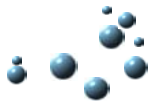


Aanpak efficiëntietesten

White paper

Auteur	Roland Mees
Versie	1.0 Definitief
Versiedatum	28 mei 2004

Inhoudsopgave

Lijst van figuren

Lijst van tabellen

1 INTRODUCTIE

Performancetesten, load- en stresstesten en efficiëntietesten zijn onderwerpen die steeds meer aandacht krijgen. Zeker nu internet belangrijker wordt als communicatiemiddel tussen bedrijven en klanten, en de concurrentie (juist op internet) groot is.
Responsetijden spelen een grote rol bij het klikgedrag van gebruikers. Slechte responsetijden zijn voor 85% van de internetgebruikers een reden om websites te vermijden [Dekkers & van der Schaaf, 2001]. Onderzoek heeft uitgewezen dat het verbeteren van de responsetijd van één seconde al leidt tot een afname van het aantal gebruikers dat de website voortijdig verlaat van zo'n 30-70% [Dekkers & van der Schaaf, 2001].

In de literatuur wordt door verschillende auteurs aandacht besteed aan testen op het gebied van efficiëntie, maar zij blijven nog teveel op globaal niveau: de artikelen en boeken missen een praktische invulling. Zie onder andere: [Anonymus, 2002; Hoeben & Sterk, 1998];

Behoefte aan een handleiding is er zeker. Dit white paper geeft, vanuit eigen ervaringen, een invulling aan de aanpak van efficiëntietesten.

Leeswijzer
Om een referentiekader te geven, wordt in hoofdstuk 2 het ISO/IEC 9126 referentiemodel besproken. Ook wordt in dit hoofdstuk de relatie gelegd met TMap, één van de teststandaarden in Nederland [Pol et al, 1999].

Hoofdstuk 3 beschrijft de aanpak van efficiëntietesten. De aanpak bestaat uit verscheidene stappen, elke stap wordt uitgelegd en met praktijkvoorbeelden toegelicht.

In hoofdstuk 4 wordt kort ingegaan op monitoring.

Een verklarende woordenlijst en een literatuurlijst zijn in de bijlagen opgenomen.

Doelgroep
Het document is vooral - maar niet alleen - bedoeld voor testers die een efficiëntietest gaan voorbereiden en uitvoeren.

2 ISO/IEC 9126, EFFICIËNTIE EN TMAP

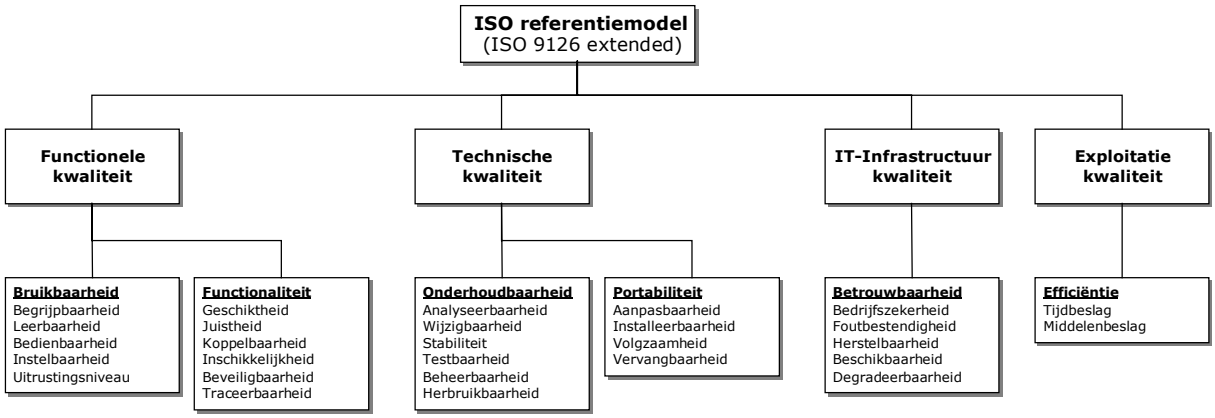
2.1 INLEIDING

In dit hoofdstuk wordt het kader geschetst waarin de aanpak voor efficiëntietesten zich afspeelt: het ISO/IEC 9126 referentiemodel, en TMap. Daarnaast wordt de plaats van efficiëntietesten in het totale testtraject aangegeven.

2.2 ISO/IEC 9126

Het ISO/IEC 9126 model, en het daarop gebaseerde Extended ISO-model in het bijzonder, biedt een standaard raamwerk voor kwaliteitseisen [ISO/IEC, 2000]. Het model gaat uit van een zestal hoofdeigenschappen (verdeeld over vier aandachtsgebieden) voor software kwaliteit. Elke hoofdeigenschap is verdeeld in een aantal deeleigenschappen.
De onderkende **aandachtsgebieden** en *hoofdeigenschappen* zijn (zie Figuur 1):

- **Functionele kwaliteit**
  - *Functionaliteit*
  - *Bruikbaarheid*
- **Technische kwaliteit**
  - *Onderhoudbaarheid*
  - *Portabiliteit*
- **IT-infrastructuur kwaliteit**
  - *Betrouwbaarheid*
- **Exploitatie kwaliteit**
  - *Efficiëntie*



Figuur 1. Het ISO 9126 model (uitgebreide versie). Bron: [ISO/IEC, 2000]

2.3 AANDACHTSGEBIED EXPLOITATIE KWALITEIT – HOOFDEIGENSCHAP EFFICIËNTIE

De hoofdeigenschap efficiëntie richt zich op de relatie tussen het prestatieniveau van een softwaresysteem en de door dat systeem gebruikte hoeveelheid aan middelen.
De eigenschap bestaat uit een tweetal deeleigenschappen:

- Tijdbeslag
- Middelenbeslag

2.3.1 Tijdbeslag

De eigenschap Tijdbeslag beschrijft de mate waarin het (software-)systeem tijd nodig heeft om te reageren op invoer of om transacties te verwerken (en de eventuele beïnvloeding door grote volumes). Denk bijvoorbeeld aan doorlooptijden van batchprocessen, responsetijden van interactieve systemen, verwerkingstijden van taken/processen, etc.

2.3.2 Middelenbeslag

De eigenschap Middelenbeslag beschrijft de mate waarin het (software-)systeem een beroep doet op beschikbare operationele middelen, zoals: processorcapaciteit, netwerk-"throughput", bandbreedte, geheugengebruik, WebSphere resources, Database-calls, schijfgebruik, printen, nabewerken, etc.

Nota bene
De term efficiëntietesten wordt in dit document gebruikt als verzamelnaam voor performancetesten, load- en stresstesten, en volumetesten.

