Waarom Workflows zo belangrijk zijn...

State Machine Workflows in SharePoint

John Sanders en Alfred de Weerd

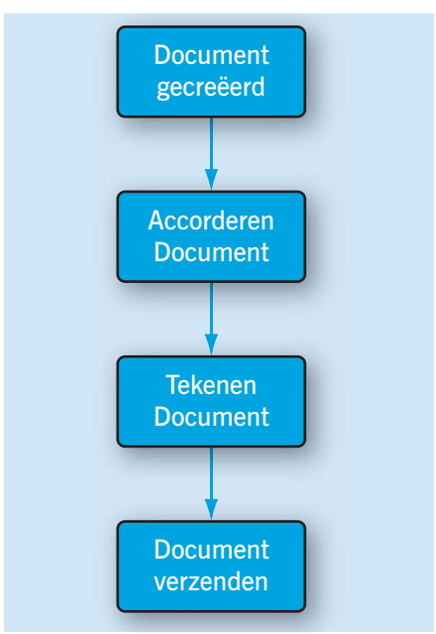
In SharePoint zijn workflows op drie manieren te definiëren: de ingebouwde workflows op basis van een template, door het samenstellen van standaard activities in SharePoint Designer of door het volledig zelf bouwen van een custom workflow in Visual Studio. Deze laatste oplossing staat centraal in dit artikel, met aandacht voor de verschillen en overeenkomsten tussen sequentiële en state machine workflows.

De afgelopen decennia

verbeterde de productiviteit van individuele kantoorwerkers spectaculair door Office-applicaties zoals Word en Excel enerzijds en de bedrijfsapplicaties anderzijds. Mogelijkheden voor samenwerken werden daarbij vaak niet geautomatiseerd, omdat de winst hier naar verhouding niet zo groot was.
In de huidige bedrijfsomgevingen is het automatiseren van workflows van groot belang, omdat we daarmee de doorlooptijd voor werk terugbrengen en de zekerheid verhogen dat een proces goed verloopt. Bovendien wordt er voldaan aan wettelijke vereisten omtrent de procesvoering.
Workflows om het samenwerken tussen mensen te ondersteunen noemen we human workflows. Er zijn ook systeem-workflows. Deze bevorderen het samenwerken tussen systemen en werken op basis van exact gedefinieerde gegevens en interfaces. Human workflows vereisen veel menselijke interactie waarbij we gebruikmaken van ongestructureerde data. Voor deze menselijke interactie is een user interface nodig die gebruikers de mogelijkheid biedt de workflow te starten, de voortgang te volgen en op de workflow te reageren.
Om gebruik te kunnen maken van de functionaliteit in Windows Workflow Foundation is er een aparte laag gebouwd voor de combinatie met SharePoint. SharePoint biedt naast een user interface namelijk ook een host-proces waarbinnen de workflow kan draaien. Verder wil je typische SharePoint-zaken meenemen in de workflow-implementatie. Wanneer een workflow van een bepaald type is gestart, worden instanties van de workflow gekoppeld aan documenten of list items. Het type workflow kun je daarbij laten afhangen van het documenttype (content type) van het item of van de List. SharePoint stelt gebruikers in staat te werken met de inhoud van document-libraries en lists. Workflows bevinden zich tussen de inhoud en de personen in en bepalen hoe de interactie tussen de twee plaatsvindt. Een kenmerk van SharePoint workflows is dan ook dat ze gebruiker- en document-georiënteerd zijn.

