achtergrond

Uw organisatie staat op het punt om een nieuwe .NET applicatie in productie te nemen. De afgelopen maanden hebben uw business- en productmanagers zorgvuldig de specificaties opgesteld. De benodigde architectuur voor de applicatie en infrastructuur is uitgebreid gevalideerd en goedgekeurd door de operations manager. Vervolgens is het gedetailleerde projectplan opgesteld en zijn de componenten en webservices gebouwd en geïntegreerd. Onlangs is de applicatie functioneel getest en ook de stress-test is goed doorstaan. Niets lijkt een succesvol gebruik nog in de weg te staan.

Oplossen van applicatieproblemen

Inzet van de juiste hulpmiddelen

Ondanks deze solide voorbereiding zeggen uw ervaring (en uw intuïtie) dat zich rond deze nieuwe applicatie problemen zullen voordoen. Gebruikers zijn natuurtalenten in het invoeren van data in ongewenst formaat of in een onvoorziene volgorde. De productieomgeving wijkt per definitie af van de testomgeving, waardoor conflicten met andere applicaties zullen ontstaan. De impact van productieverstoringen door falende applicaties kan catastrofaal zijn voor uw bedrijf juiste strategie voor probleembeheersing kan dat effect minimaliseren.

De overgang naar webservices-architecturen resulteert in toenemende distributie van componenten

Problemen in Windows/.NET applicaties

Microsoft Visual Studio .NET en het .NET Framework bieden een krachtige ontwikkel- en testomgeving voor Windows-gebaseerde applicaties, vooral wanneer zwaar geleund wordt op XML Web services.

Microsoft .NET adresseert met succes veel uitdagingen rond het ontwikkelen, testen en in productie nemen van gedistribueerde applicaties. Uiteraard worden ook nieuwe uitdagingen geïntroduceerd. De overgang naar webservices-architecturen resulteert in toenemende distributie van componenten. Hierdoor neemt de complexiteit van problemen meer dan evenredig toe. Bij het analyseren van een incident in een applicatie gaat veel kostbare tijd zitten in het zoeken naar de kern van het probleem. Het vingerwijzen tussen de specialisten van het netwerk, de configuratie en de programmacode, zal u bekend voorkomen. Hoe meer eindgebruikers afhankelijk zijn van de stabiliteit en correcte werking van de applicatie, hoe groter de druk om tot een snelle oplossing te komen.

Meest voorkomende problemen

Performance-problemen

In elke applicatieomgeving doen zich performance problemen voor. Analyse van deze problemen is complex en daarmee tijdsintensief en kostbaar. Enkele oorzaken van performanceproblemen zijn:

- *Inefficiënte applicatiecode*, zoals frequente 'excepties', (te) veel grote objecten, inefficiënte database calls;
- *Memory-gebruik:* het niet vrijgeven van niet langer gebruikte objecten, te veel memory toewijzen per 'request', onjuiste definitie van maximaal aantal 'worker' en 'IO threads;
- *Onjuiste applicatie-settings:* ontoereikende session-state provider, uitschakeling van 'buffering' voor een Web Forms-pagina;
- *Gebruik van 'legacy' Windows-code,* zoals de verwerking van grote hoeveelheden data van COM/COM+ naar managed .NET objecten;
- *Infrastructurele problemen:* bijvoorbeeld slechte netwerk-response.

Functionele problemen
Het Microsoft .NET Framework is ontwikkeld om veel programmeeractiviteiten weg te nemen bij ontwikkelaars, met name bij geheugentoewijzing en beveiliging. Fouten in de programmacode kunnen echter nog steeds voorkomen. Enkele voorbeelden hiervan zijn:

- Onjuiste business logica, zoals onjuiste berekening van rentetarieven;
- 'Thread deadlocks';
- Een ernstige fout waardoor de applicatie hangt of crasht.