2.4 RELATIE MET TMAP

TMap is een standaard in Nederland geworden, en onderkent een 17-tal kwaliteitsattributen [Pol, et al, 1999]. Twee daarvan hebben betrekking op het aspect efficiëntie:

- Performance
- Zuinigheid

2.4.1 Performance

Het kwaliteitsattribuut Performance richt zich op de snelheid waarmee het informatiesysteem interactieve en batchtransacties afhandelt.

2.4.2 Zuinigheid

Het kwaliteitsattribuut Zuinigheid richt zich op de verhouding tussen prestatieniveau van het systeem (uit te drukken in het transactievolume en de totale snelheid) en de hoeveelheid resources die daarvoor gebruikt worden (bijv. CPU-belasting, geheugenbelasting, netwerkbelasting, etc.).

2.5 PLAATS VAN EFFICIËNTIETESTEN IN HET TOTALE TESTTRAJECT

Het uiteindelijke doel van een efficiëntietest is het verkrijgen van inzicht in het te verwachten gedrag van het informatiesysteem in productie. Om dat gedrag zo nauwkeurig mogelijk te kunnen bepalen is het belangrijk om de test uit te voeren op een omgeving die sterk lijkt op, of identiek is aan de productieomgeving. In veel bedrijven komt de acceptatieomgeving hiervoor in aanmerking.

Meettraject
Het woord efficiëntie*test* suggereert dat het hier uitsluitend om een testtraject gaat. Efficiëntie*meting* zou beter op zijn plaats zijn, er worden vooral metingen uitgevoerd:

- metingen naar de hoogte van de responsetijden;
- metingen naar de mate van CPU-belasting van de server;
- metingen naar de netwerkbelasting, etc.

Dit vereist ook een andere instelling van de tester: hij/zij is vooral onderzoeker.

3 AANPAK EFFICIËNTIETESTEN

3.1 INLEIDING

De aanpak voor efficiëntietesten bestaat uit een aantal activiteiten, die hieronder worden opgesomd. De aanpak volgt de fasering van TMap, maar voegt de TMapfasen *voorbereiding* en *specificatie* samen tot één fase: *testvoorbereiding*.
In de volgende paragrafen wordt elke activiteit apart beschreven en toegelicht met een praktijkvoorbeeld.

Planning & Beheer
- Uitvoeren technische intake
- Schrijven plan van aanpak (testplan)
- Voeren van overleg
- Schrijven van voortgangsrapportage
- etc.

Testvoorbereiding
- Infrastructuur in kaart brengen
- Vaststellen verwacht gebruikersgedrag
- Ontwerpen gebruikersscenario's
- Ontwerpen belastingmodel
- Ontwerpen testinfrastructuur en meetpunten
- Definiëren testgegevens
- Schrijven testontwerp
- Inrichten testinfrastructuur, meetpunten en testgegevens
- Ontwikkelen emulatiesoftware
- Ontwikkelen fysieke testscripts
- Samenstellen draaiboek

Testuitvoering
- Uitvoeren testruns
- Analyseren resultaten

Testafronding
- Schrijven testrapport

3.2 PLANNING EN BEHEER

In aanloop naar een (mogelijk) testtraject wordt eerst een haalbaarheidsonderzoek uitgevoerd. Deze technische intake heeft als doel vast te stellen of het te gebruiken testtool overweg kan met de te testen applicatie. Op basis van dat onderzoek wordt besloten óf er een efficiëntietest uitgevoerd kan worden, en zo ja, hóe. Mocht het testtool niet overweg kunnen met de applicatie, dan zal er een andere oplossing gevonden moeten worden, als men toch een efficiëntietest wil uitvoeren.

Na de technische intake wordt het testplan geschreven. Het schrijven van een dergelijk plan van aanpak en de overige "projectleider activiteiten" wordt als bekend verondersteld, en wordt hier niet verder toegelicht.

3.3 TESTVOORBEREIDING

De testvoorbereidingsfase richt zich op de inhoud van de uit te voeren test. Een groot deel van de verzamelde informatie wordt vastgelegd in een testontwerp. Dit ontwerp is tevens een (één van de) communicatiemiddel(en) met de opdrachtgever(s).

3.3.1 Infrastructuur in kaart brengen

Bij de efficiëntietest wordt de gehele keten in beschouwing genomen. Om een goed inzicht te krijgen in deze keten worden de volgende onderdelen in kaart gebracht:

- de infrastructuur (testomgeving en productieomgeving);
- het te testen systeem / de te testen systemen;
- de raakvlakken met andere, aanpalende systemen.

In veel gevallen wordt, al doende, ook duidelijk hoe de beheerorganisatie eruit ziet, en wie waarvoor verantwoordelijk is.

Bij het inventariseren van de infrastructuur wordt zowel de testomgeving als de productieomgeving onderzocht. Zo komen snel verschillen aan het licht, en kunnen op voorhand mogelijke bottlenecks worden aangewezen.

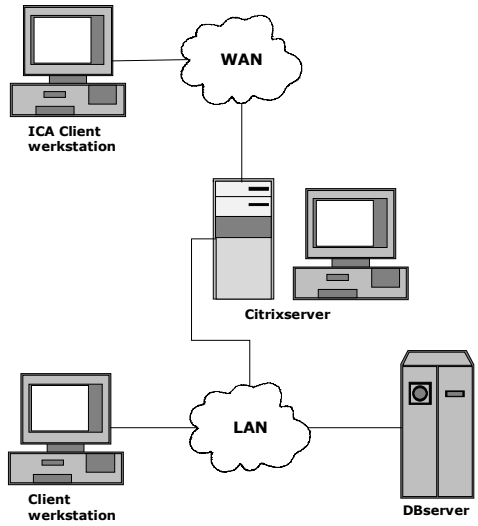
Het resultaat van deze activiteit is een tekening van de (test-)infrastructuur met een bijbehorende beschrijving.

Voorbeeld infrastructuur in kaart brengen

De infrastructuur bestaat uit de volgende onderdelen, zie Figuur 2:

- een werkstation;
- een WAN;
- een Citrixserver;
- een LAN;
- een database server.

Op het werkstation is een ICA-client geïnstalleerd. Deze ICA client heeft via het WAN contact met de Citrixserver waarop ook de applicatie is geïnstalleerd. Deze applicatie communiceert vervolgens via een ODBC-koppeling met de Oracle-db die op de Server is geïnstalleerd.
Naast dit pad is er ook een rechtstreekse toegang via een LAN verbinding (Client-applicatie op een werkstation). Dit pad valt echter buiten het beschouwingsgebied.



