Vaak wordt mij gevraagd: als ik een systeem bouw met deze 4 GL en dit RDBMS, op hoeveel uur per functiepunt moet ik dan rekenen. Bij de 4 GL en het RDBMS worden dan namen genoemd waarmee ik enige tijd heb gewerkt en waarvan de vrager dus veronderstelt dat ik er veel over kan zeggen. Mijn antwoord is dan steevast: tussen de 2 en de 20.

achtergrond

Functiepuntanalyse geanalyseerd

Succesfactoren voor efficiënte ontwikkelomgeving

Er zijn veel omstandigheden die bepalen of je kunt spreken van een efficiënte ontwikkelomgeving of juist niet. Zie hiervoor ook mijn artikel "Ontwikkelstraat, hergebruik door inzet van architectuur"[2]. Voor de theorie van de functiepuntanalyse is gebruik gemaakt van (versie 2.0 van) de publicatie "Definities en telrichtlijnen voor de toepassing van Functiepuntanalyse" van de NESMA, de Nederlandse Software Metrieken Gebruikers Associatie", gevestigd te Zeist[1].

ALGEMENE OPMERKINGEN Vierde generatie talen (4 GL's, zie ook kader) zijn speciaal ontwikkeld om gegevensstructuren te onderhouden en te rapporteren. Het ligt dan ook voor de hand dat deze functionaliteit snel, eenvoudig en betrouwbaar gebouwd kan worden, dus met een laag aantal uren per functiepunt.

Hiervan sterk afwijkende functionaliteit zal met een 4 GL moeilijk of soms in het geheel niet te bouwen zijn. In dit geval kan men (voor enkele functies of routines of soms voor het gehele systeem) zijn toevlucht nemen tot een alternatieve taal, bijvoorbeeld:

- Cobol bij een personeelssysteem om het salarisverwerkingsprogramma te bouwen;
- C om een specifieke routine te bouwen;
- SQL om database bewerkingen uit te voeren.

Als toch geprobeerd wordt om deze situaties met de 4 GL te bouwen zal dit meestal leiden tot:

- Zeer ingewikkelde programmatuur met kans op veel fouten;
- Een sterke overschrijding van de planning, dus van de uren per functiepunt;
- Een zeer slechte performance.

Fouten zullen leiden tot herstelacties. Als de tijd voor deze herstelacties meegeteld wordt zal dit nogmaals leiden tot een verhoging van het aantal uren per functiepunt.

EENVOUDIGE EN MOEILIJKE FUNCTIES Zeer eenvoudige onderhoud- en opvraagfuncties zijn in de meeste 4 GL's snel te bouwen. Behalve een gedegen gegevensmodel is meestal niet meer nodig dan een scherm lay-out. De 4 GL verzorgt de rest zoals de IO, de invoercontroles en de opmaak van velden, mits in het gegevensmodel vastgelegd.

Voor additionele functionaliteit, zoals invoercontroles (die niet in het gegevensmodel zijn vastgelegd) of berekeningen, moet in de 4 GL evenveel of soms zelfs meer coding geschreven worden dan in een 3 GL. Hierdoor komen we tot een volgende mogelijke verdeling tussen de verschillende soorten functies zoals te zien in tabel 1.

> Eén of twee dagen tijdverlies per probleem is bij een 4GL heel normaal

Volgens de standaard FPA werkwijze moeten voor onderhoudsfuncties 3 functies geteld worden, namelijk 1 voor het inbrengen, 1 voor het wijzigen en 1 voor het verwijderen. Bij een 4 GL wordt dit meestal in 1 functie gecombineerd. Er behoeft ook geen extra coding geschreven te worden, de 4 GL bevat al alle coding om de inserts, updates en deletes in de database uit te voeren. Hierdoor krijg je veel functiepunten (3x3=9) voor ongeveer 1 dag werk, dus ongeveer 1 uur per functiepunt. Als het verwijderen in een functie niet is toegestaan moet juist extra werk worden gedaan om dit te verbieden! Dus meer werk (1,1 dag en minder functiepunten 2x3=6) levert meteen een verhoging van het aantal uren per functiepunt van 1 naar 1,5 ofwel 50%!

Hieruit blijkt al direct dat het aantal uren per functiepunt voor een project afhankelijk is van het soort functies:

- Zijn er veel zeer eenvoudige functies en weinig (zeer) moeilijke functies te bouwen, dan komt het gemiddelde aantal uren per functiepunt dicht bij de 2 te liggen;
- Zijn er veel zeer moeilijke uitvoer of batch onderhoudsfuncties en weinig eenvoudige functies, dan zal het gemiddelde dicht bij de 10 komen.