Goedkeuren van documenten

We verduidelijken de toepassingen van workflow in SharePoint met een voorbeeld waarin verschillende personen documenten bewerken alvorens een document als officieel te versturen. Het proces is simpel. Voor het afgeven van een vergunning creëren we een nieuw document dat de behandelaar van zijn akkoord voorziet. Daarna volgt de ondertekening door de verantwoordelijke en mag het document worden verzonden. Op het eerste gezicht een behoorlijk simpel en lineair proces, vrij eenvoudig te implementeren in een sequentiële workflow. Het gaat hier om het meest gebruikte type workflow, waarbij stappen in een vooraf bepaalde volgorde worden uitgevoerd.
In een documentbibliotheek op een SharePoint site voegen we een nieuw document toe. Hierna start SharePoint de workflow (omdat deze aan het content type in de documentbibliotheek is gekoppeld). Deze stuurt een e-mail naar de behandelaar die moet accorderen en maakt een task aan in de tasklist. De workflow wacht op het afronden van deze task. Als de task compleet is, gaat eenzelfde bericht naar de verantwoordelijke die moet tekenen, waarna weer wordt gewacht. Als laatste stap volgt verzending van het document.
Wanneer je dit aan de opdrachtgever oplevert, blijkt het inderdaad het proces dat hij voor ogen had. Alleen ging het hier slechts



SCHEMA 1. HET PROCES (BASIS)

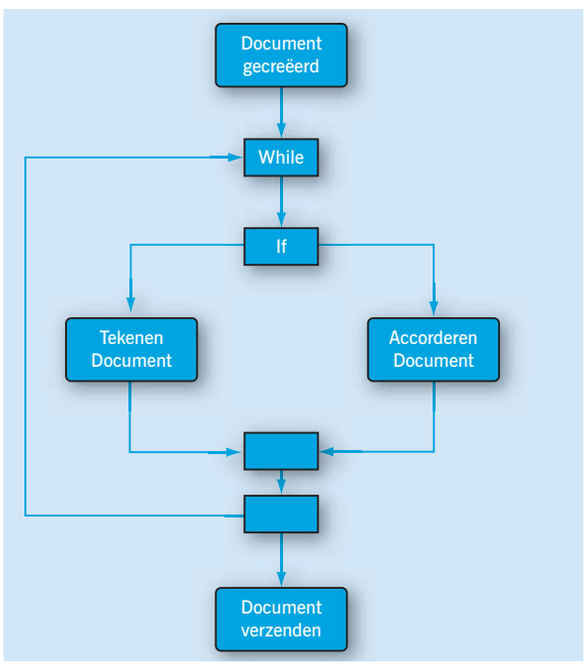
om de 'happy flow'. Het komt namelijk ook vaak voor dat het document eerst wordt getekend en daarna geaccordeerd. Dat kan je vast wel even bijbouwen? En oh ja, misschien komt er later nóg een stap bij. Of je daar alvast rekening mee wilt houden.
In een sequentiële workflow gaat dit niet zo gemakkelijk, omdat stappen altijd na elkaar komen. Terugspringen naar een eerdere stap in de procesflow kan niet in een sequentiële flow. Dit los je op door bijvoorbeeld een while-lus in te bouwen, waarbinnen alle stappen plaatstvinden. Je houdt bij wanneer alle stappen doorlopen zijn. In de conditie van de while-lus controleer je dit en je kijkt of aan alle condities is voldaan om het document te mogen verzenden.
Bekijk de flow van dit project nog eens goed… Is dit niet een state machine? Het grootste deel van logica in de workflow is gericht op het bijhouden van de toestand van het proces. Voor dit soort oplossingen bestaan state machine workflows. Laten we eens kijken naar de verschillen en overeenkomsten tussen sequentiële en state machine workflows.

