Arthur van Adrichem en Steven Brom

Flexibiliteit versus doorlooptijd en budget?

WINDOWS WORKFLOW FOUNDATION EN MICROSOFT DYNAMICS CRM 4.0

Bij Qurius komen we geregeld in aanraking met integratietrajecten waarbij naast Dynamics CRM, ook Dynamics AX of Dynamics NAV belangrijke rollen spelen. In dergelijke trajecten is het soms lastig om op een flexibele en snelle manier businesslogica te implementeren, zeker wanneer communicatie met andere systemen noodzakelijk is. Biedt Windows Workflow Foundation misschien een oplossing?

De casus in dit artikel is gebaseerd op een geïntegreerd ledensysteem dat Qurius voor een vakbond heeft ontwikkeld. In dit systeem wordt CRM ingezet voor de registratie van leden, het bijhouden van contactmomenten, servicemanagement en marketing. Dynamics AX wordt onder andere gebruikt voor de berekening en inning van de contributie. Vanuit de business hebben gegevens in de backoffice vaak gevolgen voor wat mogelijk is in de front-office. Een voorbeeld hiervan is de openstaande contributie van een lid. Mag een lid nog een nieuwe case plaatsen op het moment dat er sprake is van een openstaand contributiebedrag? En zo ja, tot welk bedrag is dit toegestaan? In bovenstaande situatie spelen additionele wensen met betrekking tot flexibiliteit een rol. De acties die moeten worden uitgevoerd als er een openstaand saldo is geconstateerd, moeten eenvoudig door de klant zelf aan te passen zijn. Het is niet wenselijk om na het aanroepen van de Dynamics AX Contributie-webservice via JavaScript in het OnLoad-event van het scherm vervolgens bepaalde velden te blokkeren of een melding te geven. Een volledige maatwerkoplossing via JavaScript is immers niet of nauwelijks door de klant aan te passen.

Wat is de oplossing?

Wij zien een mogelijke oplossing voor bovenstaand probleem in de nieuwe, op Windows Workflow Foundation gebaseerde, workflow-engine van CRM. Workflows kunnen worden gemaakt in Visual Studio, maar ook direct via de CRM-webinterface. Het wordt hierdoor kinderspel om complexe workflows te maken die de gewenste business-rules implementeren. Het is ook mogelijk de workflow gedeeltelijk in Visual Studio en gedeeltelijk in CRM te maken. Dit kan handig zijn op het moment dat bepaalde acties niet via de grafische interface van CRM kunnen worden opgelost.
Het scenario komt er hiermee als volgt uit te zien: Een lid belt met een callcenter met een verzoek tot ondersteuning. De callcentermedewerker registreert het verzoek als aanvraag (een case) in CRM. Hierbij geldt dat de aanvraag alleen in behandeling wordt genomen als het lid een niet al te grote betalingsachterstand heeft. Nadat de aanvraag is opgeslagen, wordt automatisch een workflow opgestart. De eerste stap in de workflow is het ophalen van de contributieachterstand uit het financiële systeem (zoals Dynamics AX, Dynamics NAV, enzovoort). Vervolgens wordt op basis van de contributieachterstand bepaald wat er met de aanvraag gebeurt:

1. Bij een geringe achterstand wordt de aanvraag direct in behandeling genomen.
2. Bij een gemiddelde achterstand wordt handmatig door een medewerker bepaald of de aanvraag wel of niet in behandeling wordt genomen. Denk aan het bekijken van de betalingsgeschiedenis van het lid. Is het een structurele wanbetaler, of is er nu toevallig een keer een achterstand?
3. Bij een hoge achterstand wordt de aanvraag geannuleerd, wordt er direct een e-mail naar het lid gestuurd, en wordt een signaal afgegeven naar de financiële afdeling om iets te gaan doen met de betalingsachterstand.

Er is bewust gekozen voor het inzetten van de workflow-engine binnen CRM, omdat dit proces door de tijd heen onderhavig kan zijn aan wijzigingen. Het kan zijn dat de drempelwaarde voor het wel of niet in behandeling nemen van de aanvraag wijzigt, of dat de vervolgstappen in één van de drie scenario's anders wordt ingevuld. Door de workflow-engine te gebruiken kan het proces eenvoudig worden aangepast, zonder dat daar een programmeur voor nodig is. Het enige stuk dat uitgeprogrammeerd is, is het ophalen van het openstaande saldo. Hiervoor wordt een custom workflow-activity gebruikt, die via een webservice het saldo uit het financiële systeem ophaalt. In de vervolgstappen van de workflow is dit saldo dan beschikbaar als variabele. Hiermee wordt de flexibiliteit dus gewaarborgd.