KWALITEIT VAN HET ONTWERP Als er in het ontwerp van een functie nog een paar problemen zijn, dan zal de programmeur bij de ontwerper te rade moeten gaan. De vertraging die hierbij optreedt kan snel tot enkele uren oplopen.

De hoeveelheid werk zal ook stijgen indien de ontwerper een functie zodanig heeft beschreven dat deze niet gemakkelijk in de betreffende 4 GL te bouwen is. Een kleine, functioneel niet relevante wijziging in bijvoorbeeld de scherm lay-out kan de functie beter bouwbaar kan maken.

De grootste ontwerpproblemen doen zich voor in het gegevensmodel. Omdat 4 GL's veel zaken automatisch doen aan de hand van het (fysieke) gegevensmodel zullen fouten in dit model (in veel functies) leiden tot extra coding om:

- de automatische functionaliteit uit te schakelen en om de gewenste functionaliteit via procedurele coding in te brengen.

Al deze problemen zullen leiden tot (nodeloos) veel coding en complexe programmatuur en daardoor waarschijnlijk ook tot extra fouten in de programmatuur. Deze fouten zullen weer leiden tot herstelacties en aldus tot nog meer extra werk en veel meer uren per functiepunt.

Soort functie	Dagen	Soort functie en Functiepunten	Uren per functiepunt
Zeer eenvoudig	0,5	Opvraag: 3 fp	1,33
Eenvoudig	1	Uitvoer: 4 fp	2
		Onderhoud: 3x3 fp	1
Normaal	2	Uitvoer: 5 fp	3,2
		Onderhoud: 3x4 fp	1,3
Moeilijk	5	Uitvoer: 7 fp	6
		Onderhoud: 3x6 fp	2,2
Zeer moeilijk	10 of meer	Uitvoer: 7 fp	10 of meer
		Onderhoud batch: 6 fp	13 of meer

TABEL 1. Toelichting op het aantal functiepunten voor de onderhoudsfuncties

Een combinatie van bovenstaande ontwerpproblemen zal snel tot een verdubbeling of zelfs verviervoudiging van het aantal uren per functiepunt leiden. Ik durf zelfs te stellen dat elk soort probleem leidt tot een verdubbeling van het aantal functiepunten ten opzichte van het vorige probleem, dus uitgaande van 2 uur per functiepunt geeft:

- Het eerste probleem (bijvoorbeeld een slecht ontwerp) een verdubbeling: 4 uur/fp;
- Het tweede probleem (slecht gegevensmodel) weer een verdubbeling: 8 uur/fp;
- Het derde probleem (geen goede ontwikkelstraat) weer een verdubbeling: 16 uur/fp;
- Het vierde probleem (functionaliteit eigenlijk niet geschikt voor de 4 GL) weer een verdubbeling: 32 uur/fp; etc.

NB: Indien de ontwerper een aantal functies niet heeft gespecificeerd, maar deze wel gebouwd moeten worden, bijvoorbeeld een menufunctie of het opschonen van de database, dan zal zowel de hoeveelheid werk als het aantal functiepunten stijgen ten opzicht van het oorspronkelijk gecalculeerde. Het aantal uren per functiepunt behoeft hierbij niet te stijgen. Bij een goed ingerichte ontwikkelstraat zijn veel van dit soort algemene functies al standaard beschikbaar.

> Bij een traditioneel project is nooit meer winst te behalen dan het 'terugverdienen' van de bouwkosten

FOUTEN IN DE 4 GL Fouten komen in de 4 GL producten veel vaker voor dan in een 3 GL (C of Cobol compiler). Als de programmeur zich in bochten moet

wringen om de gewenste functionaliteit te bouwen en dus ook "nog niet zo vaak betreden paden in de 4 GL bewandelt", dan zal hij ook vaker tegen fouten in de 4 GL oplopen. Deze leiden direct tot veel tijdverlies. Eén of twee dagen tijdverlies per probleem is heel normaal; soms is men al meer dan een dag kwijt aan het zoeken naar de oorzaak van het probleem voordat men zich realiseert dat het een 4 GL probleem is.

Er zijn ook andere situaties waarin de 4 GL uit de bocht vliegt. Een probleem in een bekende 4 GL was eens het opvragen in een inquiry of uitvoerfunctie van meer dan 216 (65535) records. Indien deze problemen omzeildt moeten worden, kost dit erg veel aandacht en tijd.