Sequentieel versus State Machine

Ontwikkelaars denken bij workflows als eerste aan sequentiële workflows. Een flowchart geeft vaak een schets van het proces weer. Zo'n flowchart is dikwijls bijna één op één te vertalen in een sequentiële workflow, dus de keuze is dan snel gemaakt. Een sequentiële workflow kan echter alleen een vastomlijnd proces uitvoeren. Natuurlijk kun je binnen dit proces flexibiliteit inbouwen, maar deze flexibiliteit gaat alleen zover als jij dat vooraf hebt bepaald. Om deze reden zijn sequentiële workflows bij uitstek geschikt voor 'technische processen' die immers strak gedefinieerd moeten zijn. Daarnaast kunnen sequentiële workflows nuttig zijn als je processen juist heel rigide af wilt dwingen. Het vooral op 'technische processen' gerichte BizTalk heeft bijvoorbeeld alleen de mogelijkheid om orchestrations (workflows) op een sequentiële manier uit te voeren.
Businessprocessen waarin gebruikers ook een rol spelen, hebben vaak een grilliger karakter. Er zijn verschillende wegen die naar het eindpunt van het proces leiden en soms wordt een deel van het proces meer keren achtereen uitgevoerd. Het proces kan in een aantal toestanden verkeren en vanuit iedere toestand (stap in het proces) zijn één of meer andere toestanden bereikbaar. Het zijn 'state machines'.
Een state machine workflow werkt anders dan een sequentiële workflow. Maar een aantal zaken zijn hetzelfde. Uiteindelijk moet er toch concreet werk verzet worden in de beide soorten workflows. Dat gebeurt in beide systemen door activities. Zoom je in op de states, dan zie je dat de implementatie op het laagste niveau weer bestaat uit een sequentiële workflow.
Hoewel het de moeite waard is te overwegen voor een state machine-implementatie te kiezen, vormen state machines niet in alle gevallen de beste keus. Vrijwel alle workflows zijn zowel als state machine en als sequentiële workflow te bouwen. In het ene geval vormt een state machine een betere keus, in het andere geval een sequentiële workflow. Als vuistregel mag je stellen dat state machine workflows beter begrepen worden door businessanalisten. En dat ze beter geschikt zijn voor processen waar veel interactie is met gebruikers (die soms andere ideeën hebben over de exacte volgorde van stappen in het te volgen proces). Bij een state machine workflow worden de beslissingen buiten de workflow genomen door de gebruiker, die ook de volgende stap/state bepaalt. Ze zijn complexer om te bouwen en kosten daardoor wat meer tijd.
Sequentiële workflows zijn geschikt voor processen met een 'vaste' flow. Niet de gebruiker maar de workflow is hier de baas en bepaalt de volgende stap. Ze zijn gemakkelijker te bouwen en sneller te debuggen, ook omdat je minder vrijheid hebt in het te volgen proces. Dit kan wel leiden tot een erg complexe flow. En al helemaal wanneer het proces toch niet zo sequentieel blijkt te zijn als initieel gedacht.