Wat gaan we doen?

In dit artikel doorlopen we zes stappen die noodzakelijk zijn voor het implementeren van de gewenste workflow. We beginnen met het opzetten van het Visual Studio-project. Een workflow-project kan vanuit een standaard installatie van Visual Studio 2005 namelijk niet worden aangemaakt. Als het project is opgezet, kan gestart worden met het maken van de webservice die in dit artikel dienst doet als contributieservice. Vervolgens kan de custom workflow-activity worden gemaakt, ondertekend (signed) en uitgerold.

Stap 1: Opzetten Visual Studio-project

Om zelf workflows vanuit Visual Studio te kunnen ontwikkelen, dien je te beschikken over minimaal Visual Studio 2005. Hierbij moet Microsoft .NET 3.0 zijn geïnstalleerd en ook de Visual Studio Extensions voor Windows Workflow Foundation (zie referenties). De nieuwe SDK voor CRM 4.0 (zie referenties) bevat een aantal Visual Studio-templates die geschikt zijn voor CRM-workflows. Er staat duidelijk uitgelegd hoe deze geïnstalleerd moeten worden, dit onderdeel behandelen we daarom niet. Op het moment dat je de templates succesvol hebt geïnstalleerd, kun je een nieuwe solution aanmaken van het type 'CRM 4.0 Workflow'. Voor wie gewoon

```
namespace Qurius.DotNet.IntegrationWorkflows.BackofficeWebservices
{
   [WebService(Namespace = "http://www.qurius.com/")]
   [WebServiceBinding(ConformsTo = WsiProfiles.BasicProfile1_1)]
   [ToolboxItem(false)]
   public class ContributionWebservice : System.Web.Services.WebService
   {
      [WebMethod]
      public decimal RetrieveContributionBalance(string customerId)
      {
         //Aanmaken voor random object voor genereren bedrag
         Random contributionBalanceRandomizer = new Random();

         //Functie geeft een bedrag tussen de 0 en 1000 euro terug.
         return (decimal)contributionBalanceRandomizer.NextDouble() *
 1000;
      }
   }
}
```

Codevoorbeeld 1. De webservice-stub

eens wil experimenteren met workflows in CRM adviseren wij overigens om gebruik te maken van de VPC voor Microsoft CRM 4.0. Hierbij wordt de volledige ontwikkelomgeving direct meegeleverd.

Stap 2: Aanmaken stub-webservice

Voor de eenvoud van dit artikel maken we geen gebruik van webservices die Dynamics AX of Dynamics NAV ontsluiten, maar kiezen we voor het gebruik van een stub-webservice. Voeg een standaard ASP.NET-webservice toe aan de zojuist aangemaakte Visual Studio-solution, zorg voor een string als invoerparameter en geef een decimal terug. In codevoorbeeld 1 zie je hoe de webservice een willekeurig contributiebedrag teruggeeft.

We kiezen voor het 'simple type' string, in plaats van het 'complex type' GUID als invoerparameter, omdat de webservice daarmee eenvoudig getest kan worden via de webbrowser. Als we hier zouden kiezen voor een GUID is additionele tooling nodig voor het testen (zoals WebService Studio).

Stap 3: Aanmaken workflow-activity

Zoals in de casusbeschrijving werd aangegeven, hebben we in ons voorbeeld gebruikgemaakt van een custom workflow-activity voor het ophalen van de contributieachterstand. Een dergelijke custom

```
protected override ActivityExecutionStatus Execute(
   ActivityExecutionContext executionContext)
{
   try  {
      ContributionWebservice ws =
         new ContributionWebservice();

      Guid customerId = this.Customer.Value;

      // Voor het proces zijn afgeronde bedragen voldoende
      int roundedContributionBalance = (int)Math.Round(
         ws.RetrieveContributionBalance(customerId));

      this.ContributionBalance = new CrmNumber(
         roundedContributionBalance);

      return ActivityExecutionStatus.Closed;
   }
   catch{
      return ActivityExecutionStatus.Faulting;
   }
}
```

Codevoorbeeld 2. De Execute-methode van een custom workflow-activity

```
public static DependencyProperty CustomerProperty =
   DependencyProperty.Register("Customer",
   typeof(Lookup), typeof(RetrieveContributionBalance));
```

Codevoorbeeld 3. Declaratie van het Customer Dependency-property

workflow-activity is een herbruikbare activiteit die eenvoudig in meerdere CRM-workflows kan worden ingezet. De vraag is uiteraard: Hoe maak je nu zo'n custom workflow-activity?