Configuratieproblemen
Ondanks het feit dat .NET veel problemen rond de 'DLL hell' heeft verholpen, zijn de problemen rond de applicatieconfiguratie niet verdwenen. Enkele veel voorkomende configuratieproblemen zijn:

- Onvoldoende permissie om resources te benaderen: bijvoorbeeld onvoldoende rechten om een bepaalde directory te benaderen.
- Incompatibele componenten: bijvoorbeeld onjuiste versies van COM/COM+ objecten.
- Onjuiste applicatie settings in de .NET configuratiebestanden.
- Conflicten met andere applicaties: bijvoorbeeld een antivirus tool die zorgt dat webapplicaties herhaaldelijk herstarten.

Fouten van gebruikers
.NET applicaties worden ontwikkeld om eindgebruikers te dienen. Deze gebruikers zijn ook de oorzaak van veel problemen in de productie. Een aantal van deze fouten is snel op te lossen, bij andere fouten kan het dagen of weken kosten om uit te zoeken welke exacte handeling de oorzaak is van de problemen.

AANPAK IN PRODUCTIEOMGEVINGEN Binnen een gecontroleerde testomgeving zijn in het algemeen voldoende hulpmiddelen voor probleemanalyse voorhanden. In een productie-omgeving daarentegen is het zeer lastig om inzicht te krijgen in de oorzaak van een verstoring. In deze fase is inzicht in alle aspecten van de applicatie noodzakelijk om het probleem aan te kunnen pakken. De hulpmiddelen die tijdens het testen ingezet konden worden, zoals Visual Studio-debuggingtools en profilers, zijn nauwelijks geschikt voor analyse in productieomgevingen. Hoe kunnen deze applicatieproblemen in productieomgevingen dan worden opgelost? Hierna vindt u een viertal benaderingen, twee traditionele en twee opkomende:

1. Interne Logging
Het toevoegen van interne logging-regels aan software is zo oud als het programmeren zelf. De applicatiecode wordt dan 'verrijkt' met printfuncties. Vervolgens worden tijdens executie van het programma de printregels weggeschreven naar een logbestand. Vanwege de impact op de performance vindt deze logging alleen plaats wanneer de applicatie in een speciale 'mode' draait. Deze veelvuldig toegepaste methode voor het debuggen van een applicatie heeft zo zijn tekortkomingen. Gemiddeld vijftien procent van de kostbare ontwikkeltijd gaat verloren aan het toevoegen, onderhouden en verwijderen van debug-code. Daarnaast blijkt dat in de huidige gedistribueerde omgevingen slechts een gedeelte van de problemen gerelateerd is aan onjuiste programmacode. Steeds vaker blijken configuratiefouten, fouten in *third party* componenten en dergelijke de oorzaak te zijn van een incident. 'Internal tracing' is per definitie applicatie-intern gericht en biedt dus onvoldoende inzicht in de processen buiten de grenzen van de applicatiecode.

Ondanks de beperkte waarde bij analyse is intern loggen de meest toegepaste methode voor het oplossen van problemen. Integrated Development Environments (IDE's) leken dit belangrijke aspect in eerste instantie te negeren. Ontwikkelaars moesten het wiel nog steeds opnieuw uitvinden. Met de toevoeging van 'Trace Object' werd Visual Studio .NET de eerste Microsoft-IDE die problemen kan adresseren. De .NET Trace maakt het mogelijk om interne logging toe te passen, zonder de printfuncties te hoeven verwijderen bij het in productie gaan van de applicatie. 'Trace' kan buiten de applicatie om worden geconfigureerd door aanpassing van de Config file. Het werkt vervolgens met 'thick clients' en ASP.NET applicaties.

> Gemiddeld vijftien procent van de kostbare ontwikkeltijd gaat verloren aan het toevoegen, onderhouden en verwijderen van debug-code

2. Monitoren van de executie van de applicatie
Een groeiend aantal hulpmiddelen is beschikbaar voor het monitoren van de applicatie zonder interne logging. De meeste tools richten zich op het monitoren van performance en/of beschikbaarheid. Incidenten worden snel opgespoord en voor potentiële problemen wordt pro-actief gewaarschuwd. Voor diepgaande 'root-cause analysis' zijn deze tools minder geschikt, omdat niet voldoende inhoudelijke data van het probleem gelogd wordt.