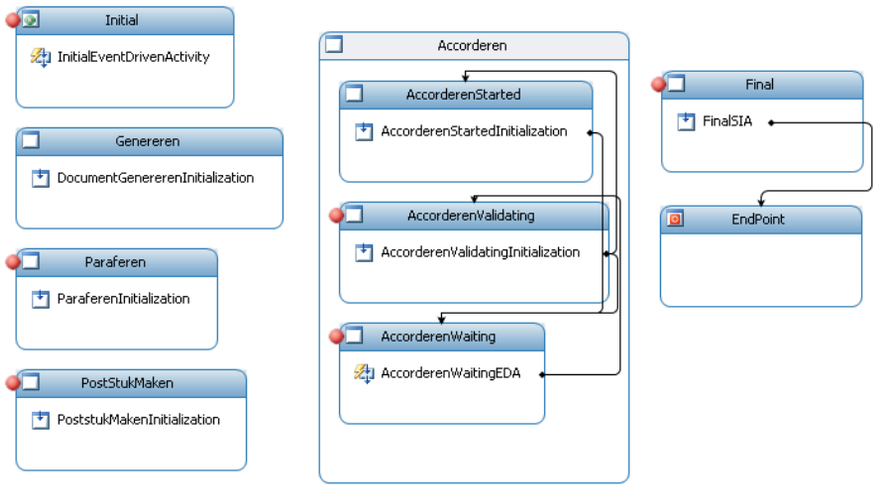
Statusovergangen dynamisch bepalen

State machine workflows kennen overgangen van de ene state naar de andere. Binnen Windows Workflow Foundation worden die geïmplementeerd met behulp van de SetState activity (niet te verwarren met de SharePoint workflow SetState activity). Iedere mogelijke statusovergang vereist zo'n SetState. In een aantal gevallen is het beter en eenvoudiger om de statusovergangen dynamisch te bepalen. Dat doe je door de sequentiële workflow binnen de state af te sluiten met een dynamische state activity. Het gaat dan om een custom activity die informatie uit de workflow of uit de database gebruikt bij het selecteren van de statusovergang. De statusovergang kan dus afhangen van business rules.
Een dergelijke situatie kwamen we in ons huidige project tegen. In een bepaalde procedure voor het verlenen van een vergunning wordt een groot aantal documententypen (content types in SharePoint) gebruikt. Aan ieder documenttype zit een workflow gekoppeld. De stappen zelf zijn daarbij gelijk voor alle soorten documenten, maar de volgorde kan verschillen. Daarbij komt nog dat niet alle stappen voor alle documenten van toepassing zijn. Zo doorloopt een bevestigingsbrief van een vergunningaanvraag achtereenvolgens de stappen Genereren, Poststuk Maken en Poststuk Verzenden. Een beschikking doorloopt Genereren, Accorderen, Tekenen, Poststuk Genereren, Poststuk Verzenden en Publiceren. Wanneer je dit met de standaard SetState activity wilt implementeren, heb je voor ieder documenttype een eigen workflow nodig. Of de workflows zouden niet te onderhouden hoeveelheden conditionele statements bevatten. Daarnaast zouden wijzigingen in de stappen voor een documenttype ook leiden tot wijzigingen in de workflow.
Met behulp van dynamische statusovergangen gaat de implementatie ineens een stuk eenvoudiger. Schema 3 toont een workflow waarbij in principe alle combinaties van Genereren, Paraferen, Poststukmaken en Accorderen voor kunnen komen. De workflow start bij de Initial state. Na wat initiële processing wordt voor de eerste maal een dynamische state activity aangeroepen. Deze activity haalt op basis van het content type gegevens op uit de database. Hierin staan de voor dit bepaalde content type de achtereenvolgend te doorlopen states. Vanuit de huidige state wordt de volgende



SCHEMA 2. DE IMPLEMENTATIE ALS EEN SEQUENTIËLE WORKFLOW

bepaald, tot aan de Final state. In die laatste state vindt nog wat afhandeling plaats, op weg naar het Endpoint, de completed state voor de workflow.
Een dynamische state activity implementeer je door in de Execute-methode van de custom activity de naam van de volgende state op te bouwen op basis van businesslogica en die vervolgens op de SetStateQueue van de workflow te zetten (zie listing 1). In de methode bepalen we eerst de volgende fase (de naam van de state in schema 3) door een aanroep van de methode GetNextFase(). Hierin wordt, door een aanroep van de businesslogica, de volgende fase bepaald. Dit gebeurt op basis van de huidige fase en het documenttype. Vervolgens wordt deze op de SetStateQueue van de workflow gezet.



SCHEMA 3. DE DYNAMISCHE STATE MACHINE WORKFLOW DIE WE GEBRUIKTEN VOOR HET VERGUNNINGVERLENINGPROCES

```
protected override ActivityExecutionStatus
Execute(ActivityExecutionContext execu-
tionContext)
{
    string nextFase = GetNextFase();
    WorkflowQueuingService wqs = (Work-
flowQueuingService)executionContext.
GetService(typeof(WorkflowQueuingSer-
vice));
    WorkflowQueue queue = wqs.GetWork-
flowQueue("SetStateQueue");
    SetStateEventArgs args = new
SetStateEventArgs(nextFase);
    queue.Enqueue(args);
    return ActivityExecutionStatus.Closed;
}
```

LISTING 1. EXECUTE-METHODE VAN DE CUSTOM ACTIVITY. OM HET DUIDELIJK TE MAKEN IS DE METHODE WAT VEREENVOUDIGD.