Een custom workflow-activity is een klasse die erft van de baseclass *System.Workflow.Activities. SequenceActivity* (Let op, deze klasse is pas beschikbaar vanaf versie 3.0 van het .NET Framework). De klasse krijgt een extra attribuut (*CrmWorkflowActivity*) om aan te geven dat het hier een CRM-workflow-activity betreft. Met behulp van dat attribuut kun je ook aangeven onder welke naam (en optioneel onder welke categorie) de custom workflow-activity terug te vinden zal zijn in de CRM 4-workflow-module. In de overerfde klasse dien je de methode Execute te overschrijven, hierin voer je de gewenste acties uit (zie codevoorbeeld 2).

In ons geval willen we het contributiesaldo ophalen van een specifiek lid, en moeten we informatie doorgeven aan de workflow-activity. Hiervoor zijn twee acties nodig. Allereerst moet de Windows Workflow- engine op de hoogte gebracht worden dat een bepaalde parameter aanwezig is. Dit doe je door een DependencyProperty te definiëren; zie codevoorbeeld 3. De naam die je hier opgeeft voor de parameter geeft aan hoe deze bekend wordt gemaakt aan de workflow-engine, en hoeft dus niet overeen te komen met de naam van de property die je in de code gebruikt. Naast de naam geef je tevens het type van de property en de bovenliggende klasse aan.

Vervolgens voeg je een property toe om de waarde daadwerkelijk door te kunnen geven, waarbij je aangeeft of het een Input- of Output-parameter is; zie codevoorbeeld 4. Zoals te zien is in dit codevoorbeeld fungeert het DependencyProperty ook als 'propertybag' voor de doorgegeven waarde. Het is dus niet de bedoeling de waardes op te slaan in een private variabele, zoals je normaal gesproken gewend bent.

```
[CrmInput("Customer")]
[CrmReferenceTarget("contact")]
public Lookup Customer
{
   get { return (Lookup)base.GetValue(CustomerProperty); }
   set { base.SetValue(CustomerProperty, value); }
}
```

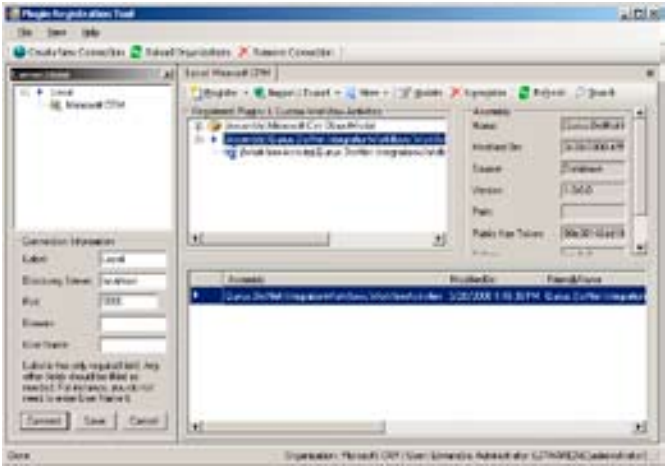
Codevoorbeeld 4. Definiëren van een CrmInput get/set-property

Zodra het saldo is opgehaald, willen we dit uiteraard weer terug commu niceren naar de aanroepende workflow. Ook hier maken we gebruik van een property, ditmaal met het kenmerk CrmOutput; zie codevoorbeeld 5. Let op: bij het teruggeven van een waarde vanuit de DependencyProperty moet je zelf casten naar het juiste type.

