beoordeling zijn. Een architectuurprincipe zou bijvoorbeeld kunnen zijn dat medewerkers de beschikking hebben over een integraal klantbeeld. Of een specifiek klantgerelateerd gegeven nu wel of geen deel uitmaakt van dit integrale klantbeeld, is niet altijd even duidelijk en vraagt al snel onderlinge afstemming.

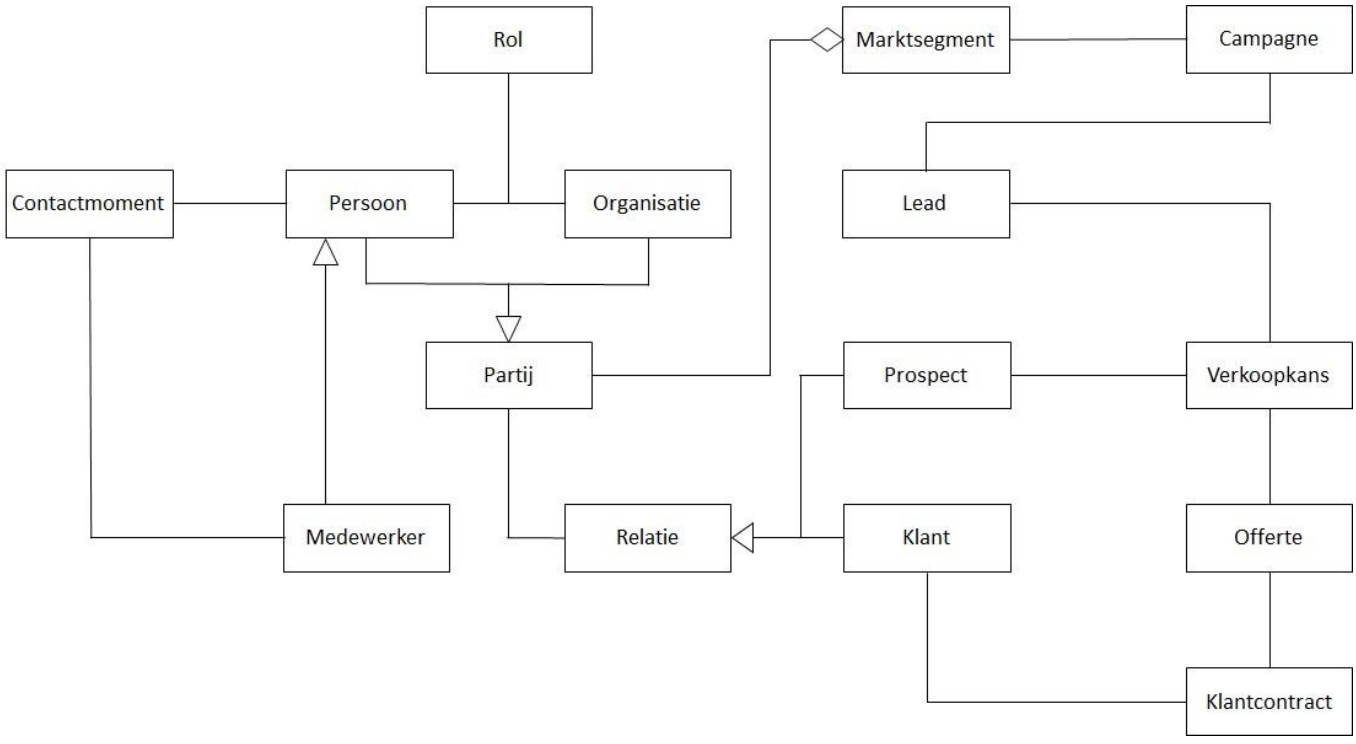
> ”Het lastige van modelgebaseerd testen is dat het relatief veel inspanning kost om de modellen te definiëren.”

Er zijn wel mogelijkheden om een deel van de architectuurkeuzes verder te formaliseren, zodat ook geautomatiseerde toetsing mogelijk is. Dat is voor een deel al gebruikelijk als het gaat over keuzes die betrekking hebben op software; het is vrij eenvoudig om automatisch controles uit te voeren op

programmacode. Zo kan bijvoorbeeld een architectuurprincipe op het niveau van softwarearchitectuur zijn dat er een strikte scheiding is tussen de presentatie en de opslag van gegevens. Een dergelijke regel kan controleren of in modules die bedoeld zijn voor presentatie geen programmacode staat die betrekking heeft op de opslag van gegevens. Dergelijke controles kunnen dan ook goed worden geïntegreerd in de dagelijkse integratie-, bouw- en deploymentcyclus. Hierdoor krijgen ontwikkelaars direct feedback op hun programmacode.

