Thema

Technisch gezien is er steeds meer mogelijk rond model driven development. Vanuit economisch en organisatorisch oogpunt is echter nog veel onduidelijk. In dit artikel gaat Jos Warmer uitgebreid in op vragen, en natuurlijk de bijbehorende antwoorden, op het gebied van return on investment, organisatie en projectuitvoering.

Model Driven Development

Kosten, baten en organisatie

Jos Warmer
Partner Ordina

De ontwikkelingen rond model driven development hebben de laatste jaren een grote vlucht genomen. Vooral op het gebied van domain specifieke talen, in het Engels domain specific languages of kortweg DSL's. In verschillende technologieën zijn de tools krachtiger en volwassener geworden. De Microsoft DSL Tools, Jetbrain's Meta Programming System (MPS), het Eclipse Modeling Project, MetaCase en Oslo zijn slechts de belangrijkste voorbeelden.
Door deze tools is de tijd die nodig is om een complete DSL inclusief geavanceerde editor en code generatie te ontwikkelen teruggegaan van jaren naar maanden.
Dat maakt het in veel meer omstandigheden economisch interessant om een model gestuurde oplossing te gebruiken. Echter, de aanpak en de organisatie rond projecten die gebruik maken van DSL's is anders.

Vaste MDD tooling

Op de markt is een aantal tools verkrijgbaar waarmee volgens een vaste manier van modelleren een of meer voorgedefinieerde codegeneratoren gebruikt kunnen worden. Deze tools karakteriseren zich door het feit dat er relatief weinig vrijheid is voor afwijkingen. Hetzij de tools doen precies wat je wilt en dan zijn ze heel bruikbaar, hetzij je wilt iets anders dan de tools doen en dan wordt dit soort tools al snel lastig.
In dit stuk ga ik niet verder in op deze variant van model driven development. Met name omdat er geen ontwikkeling in de DSL's zelf zit zal de werkwijze, organisatie en wijze van terugverdienen sterk lijken op wat we al kennen van andere tools die te koop zijn.

Flexibele DSL tooling

DSL-toolkits geven je de mogelijkheid om zelf DSL's te ontwikkelen die precies zijn toegesneden op wat je nodig hebt. Voordeel is dat je de meest effectieve oplossing hebt, je krijgt precies wat je wilt. De potentiële voordelen van eigen DSL's zijn vele malen groter dan die van vaste MDD-tools, bovendien geeft de flexibiliteit de mogelijkheid om te allen tijde mee te veranderen met de eigen wensen en eisen. Vaak kun je als startpunt gebruik maken van bestaande DSL's, zolang deze maar met de juiste tools gemaakt zijn en als de sources beschikbaar zijn.
Nadeel in deze MDD-variant is dat ontwikkeling van een DSL, zelfs met de geavanceerde tools van vandaag, nog steeds tijd en dus geld kosten. Ook krijgen we te maken met onderhoud en versiebeheer van de zelf ontwikkelde DSL's.

Wanneer DSL's

Het ontwikkelen van een DSL kent kosten die terugverdiend dienen te worden. Afhankelijk van de situatie kan dat op twee manieren.

De beste situatie doet zich voor wanneer een DSL in meerdere projecten gebruikt kan worden. De investering wordt slechts eenmaal gedaan en het terugverdienen gebeurt in meerdere projecten. Het is dus van belang om naar de eigen organisatie te kijken en te analyseren welke projecten er uitgevoerd worden. Als blijkt dat er met regelmaat gelijksoortige projecten zijn, loont het zeker de moeite om naar MDD als ontwikkelhulpmiddel te kijken.
Vaak wordt beweerd dat ieder project uniek is, maar de uniciteit wordt veelal sterk overschat. Natuurlijk zijn projecten uniek, maar bij nadere beschouwing

Alles dat niet in modellen gevangen kan worden, wordt op een offshore-locatie ontwikkeld.

Voor de zelf ontwikkelde DSL's moet je een goede opleiding maken.

lijken projecten vaak veel meer op elkaar dan men op het eerste gezicht denkt. Als uit de analyse komt dat projecten echt allemaal geheel uniek zijn, dan zal return on investment moeilijk worden. Zijn er terugkerende patronen te herkennen dan is de situatie rijp voor MDD.
Het ontwikkelen van DSL's voor één enkel project is meestal niet kosteneffectief. Een uitzondering hierop kan zijn wanneer het een heel groot project betreft, denk aan projecten waar tientallen mensjaren voor nodig zijn. In dat geval kan er binnen een project voldoende terugverdienmogelijkheid zijn zodat het toch de moeite loont.