Figuur 2. Voorbeeld van een in kaart gebrachte infrastructuur

3.3.2 Verwacht gebruikersgedrag

Het is van groot belang om in een vroeg stadium het te verwachten gebruikersgedrag en de daaraan gekoppelde efficiëntie-eisen te achterhalen. Dit gedrag kan in de vorm van aantallen per tijdseenheid worden weergegeven, zoals vereiste en gewenste responsetijden. Echter, er kunnen ook eisen zijn op het gebied van netwerkbelasting, CPU-belasting, etc.

Het resultaat van deze activiteit is een beschrijving van het verwachte gebruik van het informatiesysteem en het verwachte tijdsbeslag en middelenbeslag.

Voorbeeld verwacht gebruikersgedrag

- Er worden 4000 klanten in jaar 1, tot 20.000 klanten in jaar 5 verwacht;
- Een klant plaatst gemiddeld 12 orders per jaar;
- Op basis van 20.000 klanten wordt uitgegaan van 2000 tot 3000 aanmeldacties;
- Er worden 200 tot 600 orders per dag uitgevoerd;
- 2000 tot 3000 keer per dag vindt inzage in de portefeuille plaats;
- De gemiddelde sessieduur is 12 minuten.

Voorbeeld tijdsbeslag en middelenbeslag

Tabel 1. Voorbeeld verwacht tijdsbeslag

Scherm	Max. responsetijd
Informatie scherm	20 seconden
Portefeuille overzicht	15 seconden
Koersen hoofdfondsen	20 seconden
Koersen opties	20 seconden
Beurs orders	30 seconden
Koersen overige fondsen	35 seconden
Aanmeldscherm	5 seconden
Welkomstpagina	5 seconden
Orderscherm	5 seconden
Koers ophalen	5 seconden
Goedkeuren order	5 seconden
Order naar de beurs	5 seconden
Depotrekening/portefeuille	7 seconden
Uitstaande orders	5 seconden
Terugmelding fiattering	15 seconden
Terugmelding geplaatst	15 seconden
Plaatsing bevestigd	15 seconden

Tabel 2. Voorbeeld verwacht middelenbeslag

Onderdeel	Meetaspect	Acceptatiecriterium
Geheugengebruik	Memory pages / sec	0
	Memory usage	Gemiddeld < 80%, met slechts enkele uitschieters
Processor gebruik	processor:%processor time	Gemiddeld < 80% per CPU
	system:processor queue length	< 2 per CPU
Schijfgebruik	physicaldisk:% disk time	< 80%
	logicaldisk:% disk time	< 80%
	logical disk: avg. disk queue length	< 2
	logical disk: current disk queue length	< 2
	physical disk: avg. disk queue length	< 2
	physical disk: current diskqueue length	< 2

3.3.3 Ontwerpen gebruikersscenario's

In het verlengde van het verwachte gebruikersgedrag worden gebruikersscenario's ontworpen. Een gebruikersscenario is vergelijkbaar met een logisch testscript. In zo'n scenario worden de achtereenvolgende handelingen van een (test-)gebruiker vastgelegd. Het doel van een gebruikersscenario is het toekomstig gebruik van de applicatie te simuleren.

Het spreekt voor zich dat hierbij de inbreng van de business groot is: die afdeling zal het verwachte gebruik moeten aangeven.

Het resultaat van deze activiteit is een beschrijving van de gebruikersscenario's. De uit te voeren handelingen worden zo gedetailleerd beschreven, dat zij als basis kunnen dienen voor het maken van de testscenario's en de fysieke testscripts.

Voorbeeld gebruikersscenario's

Er worden drie gebruikersscenario's onderkend. In de onderstaande tabel wordt per scenario aangegeven uit welke handelingen deze bestaat en hoe de procentuele verdeling van de scenario's is. Tijdens de testuitvoering worden deze percentages gebruikt om de scenario's over het totaal aantal gelijktijdige testgebruikers te verdelen.

Tabel 3. Voorbeeld van gebruikersscenario's

Gebruikersscenario 1	Gebruikersscenario 2	Gebruikersscenario 3

• Aanmelden
• Opvragen saldo
• Opvragen rekeningoverzicht
• Afmelden
	• Aanmelden
	• Opvragen saldo
	• Opvoeren betalingen:
	  • Spaartransactie
	  • Betaaltransactie
	• Afmelden
		• Aanmelden
		• Aanvraag creditcard
		• Aanvraag afschriftenmap
		• Opvragen saldo
		• Afmelden

Verwachte verdeling: 60% van de gebruikers	Verwachte verdeling: 30% van de gebruikers	Verwachte verdeling: 10% van de gebruikers

3.3.4 Ontwerpen belastingmodel

In een belastingmodel wordt aangegeven op welke wijze de belasting op een systeem wordt opgevoerd. Centraal in het model staan de volgende vijf onderdelen (zie Figuur 3):

- het vereiste belastingniveau;
- performancetest;
- loadtest;
- piekbelasting;
- stresstest.

Vereiste belastingniveau

Dit niveau geeft het aantal gelijktijdige gebruikers aan waarvoor het systeem geschikt moet zijn.

Performancetest

Het doel van de performancetest is te achterhalen hoe het systeem reageert op toenemende belasting. Stel dat een vereiste is dat er 200 bezoekers tegelijkertijd met de applicatie actief moeten kunnen zijn. Tijdens de performancetest zal het aantal gesimuleerde gebruikers stapsgewijs opgevoerd worden, net zolang tot er 200 VU[1] gelijktijdig actief zijn. In Figuur 3 is als voorbeeld een reeks van: 40, 80, 120, 160, 200 VU weergegeven. Elke stap in deze test duurt bijvoorbeeld tussen de 10 en 30 minuten, afhankelijk van de tijdsduur van het doorlopen van de gebruikersscenario's. Een vuistregel is: de testduur is ongeveer 3 tot 5 maal de tijdsduur van het langstdurende gebruikersscenario.