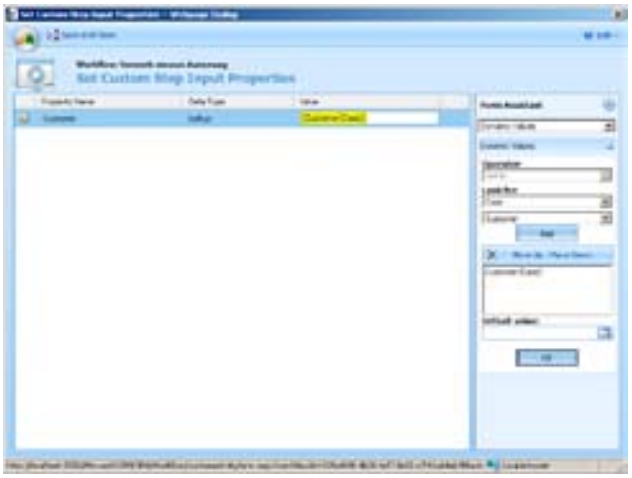
```
public static DependencyProperty
   ContributionBalanceProperty =
   DependencyProperty.Register("ContributionBalance",
   typeof(CrmNumber), typeof(RetrieveContributionBalance));

[CrmOutput("ContributionBalance")]
public CrmNumber ContributionBalance
{
   get { return (CrmNumber)
        base.GetValue(ContributionBalanceProperty); }
   set { base.SetValue(
        ContributionBalanceProperty, value); }
}
```

Codevoorbeeld 5. Definitie van de outputparameter



Afbeelding 1. De Plugin Registration Tool versie 2.



Afbeelding 3. Doorgeven van het lid

Stap 4: Signen van een workflow-activity

Voordat een assembly gebruikt kan worden in CRM dient deze te worden ondertekend. Ondanks dat dit ondertekenen of 'signen' regelmatig in het .NET Magazine terugkomt, doorlopen we toch nog even de stappen.

1. Klik met de rechtermuis knop op het project, kies 'Properties'
2. Ga naar het onderdeel 'Signing'.
3. Klik op 'Sign the assembly'
4. Maak een nieuwe strong name key aan, voer eventueel een wachtwoord in en klik op 'Okay'.
5. Selecteer indien noodzakelijk de zojuist aangemaakte sleutel.
6. Sluit het properties-venster.

Stap 5: Registreren van een workflow-activity

Het registratieproces van een custom workflow-activity is gelijk aan het proces voor het registreren van een plug-in in CRM 4. Helaas is er voor CRM 4 geen out-of-the-box tool beschikbaar, maar worden er twee voorbeeldapplicaties met broncode meegeleverd met de CRM 4 SDK. Op basis van de SDK-voorbeeldcode is de Plugin Registration Tool versie 2 (zie referenties) gebouwd, dit is ook de tool die wij hebben gebruikt om de workflow-activity te registreren. De tool is erg gebruiksvriendelijk (zie afbeelding 1) en behoeft geen verdere uitleg.

Stap 6: Benutten van een workflow-activity

Op basis van het businessscenario kiezen we er dus voor om de input van nieuwe aanvragen niet tegen te houden. De aanvraag wordt gewoon opgeslagen, de workflow zorgt er vervolgens voor dat de aanvraag naar de juiste wachtrij gaat en daarbij de juiste status krijgt. Tijdens de vorige stap is de workflow geregistreerd in CRM. De workflow-activity kan nu worden gebruikt in de CRM Workflow-editor.

Start met een nieuwe workflow. Voeg een stap toe, je ziet dat de nieuwe stap 'RetrieveContribution' er is bijgekomen, klik deze aan, geef de stap een logische beschrijving mee (zie afbeelding 2). Klik op Set Properties om er voor te zorgen dat het CustomerID naar de workflow wordt gestuurd. Deze koppeling kan worden gemaakt door op het veld 'Value' te klikken en vervolgens via de formulierenassistent de Customer te selecteren binnen de Case-entiteit (zie afbeelding 3).

Klik op OK. De waarde die deze stap teruggeeft, kan vervolgens worden gebruikt in een conditie. Voeg een conditie toe en selecteer in de lijst met variabelen de 'contributie' variabele. In het geval van een openstaande contributie lager dan €100,00 kiezen we er voor om de aanvraag de status 'In Progress' te geven en toe te wijzen aan de wachtrij 'Te verwerken aanvragen'. Voeg hiervoor (en naar eigen wens) nieuwe stappen toe die op basis van de uitkomst van de conditie moeten worden gestart; zie afbeelding 4. Als de workflow gereed is, klik je op 'Publish' om te workflow te activeren.