Nieuwe rollen

Bij gebruik van DSL's komt een aantal nieuwe rollen kijken. Specifiek voor de ontwikkeling van DSL's zijn nieuwe rollen noodzakelijk. Gezamenlijk noemen we de volgende drie rollen voor DSL-ontwikkelaars.

De eerste rol is die van DSL-enigneer: De DSL-engineer is een persoon die voor één of meer projecten DSL's ontwikkelt. Hij is op de hoogte van de mogelijkheden van de DSL-tooling en heeft ervaring in het gebruik ervan. Dit is een geheel nieuwe rol, die in de traditionele softwareontwikkeling niet bestaat. De DSL-engineer is een specialist die met gemak kan metadenken, die de anatomie van modelleertalen begrijpt en er plezier in heeft om tussen alle metaniveaus heen en weer te springen. Lenigheid van geest is hier gewenst.

De tweede rol is die van domeinexpert. De domeinexpert levert de kennis van het domein waar de DSL over gaat. Aangezien er vele domeinen mogelijk zijn is de domeinexpert meestal niet dezelfde persoon als de DSL-ontwikkelaar. Deze rol bestaat in een traditionele omgeving ook, echter het belang van deze rol is in een MDD-omgeving groter en er worden hogere eisen gesteld aan het conceptuele abstractievermogen van de domain expert.

De derde rol is die van software architect of senior ontwikkelaar. Deze rol is verantwoordelijk voor de uit de DSL gegenereerde code. In het geval dat de domeinen sterk technisch van aard zijn, kan deze rol door dezelfde persoon vervuld worden als de domeinexpert. Als het om een niet technisch domein gaat, zullen de rollen meestal gescheiden zijn. Ook deze rol bestaat al, maar verandert van karakter in een MDD-omgeving. In plaats van het opzetten van één applicatie dient de architect patronen te herkennen over applicaties heen, welke in alle soortgelijke projecten voorkomen. Net als bij de domeinexpert wordt het abstractievermogen sterk op de proef gesteld. Omdat het doel uiteindelijk werkende code is, moet de architect tevens in staat zijn om de abstracte patronen om te zetten in concrete werkende, kwalitatief hoogwaardige code, PowerPoint architecten hebben in deze rol weinig te zoeken.

Projectoverschrijdend

DSL's zullen zich noodzakelijkerwijs ontwikkelen. Verandering in de omliggende IT-wereld dienen geïncorporeerd te worden, nieuw ontdekte patronen worden toegevoegd om DSL's krachtiger te maken en lessons learned uit projecten die de DSL's gebruiken leiden tot verbeteringen. Dit leidt tot nieuwe versies van DSL's.

Aan de andere kant worden DSL's door projecten gebruikt. Een project is dan afhankelijk van de gebruikte DSL's, en wel van de specifiek gebruikte versie. Als er een nieuwe release van een DSL komt dient hier rekening mee gehouden te worden. In eerste instantie zal de oude versie van betreffende DSL's nog beschikbaar moeten blijven voor projecten die deze gebruiken. De vraag die volgt is of en zo ja wanneer projecten overstappen op de nieuwe versie van de DSL's.

In onze eigen praktijk hebben we als regel dat alle projecten zelf beslissen wanneer zij overstappen. Op deze manier kan een project de overstap laten samenvallen met andere wijzigingen / uitbreidingen van de applicatie die toch al in de planning staan. Hiermee wordt de extra overhead van het overgaan naar een nieuwe versie van de DSL's geminimaliseerd. Overigens is dit geen geheel nieuw aspect. Applicaties zijn afhankelijk van talloze componenten, tools en frameworks. Voor al deze onderdelen geldt dat er regelmatig nieuwe versies komen en dat er duidelijkheid dient te zijn over het tijdstip en de wijze waarop projecten naar nieuwe versies overstappen. Een DSL is in dit geval gewoon een extra onderdeel waarmee rekening gehouden dient te worden.

Bovenstaande is zeker niet de enige manier, maar wel een die in de praktijk goed blijkt te werken. Essentieel is dat DSL's op gedefinieerde wijze releasematig beheerd worden. Een DSL is een stuk software en alle regels die betrekking hebben op het beheer van software gelden ook hier.