Monitoring-oplossingen zijn in probleemsituaties afhankelijk van het correleren van een enorme hoeveelheid meetgegevens. In praktijk blijkt dat na probleemidentificatie vaak een tweede proces of project gestart moet worden met andere hulpmiddelen. In essentie

zijn deze oplossingen zeer goed in het melden dat een probleem is ontstaan of gaat ontstaan, maar minder goed in het verklaren waarom en waardoor.

3. Hercreëren van problemen in het lab

Applicatie-monitoring en interne logging geven het team van probleemoplossers onvoldoende inzicht in de kern van een productieprobleem. Een logische vervolgstap is dan het hercreëren van het probleem in het lab. Indien hetzelfde probleem zich daar ook voordoet, kan de analyse starten. Deze activiteit is complex en tijdverslindend. Het repliceren van de productieomgeving en het zoeken naar het juiste scenario om het probleem te hercreëren, is een hele opgave – en blijkt soms zelfs onmogelijk.

Het oplossen van applicatieproblemen, vooral in de fasen na ontwikkeling, blijft een uitdaging

4. Analyse met 'live recording' technologieën

Een relatief nieuwe trend is het inzetten van 'low-impact' monitoring-technologieën. Dergelijke oplossingen werken volgens hetzelfde principe als een vluchtrecorder of 'black box' in een vliegtuig. Continu wordt een verscheidenheid aan gegevens opgenomen en gecorreleerd – geheel in 'real time'. Na de landing of crash kan de vluchtrecorder alle relevante gebeurtenissen van de vlucht afspelen. Specialisten kunnen direct starten met de diepgaande analyse van de falende systemen. Ditzelfde concept wordt nu ook toegepast bij het oplossen van verstoringen in applicaties. Gebruikmakend van nieuwe technieken worden alle belangrijke gegevens van de applicatie en de applicatieomgeving vastgelegd, zoals:

- code executie;
- applicatie- en systeemconfiguratie;
- 'events';
- performancegegevens;
- excepties;
- exacte schermhandelingen van de gebruikers.

Wanneer een probleem ontstaat kan de betreffende opname direct worden afgespeeld, zelfs zonder dat de gebruiker daarvoor het programma behoeft te verlaten. Vervolgens wordt het ontstaan van het werkelijke probleem stap voor stap geanalyseerd. Alle relevante gegevens zijn daarbij volledig synchroon in de tijd inzichtelijk. Dit *zonder* het hercreëren van het probleem in het lab en *zonder* toevoeging van debug- en tracecode in de applicatie. De ervaring leert dat bedrijven die bij het oplossen van incidenten gebruik maken van deze live-recording technologieën, gemiddeld zestig procent op het totale proces van probleemoplossing besparen.

CONCLUSIE Visual Studio .NET en het Microsoft .NET Framework realiseren een significante productiviteitsverbetering in de ontwikkelfase van een applicatie. Het oplossen van applicatieproblemen, vooral in de fasen na ontwikkeling, blijft een uitdaging. Ontwikkelaars ontkomen niet aan het hercreëren van problemen in het lab voordat debuggers en profilers überhaupt kunnen worden ingezet. De werkelijke analyse moet dan nog starten. De meeste monitoring-oplossingen leveren high-level statistische performance gegevens, maar schieten tekort bij gedetailleerde 'root-cause analysis'. Ontwikkelaars hebben de keuze: hercreëren van het probleem, toevoegen van interne logging of inzet van live-application recorders. Dit laatste is favoriet vanwege de significante productiviteitsverbetering en snellere time to resolution.

Marcel Meeuwisse is sinds 1988 werkzaam in de IT. Na bij een aantal kleine softwarehuizen werkzaam te zijn geweest heeft hij in 1995 de overstap gemaakt naar Oracle Nederland BV (Sales MT). Na een korte stop bij Software AG Nederland (MT) is hij in 2004 toegetreden tot The Future Group als Managing Partner, verantwoordelijk voor de distributie van het product "Appsight" van Identify Software in Nederland. E-mail: meeuwisse@the-future-group.com