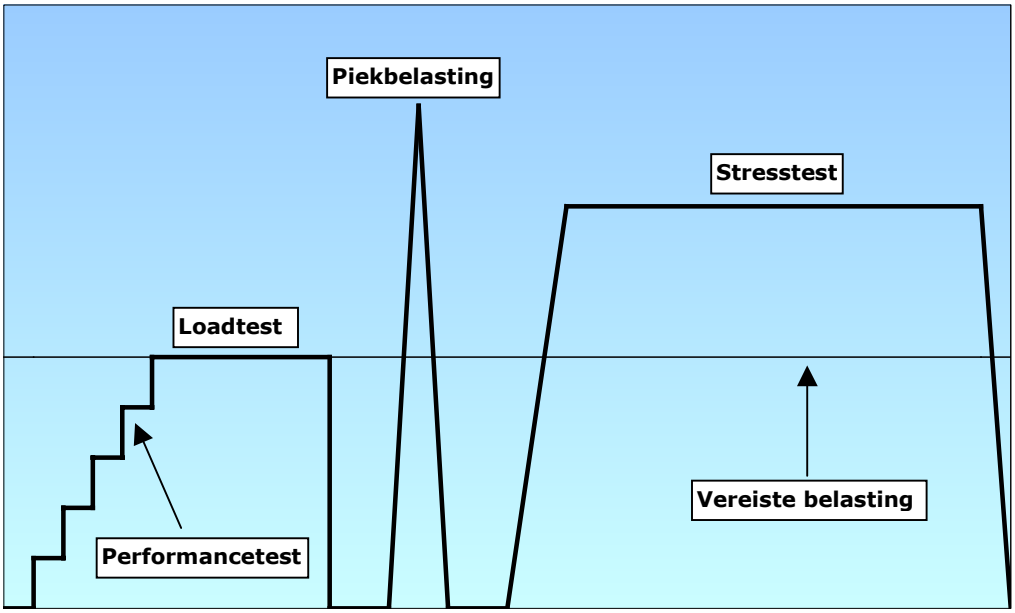
[1] VU = Virtual User = gesimuleerde gebruiker

Loadtest
Het doel van de loadtest is het achterhalen hoe het systeem zich gedraagt als gedurende een lange tijd (tot enkele uren) het vereiste aantal gebruikers gelijktijdig actief is.
Voorbeeld: gedurende 2 uur wordt een test met 200 VU uitgevoerd.

Piekbelasting
Bij een piekbelasting wordt de te testen applicatie kortdurend (extreem) zwaar belast, om eventuele breekpunten boven water te halen.
In het voorbeeld van Figuur 3 worden in korte tijd 400 VU op de applicatie "losgelaten" (vuistregel: 2 tot 2½ maal de vereiste belasting) om vast te stellen hoe de applicatie (en de infrastructuur) zich gedraagt. Let op: de meeste tijd wordt besteed aan de opstart- en de afbouwfase van de test. De tijd dat er daadwerkelijk een (zeer) hoog aantal gebruikers parallel actief is, is maar kort: 10-30 minuten.

Stresstest
Bij een stresstest wordt de applicatie langdurig zwaar belast, boven het vereiste maximum niveau, om te onderzoeken hoe het testobject zich onder die omstandigheden gedraagt.
In het voorbeeld van Figuur 3 wordt een niveau van 300 VU gehanteerd (vuistregel: 1½ maal de vereiste belasting) gedurende een tijdsperiode van 6 uur.

Het resultaat van de activiteit ontwerpen belastingmodel is:
- een invulling van het vereiste belastingniveau;
- een opsomming van de uit te voeren testvormen;
- een opsomming van de aantallen VU per testvorm.



Figuur 3. Voorbeeld van een belastingmodel

Voorbeeld belastingmodel
Voor het uitvoeren van de efficiëntietest voor applicatie X is het volgende belastingmodel ontworpen:
- het vereist belastingniveau: 100 gebruikers;
- een performancetest met de reeks 1, 5, 10, 25, 50, 75, 100 VU;
- een loadtest met 100 VU;
- een piekbelasting met 200 VU;
- een stresstest met 150 VU.

3.3.5 Ontwerp testinfrastructuur en meetpunten

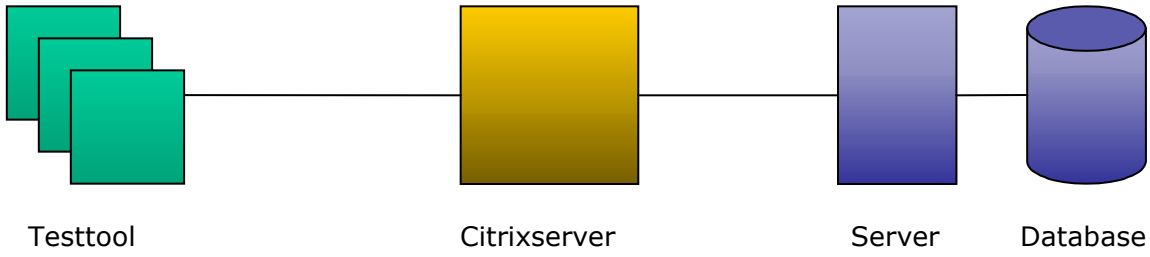
Zoals in paragraaf 3.3.1 is vermeld, wordt bij de efficiëntietest de gehele keten in beschouwing genomen. Deze keten wordt in een tekening (mét beschijving) vastgelegd.

Het is niet altijd mogelijk om de volledige infrastructuur in de test op te nemen, denk bijvoorbeeld aan web-applicaties: het is ondoenlijk (en ook niet wenselijk) het internet als onderdeel van de testinfrastructuur op te nemen.

Om een goed inzicht te krijgen in het gedrag van de verschillende componenten in de keten is het nodig om een aantal meetpunten te definiëren.
Per meetpunt wordt bepaald *wat* er gemeten gaat worden en *wie* de meting uitvoert/begeleidt. Denk bijvoorbeeld aan CPU-belasting, netwerkbelasting, geheugengebruik, disk i/o, verhouding send/resend van packets, etc.

Het resultaat van deze activiteit is een tekening van de testinfrastructuur en een tabel waarin de meetpunten beschreven staan.

Voorbeeld meetpunten



Figuur 4. Voorbeeld van een vereenvoudigde weergave van een testinfrastructuur

Tabel 4. Voorbeeld van een overzicht van de meetpunten en de meetaspecten

Apparaat	Applicaties	Wat te meten (meet aspecten)	Doel meetaspect	Methode / meet tools	Contact persoon
Client	ICA	Response tijden	Indicatie hoelang een transactie duurt	Tool A	Persoon A
		Aantal VU	Indicatie van het aantal gelijktijdige gebruikers	Tool A	
		Health (CPU & memory)	Indicatie van de belasting van de testmachines	Tool A	
Citrix server	Citrix + applicatie	Health (CPU & memory)	Indicatie van de belasting van de Citrix servers	Tool A Perfmon	
		Throughput	Indicatie van de netwerkbelasting	Perfmon Sniffermeting	Persoon C
		Disk	Indicatie van het schijfgebruik	Perfmon	
		Citrix Metaframe Performance Counters: • input counters • latency counters • output counters	Indicatie van de belasting van de Citrix verbinding. Detailinvulling later te bepalen	Tool A	
Server	AIX	Health (CPU & memory)	Indicatie van de belasting van de server	Vmstat, Nmon	Persoon D
	Oracle	Statspack	Performancemetrieken Oracle	Oracle	Persoon D

3.3.6 Definiëren testgegevens

Als de gebruikersscenario's zijn opgesteld, dan worden daarna de benodigde testgegevens gedefinieerd. Denk daarbij aan beveiligingsinformatie (klantnummer, wachtwoord, etc.), portefeuille-informatie, rekeninginformatie, saldo informatie, etc. Niet alleen testgegevens in het te testen informatiesysteem zelf zijn belangrijk: vergeet ook niet de gegevens die uit aangrenzende systemen worden opgehaald, zoals: koersinformatie, contractinformatie, etc.