EENMALIGE ACTIVITEITEN Bij het opstarten van een 4 GL project moet een groot aantal zaken geregeld worden:

- Opzetten van de ontwikkelomgeving (hardware, netwerk, DBMS, de 4 GL);
- Beschrijving van de standaard functietypen en andere standaards;
- Het schrijven van standaard routines zoals voor foutafhandeling, etc.

Indien deze omgeving en de standaards (de ontwikkelstraat – *zie ook literatuurlijst, 2*) reeds beschikbaar is of ongewijzigd overgenomen kan worden, bijvoorbeeld van een ander project bij dezelfde klant, dan behoeft hiervoor niets geteld te worden. Anders kan deze overhead tot enkele manmaanden oplopen. Voor een klein project (één of enkele maanden) betekent dit (ook) al snel een verdubbeling van de totale hoeveelheid werk. Op een groot project is de invloed mogelijk slechts enkele procenten.

Een andere eenmalige activiteit per project is het inbrengen van het gegevensmodel door de DBA. Omdat binnen de FPA methode elke tabel ook enkele (7 of 10) functiepunten toegewezen krijgt, valt dit binnen het reeds in de FPA methode geraamde werk. Indien op het project programmeurs worden ingezet die nog geen enkele ervaring met de 4 GL hebben, dan dient men rekening te houden met een extra inwerktijd van enkele weken per programmeur tot enkele maanden voor een specialist (DBA, architect, bouwer van de ontwikkelstraat).

> Het aantal uren per functiepunt voor een project afhankelijk is van de complexiteit van de functies

Model driven en screen driven 4 GL's.

- Bij Model driven 4 GL's (bijvoorbeeld Uniface en Cool-Gen) staat het gegevensmodel centraal in de ontwikkelomgeving. Er kan veel functionaliteit, zoals controles, opmaak, bewerkingen en berekeningen, in het model worden opgenomen. Elke functie die een entiteit uit dit centrale model oppakt, neemt automatisch deze centraal gedefinieerde functionaliteit over.
- Bij Screen driven 4 GL's (bijvoorbeeld Powerbuilder of Visual Basic) staan de schermen centraal. In de schermen kan een "data-window" geplaatst worden welke aan een tabel in een database wordt gekoppeld. Veel functionaliteit, die niet in een centrale database gedefinieerd kan worden zoals opmaak, moet in elke functie opnieuw worden geprogrammeerd.

Bedenk dat de bouwfase bij een model driven 4 GL start met het inbrengen van dit centrale gegevensmodel in het ontwikkeltool (de 4 GL). Dit wordt gedaan door een klein aantal (soms 1) DBA's. Omdat hier veel meer functionaliteit vastgelegd kan worden dan in de tabeldefinities van het RDBMS, zal ook de tijd die nodig is om dit (goed) te doen significant groot zijn. Pas daarna kunnen de programmeurs (efficiënt) aan de slag om hun functies te bouwen. Als hiermee geen rekening gehouden is in de planning kan hier een probleem ontstaan:

- De programmeurs moeten wachten, dus slecht voor de planning.
- De programmeurs beginnen als het model nog niet helemaal klaar is en later moeten de wijzigingen in het model nog in de programma's worden verwerkt.

Bedenk ook dat in de beginfase slecht met een kleine groep DBA's aan dit model gewerkt kan worden. Dit betekent voor de totale planning mogelijk een laag aantal uren maar wel een langere doorlooptijd. Het is dan ook van belang om deze DBA's zo vroeg mogelijk in het project te laten instromen.

TECHNISCH ONTWERP EN DOCUMENTATIE Voor een "recht-toe-recht-aan" 4 GL onderhoud- of opvraagfunctie is geen technisch ontwerp nodig, althans indien een ontwikkelstraat met goede standaards en sjablonen beschikbaar is (*zie ook literatuurlijst, 2*). Voor een ingewikkeld programma zal de programmeur eerst eens goed moeten nadenken hoe hij dit probleem gaat aanpakken (= technisch ontwerp maken). Men kan vervolgens:

- Het programma bouwen;
- Het technisch ontwerp weggooien;
- De gewenste documentatie genereren uit de coding.

Het alternatief om "handmatig" de technische documentatie over alle functies van het systeem bij te hou-

den kan bij eenvoudige functies leiden tot meer werk aan het onderhoud van de documentatie dan aan het bouwen van de functies zelf, waardoor ook het aantal uren per functiepunt weer stijgt (verdubbelt).