Het belang van modellen

Een ander belangrijk deel van architectuur is het gebruik van modellen voor het beschrijven van processen, gegevens, regels en applicaties. Deze modellen zouden de basis moeten zijn voor gedetailleerdere ontwerpmodellen, zodat een traceerbaar geheel ontstaat. Als deze modellen ver genoeg worden verdiept en worden voorzien van alle relevante regels, dan is het zelfs mogelijk hier werkende applicaties uit te genereren. Dit kan de productiviteit van applicatieontwikkeling sterk verhogen. Het leidt ook tot kwalitatief betere applicaties, die ook langer waardevol blijven en dus duurzaam zijn.



Figuur 1 Conceptueel model voor CRM

Ook vanuit het perspectief van testen is een modelgebaseerde benadering ideaal. Uit modellen kunnen geautomatiseerd (delen van) testgevallen, testscripts en testdata worden gegenereerd. Dat leidt tot een belangrijke versnelling. Bijgaand conceptueel model geeft bijvoorbeeld de fundamentele structuur weer van de voor CRM relevante gegevens. Uit dat model vloeien onder meer de volgende testgevallen voort:

- Maak relatie aan van type prospect (testinput: Janssen) (testverwachting: Janssen is aangemaakt).
- Maak verkoopkans aan in fase gesprek voor prospect Janssen (testinput: verkoopkans €5k ontwikkelen architectuurmodel) (testverwachting: verkoopkans is aangemaakt).
- Maak offerte aan voor verkoopkans €5k (testinput: verkoopkans €5k ontwikkelen architectuurmodel) (testverwachting: offerte is aangemaakt).
- Maak contract aan voor offerte €5k (testinput: verkoopkans €5k ontwikkelen architectuurmodel) (testverwachting: contract is aangemaakt en relatie type Janssen is klant).

Het lastige van modelgebaseerd testen is dat het relatief veel inspanning kost om de modellen te definiëren. Modelgebaseerd testen is een goede combinatie met testautomatisering, maar de laatste is vaak onvoldoende rechtvaardiging om formele modellen op te stellen. Als echter een combinatie wordt gemaakt met een architectuurgebaseerde benadering en een modelgebaseerde applicatieontwikkelaanpak, dan ontstaat een logisch geheel. Het zal ook een belangrijke boost geven aan de professionalisering van softwareontwikkeling en testen. Een dergelijke benadering past ook goed bij het automatisch genereren van testdata. Privacyoverwegingen (onder meer vanuit de AVG) maken het gebruik van productiedata voor testen lastig.

Uitdagingen met modellen

Het lastige aan modellen is dat de keuzes die eraan ten grondslag liggen vaak niet expliciet zijn gemaakt. Toetsen of oplossingen aan modellen voldoen, is daarmee niet eenvoudig; hoe erg is het als een oplossing niet precies voldoet aan een model? Overigens kan er prima een relatie gelegd worden van onderdelen van een model naar eisen of keuzes die eraan ten grondslag liggen. Dit vraagt wel toolondersteuning om het op een beheersbare manier te kunnen doen. Architectuurmodellen zijn vaak meer bedoeld als communicatie-instrument dan als formeel model. Ze worden vooral gebruikt om over dingen te kunnen praten en mensen inzichten te geven. Modellen in die categorie zijn conceptuele modellen; ze de niëren relevante concepten en begrippen en hun samenhang. Een misvatting is dat conceptuele modellen ook per de nitie 'vage' en slecht gede nieerde modellen zijn. Dat is niet de intentie van conceptuele modellen; ze zijn juist bedoeld om concepten duidelijk te maken en de juiste betekenis over te dragen.

> ”Tijdens de inrichtingsfase worden bepaalde cruciale keuzes 'in de mond' gelegd door de leverancier om de voortgang van de implementatie te borgen.”