Het lastige aan deze activiteit is, dat er gezorgd moet worden voor een consistente set van gegevens, die ook nog eens beheerd moet worden. Het volstaat in dezen niet om "zomaar" een kopie van de productiegegevens te nemen. Immers: het is vaak niet bekend of alle te testen situaties zich daadwerkelijk in de productie omgeving bevinden.

Het resultaat van deze activiteit is een beschrijving van de benodigde testgegevens, zoals hieronder in een voorbeeld wordt aangegeven.

Voorbeeld van een beschrijving van testdata.
Schatting van het aantal transacties
200 gebruikers gaan gedurende 1½ uur effectenorders invoeren (waarbij een orderinvoer ongeveer 3 minuten duurt). Dit betekent 6000 transacties[2]. Per gebruiker zijn dat 30 transacties. Verspreid over de hele testperiode wordt rekening gehouden met zo'n 300 effectentransacties per klant.

Fondsen
Voor de fondsen wordt gebruik gemaakt van de door een externe partij aangeleverde fondsgegevens. Er zal dus gebruik worden gemaakt van productiegegevens. In deze gegevens wordt niet gemanipuleerd ten behoeve van testdoeleinden.

Klanten
Er zullen minimaal 425 klanten beschikbaar moeten zijn, met ieder zijn eigen klantnummer en wachtwoord. Deze klantenkring is alleen voor de performancetest beschikbaar. Elke klant mag alle soorten orders uitvoeren, en heeft een geldsaldo van minimaal € 1.000.000,=. Dit moet meer dan voldoende zijn om de geschatte 300 ordertransacties per testklant te kunnen bekostigen.
De klantgegevens worden (handmatig) ingebracht in de backoffice en via de daarvoor bestemde (en al bestaande) weg gedistribueerd naar de servers.

Portefeuilles
Elke testklant dient over een portefeuille te beschikken, waarin een mix van fondsen en opties voorkomt. In de onderstaande tabellen wordt de samenstelling van de (test-)portefeuille aangegeven. De portefeuillegegevens worden (handmatig) ingebracht in de server omgeving.

Tabel 5. Voorbeeld van aandelenfondsen per testportefeuille per klant

Fondssymbool	Fondscode	Aantal	Historische koers
FOR	30086	1.000	€ 20,00
CMG	43037	1.000	€ 10,00
LAU	34075	1.000	€ 5,00

Tabel 6. Voorbeeld van opties per testportefeuille per klant

Optieklasse	Optieserie	Aantal	Verloopdatum
FOR	FOR C OCT 2004 20,00	1.000	Oktober 2004
CMG	CMG P APR 2004 3,00	1.000	April 2004

[2] 200 gebruikers * (90 minuten / 3 min per transactie) = 6000 transacties

3.3.7 Testontwerp

Alle beschrijvingen, tekeningen en tabellen die tot nu toe zijn gemaakt, worden opgenomen in een document: het testontwerp. Dit ontwerp beschrijft daarmee de inhoud van de uiteindelijk uit te voeren testen.
In het testontwerp worden de gebruikersscenario's, het belastingmodel, eventuele technische aspecten van het te gebruiken testtool en de uitvoeringsaspecten samen genomen in een hoofdstuk testscenario's. In dit hoofdstuk wordt aangegeven welke testen, met welk gebruikersscenario, met welk aantal gebruikers, gedurende welke tijd worden uitgevoerd.

Voorbeeld testscenario
Met het gebruikersscenario "Opvoeren Bedrijfsbeoordeling" wordt een drietal meetseries uitgevoerd. Elke meetserie bestaat uit een aantal meetsessies met een vast aantal gebruikers en een vaste tijdsduur.

Tabel 7. Voorbeeld uitvoeringsschema meetsessies

	Testscenario	**Aantal VU**				
Nr	Opvoeren Bedrijfsbeoordeling	**1**	**5**	**10**	**25**	**50**
1	Performancetest 2 CPU	15 min.	15 min.	15 min.	15 min.	15 min.
2	Performancetest 1 CPU, SMS Simulatie	15 min.	15 min.	15 min.	15 min.	15 min.
3	Performancetest 1 CPU, SMS Normaal, Servlet Caching aan	--	15 min.	15 min.	--	--

3.3.8 Inrichten testinfrastructuur, meetpunten en testgegevens

Zoals eerder is opgemerkt, is het de bedoeling dat tijdens de efficiëntietest de gehele keten in beschouwing wordt genomen, en dat de testomgeving zoveel mogelijk gelijk is aan de productie-omgeving. Dan pas hebben de te verzamelen gegevens enige zeggingskracht.
Echter, in de praktijk is de testomgeving slechts een afspiegeling van de (uiteindelijke) productie omgeving, om een aantal redenen:

- Het wordt als te duur gezien om twee (of meer) identieke omgevingen aan te schaffen, te onderhouden en te beheren;
- Op het productiesysteem draaien processen, die gezamenlijk voor een achtergrondbelasting zorgen. Deze achtergrondbelasting is op de testomgeving niet, of slechts met zeer grote moeite (lees: hoge kosten), te realiseren/simuleren.

Vanuit een praktisch oogpunt gezien betekent dit, dat er gestreefd moet worden naar een zo goed mogelijk testsysteem, met als minimale eis:

- een volledige keten;
- de verschillen tussen test en productie (CPU-capaciteit, geheugen, netwerksnelheid, etc) zijn bekend.

Het resultaat van deze activiteit is een werkende testomgeving, inclusief de meetpunten (zoals beschreven in paragraaf 3.3.5) en de testgegevens (zoals beschreven in paragraaf 3.3.6).

3.3.9 Emulatiesoftware

Soms wordt er voor een informatiesysteem een speciaal beveiligingshulpmiddel gebruikt, zoals:

- een smartcard voor Online Banking (Fortis Bank);
- de digipas bij Rabo Internet bankieren (Rabobank);
- de TAN-lijst van Girotel (Postbank).

Het is noodzakelijk om eerst te onderzoeken hoe de beveiliging werkt, en daarna hoe dit onderdeel te simuleren valt: zonder een goede simulatie van de beveiliging is er geen efficiëntietest mogelijk.

