Pemberton heeft gelijk!

Door Dino Seelig,
visualrose@visualrose.nl

Steven Pemberton gaf in zijn presentatie op XML Holland 2010 aan het verstandiger te vinden om met XML, XForms en XSLT applicaties platformonafhankelijk te *beschrijven* dan ze moeizaam met JScript of een andere procedurele taal te *bouwen*.
ICT-architect Dino Seelig gaat in dit artikel in op de vraag waarom het verstandig is een scheiding tussen techniek (platformspecifiek) en functionaliteit (platformonafhankelijk) aan te brengen en geeft tevens antwoord op de vraag: dekt XML met XForms voldoende de lading om, in de vorm van metadata die beperkingen aan de data-invoer opleggen, de gewenste functionaliteit in voldoende detail te kunnen beschrijven?

De behoefte

"Verstandiger" klinkt als een oplossing voor een probleem, maar welk probleem lost Pemberton op. Wat was zijn behoefte?
Het aanbrengen van scheiding tussen techniek en functionaliteit komt voort uit de behoefte om kwaliteit te beheersen en om voortschrijdend inzicht zowel functioneel als technisch met terugwerkende kracht te kunnen verwerken.
Managementadviseur William Edwards Deming gaf in zijn *Deming Cyclus* aan dat kwaliteit alleen beheerst kan worden als men zegt wat men doet, doet wat men zegt, controleert of datgene wat men doet effectief is en men aan de hand van die bevindingen het proces bijstelt: *Plan, Do, Check, Act* (PDCA).
PDCA impliceert een hoge mate van herhaling en herhaalbaar uitvoeren van de herhaling (een hoge mate van standaardisatie). Binnen PDCA zullen ontwikkelaars vergelijkbare vraagstukken (denk aan een CRUD-patroon voor het *Creëren, Raadplegen, Updaten en Deleten* van data) eenduidig oplossen en coderen.
In mijn optiek is een hoge mate van standaardisatie slechts één stap in de goede richting.
Geautomatiseerde herhaling is het sleutelwoord om met terugwerkende kracht voortschrijdend inzicht te kunnen verwerken. In de traditionele werkwijze is het te kostbaar om voortschrijdend inzicht in reeds gerealiseerde applicaties te verwerken. Met als gevolg dat op den duur de geschiedenis in al zijn facetten in de ontwikkelde software terug te vinden is. Door opkomst en ondergang van verschillende technieken ontstaat daarmee geleidelijk een automatiseringsinfarct. Bypass kan het leven nog even rekken waarna men besluit maar opnieuw te beginnen.
Scheiding tussen functionaliteit en techniek is de basis voor niet-herhaald programmeren.
Pembertons advies: de functionaliteit te beschrijven en deze te vertalen richting het ontwikkel platform van keuze via XHTML is dan ook verstandig.

Platformonafhankelijke metadata

De vraag die ik me stelde luidde: is XForms geschikt om platformonafhankelijk de gewenste functionaliteit te beschrijven? Om de functionaliteit platformonafhankelijk te kunnen beschrijven, zal je de informatiebehoefte moeten vastleggen (de elementen, de onderlinge samenhang, de datatypes en het waardebereik), de presentatielogica in de vorm van rapportage- en invoerformulieren en business logica.
XForms kan voor de beschrijving van de informatiebehoefte gebruik maken van *XML Schema Definitions* (XSD). Een XSD beschrijft de elementen, de onderlinge samenhang, de datatypes en het waardebereik. Metadata beschreven in een XSD dekken daarmee dezelfde lading als metadata beschreven in een *Entity Relationship Diagram* (ERD) of UML-klassediagram.
In menig modelleeromgeving ontbreekt echter de schakel om presentatielogica vast te leggen in de vorm van rapportages en invoerformulieren. Met behulp van XForms kan deze behoefte grotendeels abstract worden beschreven.
Door een XML-structuur te maken waarin XML-Schema's de informatiebehoefte beschrijven en XForms de rapportages en invoerformulieren kan men een groot deel van de functionaliteit platformonafhankelijk definiëren.

Rest de vraag: ontbreekt in Pembertons aanpak de mogelijkheid om complexe *business rules* te definiëren? Voor een antwoord onderzocht ik een aantal business rules-talen: Schematron (xPath), xRules (XPath), RuleML (XML notatie if then). Deze talen maakten direct of indirect gebruik van XPath, waaruit ik snel de boute conclusie trok dat menig business rule

gevalideerd zou kunnen worden met behulp van XSLT. Pembertons opmerking het verstandiger te vinden om met XML, XForms en XSLT applicaties in XHTML te bouwen is in mijn optiek zo gek nog niet.
Een XML-structuur gebaseerd op XForms en XML Schema is voldoende platformonafhankelijk en voldoende rijk om met behulp XSLT een werkende applicatie te genereren.

Bijkomende voordelen Pembertons advies

Zoals hierboven beschreven is het grote voordeel van deze werkwijze beheersing van kwaliteit en het kunnen verwerken van voortschrijdend inzicht in bestaande programmatuur. Daarnaast zijn er voor de business bijkomende voordelen:

- Business-analisten kunnen door de business abstract vast te leggen (eventueel ondersteund met een gereedschap) zich focussen op de business. Het werk van deze business-analisten wordt vertaald in een werkende applicatie volgens vaste regels zonder tussenkomst van een programmeur.
- Programmeurs hoeven zich alleen bezig te houden met techniek. Zij ontwikkelen en onderhouden hun framework en codegeneratoren. Programmeurs focussen daarmee op waar ze goed in zijn: techniek.
- De opdrachtgever krijgt datgene wat deze vroeg. Door hoge mate van standaardisatie en genereren van de technische implementatie neemt de Time2market af, nemen de testwerkzaamheden af, neemt de *Total Cost Of Ownership* (TCO) af, en neemt de *Return On Investment* (ROI) toe.

Conclusie

Ik deel Pembertons aanbeveling dat het verstandig is met behulp van XML, XForms en XSLT applicaties in XHTML te bouwen. De aanbevolen werkwijze kan breder toegepast worden.
Verder zijn in dit licht een aantal zaken nader onderzoek waard:

- Microsoft Lightswitch (nieuw en veel belovend),
- Mendix (abstract op de cloud),
- Uniface + Unifast framework (robuust en snel) implementeren de hierboven beschreven werkwijze.

Dino Seelig werkt bij Research & Development van ITIS (http://www.itis.nl/).