Projecten die gebruik maken van DSL's kunnen tegen problemen bij het gebruik aan lopen. Omdat DSL's voor projecten altijd op het kritieke pad liggen, is het dus nodig om een supportteam voor de DSL's beschikbaar te hebben. Het is van belang om ervoor te zorgen dat expertise vanuit de DSL-ontwikkelaars snel beschikbaar is. In veel gevallen, zeker wanneer het gebruik van MDD relatief nieuw is, kan dit een parttime rol zijn.
Een laatste aspect dat van belang is betreft de opleiding voor het gebruik van DSL's. Voor zelf ontwikkelde DSL's zijn geen opleidingen op de markt te koop, dus zo'n opleiding dient binnen de eigen organisatie gemaakt en gegeven te worden. Dit hoeft soms maar een workshop van een dag te zijn, maar mag niet vergeten worden.

Projectinrichting

Projecten die gebruik willen maken van DSL's hebben naast de traditionele taken enkele extra aspecten waarmee rekening gehouden dient te worden. Aan het begin van een project is een extra taak om zo snel mogelijk een keuze te maken voor welke DSL's gebruikt kunnen worden. Soms kan een DSL ongewijzigd gebruikt worden, het komt ook voor dat een DSL enkele aanpassingen nodig heeft om geschikt te zijn voor een project. Indien wijzigingen nodig dienen deze in de eerste uitgevoerd te worden.
Indien binnen een groot project besloten wordt dat het nuttig is om een of meer DSL's speciaal voor het project te ontwikkelen ontstaat een geheel nieuwe situatie. Het is aan te raden om een apart deelproject voor de DSL ontwikkeling in te richten, die DSL's oplevert welke door het overkoepelende project gebruikt worden. Dit DSL project dient snel opgestart te worden omdat het op het kritieke pad van het overkoepelende project ligt.
Voor zo een DSL project werkt een agile aanpak het best. Lever zo snel mogelijk een eerst versie van de DSL's op zodat ze direct getoetst kunnen worden in het overkoepelende project en gebruik de feedback om de DSL's verder te ontwikkelen.

Projectuitvoering

Als de DSL's eenmaal gekozen en beschikbaar zijn is de projectuitvoering over het algemeen weinig anders dan zonder gebruik van DSL's. Er is een extra taal met tool beschikbaar dat gebruikt wordt door de projectleden. Afhankelijk van het karakter van de gebruikte DSL's zijn bij sommige projectleden misschien extra vaardigheden nodig waarvoor een stukje opleiding gewenst is.

De uitvoering van MDD projecten biedt wel nieuwe mogelijkheden. Omdat de stap van model naar werkende applicatie veel sneller verloopt wordt een agile werkwijze gefaciliteerd. Of een project hier gebruik van maakt is een projectbeslissing.

MDD en offshoring

In software ontwikkelprojecten is offshoring een niet weg te denken fenomeen. Een interessante vraag is welke rol MDD kan spelen in relatie met offshoring.
Een van de grote problemen bij offshoring is de kwaliteit. Het is bijzonder lastig om de kwaliteit van de opgeleverde software goed te controleren. Zelfs als er afspraken over de te volgen architectuur gemaakt zijn blijkt dat de controle hierop veelal lastig is. Vaak wordt daarom alleen op grote lijnen een controle gedaan. Dat is onvoldoende, omdat juist in de details nog heel veel fout kan zitten.
Het gebruik van MDD kan hiervoor een oplossing geven. Vanuit de modellen wordt code gegenereerd. Deze code is geheel onder controle van de codegenerator. Hoewel meestal niet alle code gegenereerd wordt, is het juist de architectureel bepalende code welke wel kan worden gegenereerd. Dit zijn met name de structurele onderdelen en alle boilerplate code die ervoor zorgt dat alle architectuurlagen op de juiste wijze met elkaar communiceren.

Specifieke business logica is vaak lastiger te modelleren, en wordt grotendeels met de hand naast de gegenereerde code geschreven. Deze code moet echter passen binnen de kaders van de gegenereerde code en passen derhalve per definitie op de juiste wijze in de architectuur.

Model Driven offshoring

In model driven offshoring worden de modellen, welke het dichtst bij de business liggen, in Nederland gemaakt. De modellen bevinden zich op een hoger abstractieniveau en liggen dichter bij de klant dan code, dus is het goed om voor deze activiteit dicht bij de klant te zitten. Vanuit de modellen wordt code gegenereerd die de architectuur van de applicatie bepaald.