Een TAN-lijst is makkelijk te automatiseren, een Smartcard wordt al iets moeilijker, omdat er dan interne Smartcardsleutels in een bestand geplaatst moeten gaan worden.

Een andere mogelijkheid is natuurlijk om, voor de efficiëntietest, de beveiliging te omzeilen. Toch is dit niet aan te bevelen, immers: in een productieomgeving wordt de beveiliging wel degelijk uitgevoerd, en speelt deze dus ook een rol in het gedrag van het systeem.

Kortom: er zal nagedacht moeten worden over hoe het beveiligingsmechanisme te simuleren valt, voor meer dan 1 persoon tegelijkertijd. In vele gevallen betekent dat, dat er speciale simulatieprogrammatuur ontwikkeld moet worden.

Het resultaat van deze activiteit is een simulatieprogramma (of een combinatie van hardware en speciale software) dat de gebruikershandelingen op beveiligingsgebied effectief en efficiënt simuleert.

3.3.10 Fysieke testscripts

Last but not least: al deze voorbereidingen dienen uiteindelijk om op enigerlei wijze fysieke testscripts te maken, waarmee de performance van de te testen applicatie getest kan worden.
Het zou, in het kader van dit document, te ver voeren om het mechanisme van opnemen van testscripts volledig te beschrijven, daarom wordt volstaan met een globale uitleg.

Het opnemen van testscripts is gebaseerd op *Record & Playback*. De tester doorloopt met een van de gebruikersscenario's het informatiesysteem, en laat ondertussen het testhulpmiddel de handelingen vastleggen. Het resultaat is een leesbaar script, waarin zonodig speciale aanpassingen/aanvullingen aangebracht worden (denk aan logmeldingen of afdrukken van tijdsintervallen van bepaalde handelingen). Het testhulpmiddel voert vervolgens het script uit voor 1, 2, 3, veel… gebruikers, en doet tegelijkertijd metingen naar de responsetijden, de netwerkbelasting, etc.

3.3.11 Draaiboek

Nadat de individuele testscripts zijn gemaakt (en getest!!) kan het draaiboek worden samengesteld. In het draaiboek wordt beschreven, welke tests in welke volgorde uitgevoerd gaan worden, met welke duur en met welk aantal gebruikers. Eventuele andere handelingen worden ook in het draaiboek opgenomen (bijv. backup/restore van testgegevens).

Een draaiboek is niet altijd een apart document: de informatie kan ook al beschreven zijn in het testontwerp, onderdeel testscenario's (zie paragraaf 3.3.7).
Wanneer de uitvoering van de efficiëntietest complex is, of er zijn veel partijen bij betrokken, dan is het wel zinvol een apart draaiboek op te stellen.

3.4 TESTUITVOERING

3.4.1 Inleiding

De fase *testuitvoering* valt uiteen in een drietal activiteiten:
- Uitvoeren pretest;
- Uitvoeren testruns;
- Analyse testresultaten.

3.4.2 Uitvoeren pretest

Voordat de testruns worden uitgevoerd, wordt aanbevolen om, als een vorm van intake, eerst met één testgebruiker een pretest uit te voeren. Met een van de testscenario's (/testscripts) zal een kortdurende test (van enkele minuten) worden uitgevoerd, om te controleren of de testomgeving ook daadwerkelijk "up & running" is. Mocht uit deze test blijken dat onderdelen van de infrastructuur niet

beschikbaar zijn, of er worden fouten in de applicatie gevonden, dan zullen deze zaken eerst hersteld moeten worden, voordat de test verder kan gaan.

3.4.3 Uitvoeren testruns

In het testontwerp is, bij het onderdeel testscenario's, het aantal meetseries en meetsessies bepaald. Deze meetseries en meetsessies worden op de aangegeven volgorde uitgevoerd. Tegelijkertijd worden metingen verricht op de aangegeven meetpunten naar bijvoorbeeld de CPU-belasting, de diskactiviteit, het netwerkgebruik, de responsetijden, etc.

Resultaat van deze activiteit is een aantal uitgevoerde testruns, en een grote hoeveelheid te analyseren meetgegevens.

Voorbeeld testruns
Er wordt een drietal meetseries uitgevoerd, met elk een aantal meetsessies. Tijdens de eerste meetserie wordt het testscenario "Opvoeren Bedrijfsbeoordeling" toegepast met de reeks 1, 5, 10, 25, 50 VU. Elke meetsessie duurt 15 minuten. Tijdens deze meetserie wordt gebruik gemaakt van een machine met 2 CPU's.

Bij de tweede meetserie wordt gebruik gemaakt van hetzelfde testscenario en dezelfde reeks met gebruikersaantallen, maar nu wordt een machine met slechts 1 CPU gebruikt. Tevens wordt de toegang tot SMS gesimuleerd.

Bij de derde test wordt de normale toegang tot SMS gebruikt, maar wordt Servlet Caching aangezet. Er worden twee meetsessies gehouden met respectievelijk 5 en 10 VU, elk met een testduur van 15 minuten.

3.4.4 Analyse testresultaten

Tijdens de meetsessies worden veel (!) meetgegevens gegenereerd, zowel door het geautomatiseerde hulpmiddel, als op de meetpunten in het netwerk. Deze gegevens worden verzameld, geanalyseerd en bewaard.

Het is raadzaam om de resultaten van de analyses tussentijds bekend te maken. De ervaring heeft namelijk geleerd dat het ondoenlijk is om alle analyseresultaten pas in een eindrapport te publiceren: het rapport zou dan veel te dik worden.
Bij het tussentijds bekend maken van de resultaten is ook het voordeel dat eventuele knelpunten op tijd gesignaleerd worden, en mogelijk ook snel verholpen kunnen worden.

Nota bene:
Er wordt benadrukt dat er een aantal herhalingen plaats zal vinden:
- er wordt een serie van testruns uitgevoerd;
- de meetgegevens worden geanalyseerd;
- de resultaten worden in rapportvorm vastgelegd en gepubliceerd;
- de volgende serie testruns wordt uitgevoerd;
- de gegevens worden geanalyseerd;
- de resultaten worden gepubliceerd;
- etc.

Voorbeeld analyse testresultaten
Een aantal analyses is eenvoudig (en vlot) te maken. In Figuur 5 wordt het verloop van de responsetijden van het onderdeel Aanlog in de reeks 8, 16, 24, 36, 48 VU getoond. In de grafiek worden verschillende gegevens weergegeven:

- de gemiddelde responsetijd;
- de maximale en minimale responsetijd;
- de 90-percentiel score van de responsetijd;
- het aantal geslaagde en niet-geslaagde aanmeldpogingen.