Workflow Foundation en SharePoint

Realiseer je dat SharePoint workflows gebruiker- en documentgeoriënteerd zijn. In de meeste gevallen wordt aan die voorwaarden voldaan en levert de samenwerking van MOSS en Workflow Foundation een krachtig gereedschap om workflows te implementeren. Er komen echter ook situaties voor dat we niet aan die de voorwaarden kunnen voldoen. Vooral als we naast de standaardfunctionaliteit ook nog andere functies in SharePoint willen hosten. Ook in die situaties bieden workflows in SharePoint een krachtig startpunt om op voort te bouwen. Zo kende het eerder genoemde vergunningverleningsysteem vanuit de SharePoint-omgeving de volgende eisen, die het onhandig zouden maken een zuivere SharePoint- en Workflow Foundation-oplossing te implementeren:

- Er bestaan verschillende vergunningverleningprocedures, met vijf hiërarchische niveaus.
- De procedures moeten te configureren zijn. Daarbij worden vaststaande Taken (kleine workflows) aan elkaar geregen tot een hoger niveau, waar vervolgens weer groepering plaatsvindt tot in het hoogste niveau.
- Delen van de workflow zijn niet op documenten van toepassing, maar op de interactie met de achterliggende database. Hiermee voldoen we niet aan het eerder genoemde uitgangspunt dat SharePoint workflows documentgecentreerd zijn.
- Een groot deel van de stappen in de procedures zijn optioneel. Op basis van de situatie kan de behandelaar besluiten dat een groep acties niet relevant is.
- De behandelaar kan een aantal stappen op verschillende niveaus gelijktijdig onderhanden hebben.

Als je het beschreven systeem helemaal met SharePoint Workflows zou willen implementeren, loop je tegen een aantal problemen aan. Hoe moeten we bijvoorbeeld het hiërarchische aspect invullen? We kunnen Workflows starten vanuit andere workflows. Maar binnen SharePoint is de WorkflowRuntime afgeschermd, waardoor local services niet gemakkelijk voor de communicatie zijn te gebruiken. Voor het configureerbaar maken van de workflows bestaan verschillende opties binnen SharePoint Workflows, die echter geen van allen echt voldoen. Zo was de eerste gedachte om Designer Hosting toe te passen. Maar dit bracht als nadeel dat de gebruiker een jaar uit de roulatie zou zijn omdat hij Sharepoint Workflows moet leren begrijpen.
Voor al deze problemen zijn wel oplossingen te vinden binnen SharePoint en Workflow Foundation. Maar daarmee rekken we de uitgangspunten zover op dat we toch niet op een standaardoplossing uitkomen. Daarnaast maken we de oplossing technisch en conceptueel erg complex. In dit soort gevallen is het beter naar de meest 'schone' oplossing te zoeken. Gebruik SharePoint Workflows dan ook waar mogelijk, maar als je zaken eenvoudiger en beter op een andere manier kan oplossen, doe dat dan.
Met dit uitgangspunt ontwierpen wij een oplossing waarbij we SharePoint Workflows inzetten voor de onderste laag van de vijf niveaus en custom code voor de bovenliggende lagen. Op deze manier lossen we alle probleempunten op buiten SharePoint Workflows om. Dit deel van de oplossing vergt ongeveer tien procent van de ontwikkeltijd. Technisch hebben we de oplossing gerealiseerd door definities en instanties van de verschillende procedures in een database te modelleren, met een aantal klassen die zorgen voor het realiseren van de genoemde functionaliteit. De onderste laag is zodanig opgedeeld dat iedere Taak overeenkomt met een SharePoint Workflow, op te vatten als een atomaire Taak. Dus negentig procent van de oplossing is nog steeds Standaard SharePoint Workflow.

Links

Walkthrough: Creating and Debugging a SharePoint Workflow Solution, http://msdn.microsoft.com/en-us/library/bb386168.aspx

How Do I: Create a State Machine Workflow for SharePoint? http://msdn.microsoft.com/en-us/office/cc514057.aspx

John Sanders (john.sanders@logica.com) en **Alfred de Weerd** (alfred.de.weerd@logica.com) zijn Software Architect bij Logica.