Bouwkosten versus de totale projectkosten Indien in een optimaal bouwproject het aantal uren per functiepunt gereduceerd kan worden van bijvoorbeeld 10 (voor een inefficiënt 4 GL of traditioneel Cobol project) naar 2, dan betekent dit niet dat de kosten van het gehele project ook met een factor 5 kunnen worden gereduceerd. Voor veel andere aspecten van het totale project, bijvoorbeeld het herinrichten van het proces dat de door het nieuwe systeem wordt ondersteund, het opleiden van de gebruikers, de definitiestudie fase, etc. heeft deze reductie geen invloed.

Sommige kosten kunnen mogelijk zelfs hoger worden, bijvoorbeeld de aanschaf van hardware in verband met het vaak hogere CPU en IO gebruik van de 4 GL.

Speciale aandacht verdient ook het functioneel ontwerp. Er moet goed gekeken worden naar een goede aansluiting van het functioneel ontwerp op de bouw. Als de bouw sterk is gestandaardiseerd, een onderhoudsfunctie er altijd hetzelfde uitziet, dan is het zeer ongewenst dat een ontwerper in elke functie deze standaard werking beschrijft (*zie ook literatuurlijst, 2*):

- Het is veel werk en later nog veel meer onderhoud.
- Het moet gelezen worden door de bouwer om te zien of er in al die tekst toch niet nog een stukje extra functionaliteit vermeld staat.
- Het kan zijn dat het ontwerp op een enkel punt onnodig afwijkt van de standaard met alle gevolgen voor de efficiëntie van het bouwproces en ook voor de gebruiksvriendelijkheid van de functie.

Het functioneel ontwerp voor een onderhoudsfunctie voor een tabel, waarvan de werking al geheel in de standaards is vermeld en waarbij de betreffende tabel reeds geheel in het gegevensmodel is uitgewerkt, heeft aan een paar regels (hooguit een halve pagina A4) tekst voldoende. Dit betekent dus ook een reductie op de hoeveelheid werk in de functioneel ontwerp fase. Wordt in deze fase een "oude 3 GL werkwijze" gehanteerd met vele pagina's tekst per functie dan zal de winst op het totale project niet groot zijn omdat de ontwerpfase mogelijk meer tijd gaat kosten dan de bouwfase.

Samenvatting Als alle omstandigheden mee zitten, is in een 4 GL omgeving een project te realiseren met ongeveer 2 uur per functiepunt. De volgende omstandigheden kunnen echter elk leiden een verdubbeling of erger, waardoor het totale aantal uren per functiepunt op een veelvoud van twee uitkomt en mogelijk zelfs in de buurt van de 32 uren of meer komt:

- Weinig eenvoudige onderhoud- en opvraagfuncties en veel moeilijke functies;

> Er moet goed gekeken worden naar een goede aansluiting van het functioneel ontwerp op de bouw

- Het ontbreken van een ontwikkelstraat;
- Een slecht of niet op de 4 GL en ontwikkelstraat afgestemd ontwerp;
- Een klein project met veel overhead;
- Een overdaad aan handmatig bijgehouden documentatie;
- In de 4 GL onervaren medewerkers;
- Verkeerde beslissingen bij het bouwen van speciale functionaliteit.

Tot slot Als de bouwkosten bij een traditioneel project 50 % van de totale projectkosten vormen, dan is met de meest efficiënte bouwwijze (theoretisch dus 0 uren per functiepunt) toch nooit meer winst te behalen dan deze helft van de projectkosten.

Literatuur

1 Definities en telrichtlijnen voor de toepassing van Functiepuntanalyse, Nederlandse Software Metrieken Gebruikers Associatie, 1996

2 Loonen, Ontwikkelstraat, hergebruik door inzet van architectuur. Software Release 2000/7,8

3 Loonen, Mutatierapportage, de tijdgeest van de database. Database Magazine 1999/3.

4 Heemstra, F.J. en Gerlof, P.A.M. (1992), Schattingstechnieken voor IT projecten, Cap Gemini Publishing, Rijswijk, ISBN: 90 71996 65 4.

Toon Loonen is als consultant werkzaam bij Cap Gemini Ernst & Young. Hij heeft ontwikkelstraten opgezet voor diverse 4 GL/RDBMS omgevingen. Hij is bereikbaar via e-mail: toon.loonen@cgey.nl of toon.loonen@inter.nl.net.