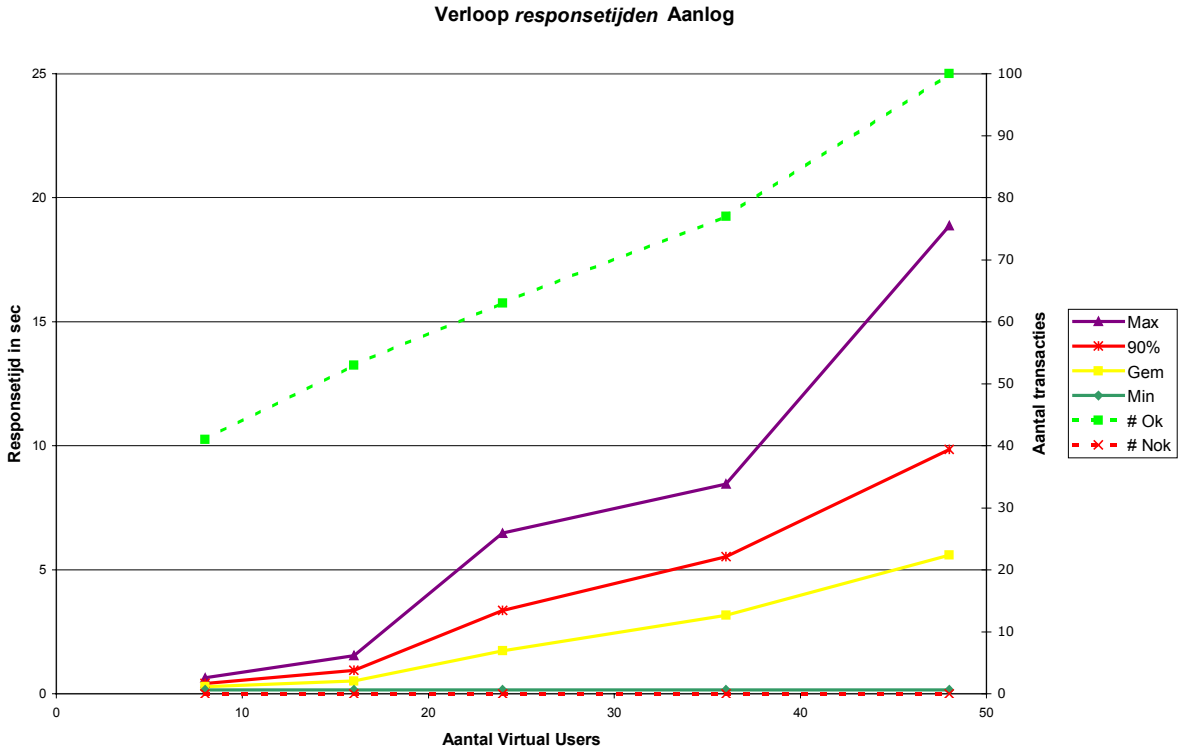
In de grafiek zijn, op het gebied van de responsetijden, de lijnen van de gemiddelde responsetijd en de 90-percentiel score de meest belangrijke onderdelen. De andere twee (max. en min.) zijn minder van belang, omdat die lijnen maar door 5 punten per lijn worden bepaald (een maximum- of minimum-waarde komt slechts éénmaal voor per meetsessie).

Uit de grafiek is af te lezen dat er een zwak exponentieel verband bestaat tussen de responsetijden en het aantal VU; er is zeker geen lineair verband.
Stel dat een acceptatiecriterium is dat deze actie niet meer dan 5 sec. mag duren, dan kan uit deze grafiek af gelezen worden, dat, rekening houdend met de 90-percentiellijn, de applicatie op dit onderdeel ruim 30 bezoekers gelijktijdig kan verwerken.

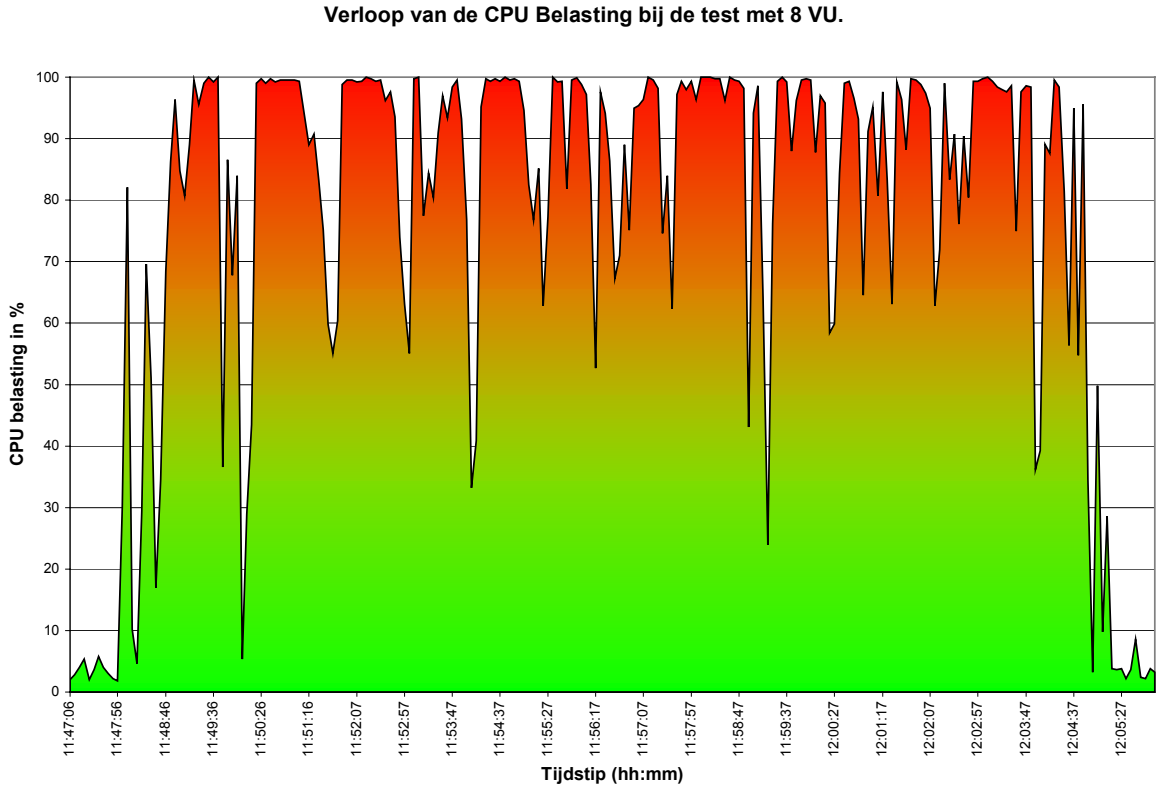
In Figuur 6 is het verloop van de CPU-belasting van de server weergegeven, bij de test met 8 VU. In dit voorbeeld is de test gestart op tijdstip 11:48 en gestopt op 12:05. Uit de grafiek is af te lezen dat de CPU tijdens de test vaak het verzadigingspunt (100%) nadert, en een aantal keer "gewoon" verzadigd is.
De CPU-belasting van de server vóór en ná de test ligt rond de 5%. Kennelijk legt de applicatie een groot beslag op de CPU-capaciteit van de server, gezien het lage aantal gelijktijdige gebruikers: 8 VU.