Alles dat niet in modellen gevangen kan worden, wordt vervolgens op een offshore locatie ontwikkeld. Dit is het arbeidsintensieve deel, precies datgene waarvoor offshore interessant is, zowel; vanwege de prijs als vanwege de beschikbaarheid van mensen. Doordat de offshore ontwikkelaars alleen code kunnen toevoegen aan de uit de modellen gegenereerde code blijft de architectuur van de applicatie 100% gegarandeerd. Daarmee is MDD een prima instrument om controle te krijgen op de kwaliteit van de applicatie en de door de offshore partner geschreven code.
De opgeleverde applicatie kan, omdat hij voldoet aan een duidelijke en bekende architectuur, zowel

Het is raadzaam om een apart deelproject voor de DSL ontwikkeling in te richten

in Nederland als op de offshore locatie onderhouden worden. De afhankelijkheid van de offshore partner wordt hierdoor verminderd, wat de bedrijfsvoering meer flexibiliteit geeft.

Model Driven "inshoring"

Afhankelijk van de grootte van een project kan MDD er voor zorgen dat offshoring steeds minder toegevoegde waarde heeft. Hoe meer code er gegenereerd wordt, des te minder het arbeidsintensieve handmatige werk wordt. Als MDD ver genoeg gevorderd is om alle aspecten van software te modelleren kan de offshore route misschien wel overbodig worden. Het hele software ontwikkelproces is dan zo efficiënt dat lage lonen niet meer nodig zijn. Daar staan we nog niet, maar het is wel een interessante mogelijkheid waarmee strategisch rekening gehouden kan worden.

MDD en onderhoud

MDD wordt in de praktijk vaak specifiek in verband gebracht met nieuwbouw trajecten. Je zou hieruit haast opmaken dat MDD in onderhoud geen rol speelt. Niets is minder waar! Een goede MDD omgeving met goede code generatie is zo opgezet dat de modellen gedurende de hele levensduur van een applicatie gebruikt blijven worden.

De meeste kosten van software zitten, zoals alle onderzoeken uitwijzen, in het onderhoud van applicaties. Als we kijken naar een return on investment voor toepassing van MDD is het dus van groot belang dat we niet alleen naar de eerste nieuwbouw kijken. Er valt in de vele jaren van onderhoud substantieel terug te verdienen. Het terugverdienen gebeurt langs twee wegen.

Ten eerste is de hoeveelheid werk om modellen te veranderen veel kleiner dan de hoeveelheid werk van een handgemaakte applicatie. Tijdens onderhoud wordt er veel aan een applicatie gesleuteld en het resultaat is over het algemeen dat de applicatie steeds meer "bulten" krijgt. Aanpassingen volgen steeds minder de originele architectuur en de applicatie gaat in kwaliteit achteruit. Wijzigingen worden hierdoor steeds duurder en steeds risicovoller. In het uiterste geval durft men een applicatie zelfs niet eens meer aan te raken.
Als onderhoud gebruikt maakt van MDD, dan worden wijzigingen in modellen doorgevoerd. Vanuit de modellen wordt altijd architectureel consistente code gegenereerd. Dat betekent dat de kwaliteit van de code gedurende de gehele levensduur op hoog niveau blijft. Onderhoud wordt niet duurder en de applicatie blijft onder controle.

Een tweede onderhoudsaspect dat positief uitvalt bij MDD is de hogere kwaliteit van de code vanaf het begin. Een applicatie die met MDD gebouwd is blijkt minder fouten te bevatten en dus zijn er minder problemen op te lossen. Dit is te verklaren uit het feit dat code generatoren vaak hergebruikt worden en dus al snel een hoge kwaliteit leveren. Ook zijn codegeneratoren een stuk consistenter dan een team van handmatige ontwikkelaars.
Hoewel harde cijfers op dit gebied ontbreken zou het goed kunnen zijn dat het grootste voordeel van MDD bij het onderhoud blijkt te liggen. Niet erg sexy, maar wel een grote kostenbesparing.

Conclusie

MDD kan de moeite lonen wanneer er bepaalde ontwikkelactiviteiten zijn die herhaaldelijk worden uitgevoerd. Het herkennen van dit soort terugkerende patronen is daarom de eerste stap in adoptie van MDD. Meestal zal dit over meerdere projecten het geval zijn, bij uitzondering ook voor hele grote projecten. Omdat DSL's een cruciaal deel van de ontwikkeltooling worden is het van belang om de ontwikkeling en beheer van DSL's structureel in de organisatie in te bedden, Het is geen eenmalige activiteit. MDD en offshoring kunnen prima samengaan en leveren voordelen op het gebied van kwaliteit en flexibiliteit. De besparing die met MDD tijdens het onderhoud oplevert wordt vaak vergeten, maar is substantieel en dient zeker in berekeningen meegenomen te worden. «

Alles dat niet in modellen gevangen kan worden, wordt op een offshore-locatie ontwikkeld.