In de praktijk worden 'echte' conceptuele modellen ook nogal eens overgeslagen omdat ze als overlast worden gezien. De modellen die worden gemaakt, zijn vaak een mix van dingen. Concepten, ontwerp en implementatie lopen daarbij door elkaar heen. Om het werken met conceptuele modellen te laten slagen, moeten ze volledig zijn geïntegreerd in het proces en moeten tools aanwezig zijn die naadloze integratie ondersteunen. Dergelijke tools kunnen conceptuele modellen automatisch vertalen naar logische en fysieke modellen, en wijzigingen synchroniseren naar andere niveaus. De conceptuele modellen zelf moeten vooral een goede re ectie zijn van het domein en de taal die wordt gebruikt door

medewerkers die de processen uitvoeren. Ze zouden geen uitspraken moeten doen over inrichting of implementatie.

Uitdaging bij standaardapplicaties

Als organisaties besluiten gebruik te maken van standaard (ERP-)applicaties, dan is voorgaande niet een-opeen toe te passen. Bij standaardapplicaties valt er veel minder te kiezen en liggen de gegevens, processen en regels voor een belangrijk deel al vast. Ook hanteren dergelijke applicaties een eigen taal die voor gebruikers zichtbaar is in de woorden die gebruikt worden in de gebruikersinterface. Het los van de applicatie beschrijven van gedetailleerde modellen is dan verspilde moeite, omdat veel ervan al vast ligt. Conceptuele modellen zijn dan de meest waardevolle modellen om wel op te stellen, omdat zij niet ingaan op de inrichting en implementatie.

Architectuur en conceptuele modellen zorgen ervoor voor dat essentiële discussies eerder in het proces plaatsvinden, waardoor potentiële problemen en verkeerde keuzes sneller zichtbaar worden. Voorafgaand aan de aanschaf van een applicatie helpen ze om te bepalen of een standaardapplicatie kan passen op de organisatie. Ze helpen ook om te bepalen waar een standaardapplicatie standaard kan blijven en waar maatwerk nodig is. Door vooraf een aantal goede architectuurkeuzes en modellen te definiëren, wordt veel duidelijk over wat belangrijk is voor de organisatie en standaard niet voldoet. Is het bijvoorbeeld acceptabel om voor producten bepaalde kortingsregelingen niet te kunnen definiëren?

Een goed testontwerp is belangrijk om de gehele keten van een ERP-implementatie te borgen op kwaliteit en bruikbaarheid. Je kunt je voorstellen dat een testontwerp zonder conceptuele modellen als grondlegger lastiger en complexer te maken is. Om nog maar te zwijgen over de traceerbaarheid naar bepaalde principes en uitgangspunten. In de praktijk zien we vaak dat bij een ERP-implementatie waarbij conceptuele modellen gemaakt zijn, snel een eerste testronde opgezet kan worden met behoud van de totale samenhang. De testers hoeven dan niet in detail alle mogelijke varianten te doorlopen, maar kunnen met een wat abstracter testontwerp de belangrijkste transacties testen. We zien ook vaak dat tijdens de inrichtingsfase bepaalde cruciale keuzes “in de mond” worden gelegd door de leverancier om de voortgang van de implementatie te borgen. Met de eerste hoog-overtest komen deze direct boven tafel en kunnen dan expliciet ter discussie en besluitvorming worden aangeboden. Hierdoor borg je dus de architectuurprincipes in relatie tot de inrichtingskeuzes en de afgeleide testresultaten en issues.

Meer succes

Wij denken dat een combinatie van architectuur en testen essentieel is om ervoor te zorgen dat applicaties duurzaam worden ingericht en ook duurzaam worden gebruikt in de organisatie. Het testproces moet dan ook worden verrijkt met architectuur. Dit voorkomt dat architectuurkeuzes worden vergeten. Het zijn immers belangrijke eisen en het toetsen aan deze eisen moet dan ook onderdeel uitmaken van een integraal testproces. Uit onze ervaring zijn ERP-implementaties waarbij

vooraf architectuurkeuzes en goede conceptuele modellen zijn opgesteld, die tevens als basis worden gebuikt voor testen, meer succesvol.

DANNY GREEFHORST

is directeur van ArchiXL en werkzaam als enterprise-architect en consultant. Hij is voorzitter van de SIG architectuur van het KNVI alsook voorzitter van de Stichting Digital Architecture.

RENÉ CEELEN

is directeur van TestMonitor, een adviesbureau op het gebied van ERP-testmanagement. Hij is onderzoeker op de Radboud Universiteit Nijmegen op de faculteit Institute for Computing and Information Sciences.