Uit de getoonde gegevens is de oorzaak van de hoge belasting niet af te leiden. Daarvoor zal verder onderzoek moeten worden uitgevoerd. Mogelijke oorzaken kunnen zijn:

- inefficiënte programmaopzet;
- onvoldoende intern geheugen (waardoor veel "swap"-activiteit zal plaatsvinden);
- complexe datamanipulatie;
- er wordt een groot aantal Java-objecten aangemaakt, waardoor de Java Garbage Collector veelvuldig actief moet worden omdat anders de Java Runtime omgeving volloopt.



Figuur 5. Voorbeeld van het verloop van de responsetijden en het aantal goed of fout verlopen transacties bij het onderdeel aanlog



Figuur 6. Voorbeeld van het verloop van de CPU-belasting bij de test met 8 VU; meetinterval 5 seconden

3.5 TESTAFRONDING

3.5.1 Eindrapport

Na afloop wordt er een rapport gemaakt waarin de meetresultaten afgezet worden tegen de doelstellingen van de test. Het eindrapport zal in de regel weinig meetresultaten bevatten, tenzij er geen tussenrapporten zijn verschenen.

4 TOT SLOT

In dit document is een aanpak beschreven voor het voorbereiden en uitvoeren van een test op het gebied van efficiëntie. Het te gebruiken testhulpmiddel is bewust niet beschreven: er is zo'n grote verscheidenheid aan hulpmiddelen op dit gebied op de markt verkrijgbaar, dat het ondoenlijk is om dat in dit document op te nemen. Belangrijker is echter, dat de aanpak net zo belangrijk, misschien wel belangrijker is dan het testhulpmiddel.

Monitoring

De geschetste aanpak is niet 1-op-1 toepasbaar in een monitoringtraject, waarbij de productie omgeving in de gaten wordt gehouden. Het verschil zit hem voornamelijk in de aantallen. Bij efficiëntietesten worden veel gelijktijdige gebruikers gesimuleerd, bij monitoring wordt gebruik gemaakt van slechts enkele gebruikers, die als "tracers" fungeren. De "echte" gebruikers zorgen immers voor de achtergrondbelasting. Door het volgen van enkele gesimuleerde gebruikers krijgt men een inzage in de response die de werkelijke klanten ook ervaren.
In veel gevallen zullen de productie servers al regulier in de gaten gehouden worden voor wat betreft de hoogte van de CPU-belasting, de hoogte van het netwerkverkeer, etc.

I BEGRIPPEN EN AFKORTINGEN

In deze bijlage wordt een aantal begrippen en afkortingen toegelicht.

Bandbreedte
Een maat voor de maximale snelheid waarmee (een onderdeel van) het netwerk gegevens kan transporteren. Basiseenheid is bit per seconde.

Batch proces
Een proces dat apart (door een gebruiker) opgestart wordt, en vervolgens autonoom verder werkt, totdat het betreffende proces afgerond is.

CPU
Afkorting van: central processing unit, de centrale verwerkingseenheid van de computer, ook wel de processor genoemd..

Interactief proces
Een proces waarbij de gebruikershandelingen centraal staan, bijvoorbeeld het invullen van schermen.

Kwaliteitsattribuut
Een beschrijving van een eigenschap van een informatiesysteem.

Meetserie
De uitvoering van één of meer meetsessies voor een (groep van) gebruikersscenario('s).

Meetsessie
Een testuitvoering van een (groep van) gebruikersscenario('s) voor één bepaald aantal testgebruikers (VU), gedurende een bepaalde tijd.

Middelenbeslag
De eigenschap Middelenbeslag beschrijft de mate waarin het (software-)systeem een beroep doet op beschikbare operationele middelen

Resource
Middelen waarover de software kan beschikken voor het uitvoeren van taken, zoals: processor, intern geheugen, disk capaciteit, netwerkcapaciteit, etc

Responsetijd
De tijd die een informatiesysteem nodig heeft om op een gebruikershandeling te reageren.

Tijdbeslag
De eigenschap Tijdbeslag beschrijft de mate waarin het (software-)systeem tijd nodig heeft om te reageren op invoer of om transacties te verwerken (en de eventuele beïnvloeding door grote volumes).

TMap
Samentrekking van: Test Management Approach. TMap is een standaard voor testen geworden.

VU
Afkorting van Virtual User, ofwel: gelijktijdige, gesimuleerde gebruiker.

II LITERATUURLIJST

Auteur	Jaar	Titel
Anonymus	2002	Whitepaper TPerf, gestructureerd performancetesten met TMap® Versie 3.0, 10 jan 2002; 9 pp. IQUIP Informatica BV
M. Dekkers & B. van der Schaaf	2001	Leidraad voor testen e-business Het testen van internettoepassingen volgens KWTS 1e druk, Eburon Delft, 179 pp. ISBN 90 5166 838 4
F. Hoeben & M. Sterk	1998	Performance modellering White paper, versie 1.0, 16-10-1998, 15 pp. Stichting SERC, Utrecht
ISO/IEC	2000	ISO/IEC 9126 Software Product Quality; part 1: Quality model Final draft version
ISO/IEC	2000	ISO/IEC 9126 Software Product Quality; part 2: External metrics Final draft version
ISO/IEC	2000	ISO/IEC 9126 Software Product Quality; part 3: Internal metrics Final draft version
M. Pol, R. Teunissen & E. van Veenendaal	1999	Testen volgens Tmap 2e druk, Uitgeverij Tutein Nolthenius, 's-Hertogenbosch, 633 pp. ISBN 90 7219 458 6

III OVER DE AUTEUR

Roland Mees is ruim 3½ jaar werkzaam bij Fortis Bank als senior testmanager. Daarvoor is hij werkzaam geweest als tester en testcoördinator bij de Postbank. In de afgelopen 2½ jaar heeft hij zich gespecialiseerd op het gebied van efficiëntietesten, en daarin veel ervaring opgedaan bij interne projecten.
Zijn deelname aan de werkgroep Testtechnieken van de vereniging TestNet is de aanzet geweest voor dit white paper.