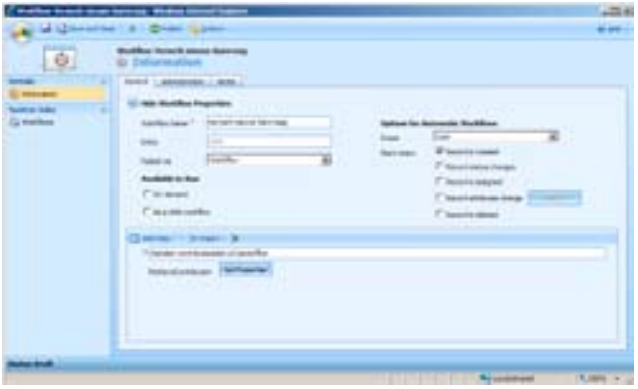
Tips voor ontwikkelaars

Als je de interface wijzigt van methodes of properties in je custom workflow-activity, dan lijkt het updaten via de plugin-registration tool prima te werken. Echter, jouw workflows zullen niet meer correct functioneren. De enige oplossing lijkt het herstarten van de CRM Async-service via de volgende commando's:

net stop "Microsoft CRM Asynchronous Processing Service"
gevolgd door:
iisreset
gevolgd door:
net start "Microsoft CRM Asynchronous Processing Service"

Tot slot

De in dit artikel gedemonstreerde workflow is opzichzelfstaand misschien niet de meest ingewikkelde workflow ooit, maar dit was ook niet de intentie. In dit artikel willen we de lezer wijzen op een mogelijke implementatie van business-



Afbeelding 2. Toevoegen van de zojuist aangemaakte workflow-activity

Afbeelding 4. Verder opzetten van de workflow

rules via de workflow-engine van CRM. De klant krijgt de gewenste flexibiliteit tegen lage kosten. Voorwaarde voor het inzetten van deze werkwijze is dat controle op ingevoerde gegevens achteraf kan worden toegepast. In situaties waar controle achteraf niet mogelijk is, moet worden teruggevallen op de plug-in architectuur van CRM. In onze ervaring kan de klant in veel gevallen prima leven met een scenario, waarbij controle op het naleven van business-rules naar boven komt door controle en sturing achteraf, in plaats van weigering bij invoer. Zeker gezien het feit dat het gebruikers niet stagneert in hun werkzaamheden. Vanuit deze optiek hopen wij dat dit artikel een bijdrage heeft kunnen leveren aan de oplossing voor de besproken problematiek. Wil je workflow in CRM echter ergens anders voor inzetten? Dan hopen wij dat het artikel praktisch genoeg is om zelf met workflow aan de slag te gaan.

Arthur van Adrichem is als technisch consultant werkzaam bij Qurius AS (www.qurius.nl) in Rijswijk. Zijn interesses liggen vooral op het gebied van Microsoft Dynamics CRM en Microsoft Office SharePoint Server. Voor vragen en opmerkingen is hij te bereiken via arthur.van.adrichem@qurius.nl.

Steven Brom is als technisch architect werkzaam bij Qurius AS (www.qurius.nl) in Rijswijk. Zijn interesses liggen vooral op het gebied van Microsoft Dynamics CRM. Voor vragen en opmerkingen is hij te bereiken via steven.brom@qurius.nl

Referenties

Complete code:
http://www.softwarestraat.net/magazine/2008/juni/CRMWorkflow.aspx
Microsoft CRM 4.0 SDK (versie 4.03):
http://www.microsoft.com/downloads/details.aspx?FamilyId=82E632A7-FAF9-41E0-8EC1-A2662AAE9DFB
Microsoft CRM 4.0 VPC:
http://www.microsoft.com/downloads/details.aspx?Familyid=DD939ED9-87A5-4C13-B212-A922CC02B469
Visual Studio 2005 extensions for .NET Framework 3.0 (Windows Workflow Foundation):
http://www.microsoft.com/downloads/details.aspx?FamilyId=5D61409E-1FA3-48CF-8023-E8F38E709BA6&displaylang=en
Plugin Registration Tool: http://code.msdn.microsoft.com/crmplugin/Release/ProjectReleases.aspx?ReleaseId=90