De COSMIC Functionele Omvang Meetmethode

Versie 3.0

Methode Overzicht

Nederlandse Vertaling

November 2008

MET DANK AAN

'COSMIC-FFP' METHOD[1] VERSIE 2.0 AUTEURS (alfabetische volgorde)

Alain Abran, École de technologie supérieure – Université du Québec,
Jean-Marc Desharnais, Software Engineering Laboratory in Applied Metrics - SELAM,
Serge Oligny, Bell Canada,
Denis St-Pierre, DSA Consulting,
Charles Symons, Software Measurement Services, UK

Versie 2.0 reviewers 1998/1999 (alfabetische volgorde)		
Moritsugu Araki, JECS Systems Research, Japan	Thomas Fetcke, Germany	Patrice Nolin, Hydro Québec, Canada
Fred Bootsma, Nortel, Canada	Eric Foltin, University of Magdeburg, Germany	Marie O'Neill, Software Management Methods, Ireland
Denis Bourdeau, Bell Canada, Canada	Anna Franco, CRSSM, Canada	Jolijn Onvlee, The Netherlands *
Pierre Bourque, , ÉCole de Technologie supérieure, Canada	Paul Goodman, Software Measurement Services, United Kingdom	Laura Primera, UQAM, Canada
Gunter Guerhen, Bürhen & Partner, Germany	Nihal Kececi, University of Maryland, United States	Paul Radford, Charismatek, Australia
Sylvain Clermont, Hydro Québec, Canada	Robyn Lawrie, Australia	Eberhard Rudolph, Germany
David Déry, CGI, Canada	Ghislain Lévesque, UQAM, Canada	Grant Rule, Software Measurement Services, United Kingdom*
Gilles Desoblins, France	Roberto Meli, Data Processing Organization, Italy	Richard Stutzke, Science Applications Int'l Corporation, United States
Martin D'Souza, Total Metrics, Australia	Pam Morris, Total Metrics, Australia*	Ilionar Sylva, UQAM, Canada
Reiner Dumke, University of Magdeburg, Germany	Risto Nevalainen, Software Technology Transfer Finland, Finland *	Vinh T. Ho, UQAM, Vietnam
Peter Fagg, United Kingdom	Jin Ng, Hmaster, Australia	

* Eerste leden van het COSMIC kernteam, samen met de COSMIC-FFP Methode v2.0 auteurs.

[1] versie 2.0 was de eerste publiekelijk verkrijgbare versie van de COSMIC-FFP methode

Versie 3.0 reviewers 2006/07 (alfabetische volgorde)		
Alain Abran, École de Technologie Supérieure, Université du Québec, Canada	Jean-Marc Desharnais, Software Engineering Lab in Applied Metrics – SELAM, Canada	Arlan Lesterhuis*, Sogeti, The Netherlands
Bernard Londeix, Telmaco, United Kingdom	Roberto Meli, Data Processing Organization, Italy	Pam Morris, Total Metrics, Australia
Serge Oligny, Bell Canada	Marie O'Neill, Software Management Methods, Ireland	Tony Rollo, Software Measurement Services, United Kingdom
Grant Rule, Software Measurement Services, United Kingdom	Luca Santillo, Agile Metrics, Italy	Charles Symons*, United Kingdom
Hannu Toivonen, Nokia Siemens Networks, Finland	Frank Vogelezang, Ordina, The Netherlands	

* Samenstellers van versie 3.0 van de COSMIC methode

VERSIE BEHEER

De volgende tabel vat de wijzigingen ten aanzien van dit ‘Methode overzicht’ document samen.

DATUM	REVIEWER(S)	Wijzigingen / Toevoegingen
September 2007	COSMIC Measurement Practices Committee	Eerste versie voor publieke uitgave. De inhoud is gedeeltelijk gebaseerd op hoofdstuk 2 van het meethandboek v2.2, maar is significant uitgebreid.
November 2008	Nesma werkgroep COSMIC	Nederlandse vertaling
December 2015	Nesma werkgroep COSMIC	Update van logo's en website verwijzingen

Nederlandse vertaling 2008	
Harold van Heeringen, Metri	Hans van den Brink, Centric
Frank Vogelezang, Ordina	Jolijn Onvlee, Onvlee Opleidingen & Advies

VOORWOORD

De COSMIC methode is een gestandaardiseerde methode om de functionele omvang van software te meten binnen de domeinen die meestal worden aangeduid als 'administratieve software' (of management informatie systemen) en real-time software, of hybriden van deze.

De COSMIC methode is geaccepteerd door ISO/IEC JTC1 SC7 in december 2002 als internationale standaard ISO/IEC 19761 'Software Engineering – COSMIC-FFP – A functional size measurement method', (hierna aangeduid als 'ISO/IEC 19761').

Doel van dit document

Het doel van dit 'Methode overzicht' is het verschaffen van een samenvatting van de COSMIC functionele omvangmeetmethode, versie 3.0. Dit document is interessant voor lezers die:

- een overzicht van de methode nodig hebben, maar niet alle details hoeven te weten;
- nieuw zijn met het concept van het meten van de functionele omvang van software en die enige introductie nodig hebben op dit gebied;
- bekend zijn met een bestaande '1^{e} generatie' functionele omvangmeetmethode (zoals 'de IFPUG', 'MarkII' of 'Nesma' methodes) en die overwegen om verder te gaan met COSMIC.

COSMIC Methode documentatie

Voor een volledig overzicht van alle documentatie met betrekking tot de methode wordt verwezen naar het document 'COSMIC documentatie overzicht en begrippenlijst'. De begrippenlijst in dat document bevat de definitie van alle termen die worden gebruikt binnen de COSMIC methode. Dit document en alle documenten die hieronder worden genoemd kunnen gratis worden gedownload van www.cosmic-sizing.org.

Voor meer achtergrond gegevens worden de lezers van dit document verwezen naar de volgende documenten:

- De ISO/IEC 19761 standaard, dat de fundamentele normatieve definities en richtlijnen van de methode bevat.
- Het 'COSMIC Meethandboek, versie 3.0', dat de telrichtlijnen en definities geeft en verder ook tot doel heeft om verdere uitleg en meerdere voorbeelden te geven, om de analisten te helpen de methode volledig te begrijpen en toe te passen.
- De 'COSMIC Method version 3.0, Advanced & Related Topics' dat materiaal bevat dat verder gaat dan de basis methode. Behandeld worden o.a. methoden van snel of vroegtijdig indicatief meten en de converteerbaarheid van functionele omvang gemeten met andere functionele omvangmeetmethodes tot een omvang in COSMIC functiepunten. (nog niet in het Nederlands beschikbaar).

Andere beschikbare COSMIC documenten zijn de 'Guidelines' die de toepassing van de methode in bepaalde software domeinen beschrijft, diverse case studies, wetenschappelijke artikelen, benchmark data, etc. Vertalingen van het meethandboek in andere talen zijn tevens verkrijgbaar. Al deze documenten zijn te vinden op www.cosmic-sizing.org.

Meer algemene achtergrondinformatie over functionele omvangmeetmethodes, de voordelen van de COSMIC methode, de COSMIC organisatie en haar activiteiten, leveranciers van COSMIC gerelateerde diensten, COSMIC nieuwsbrieven, etc kan men vinden op www.cosmic-sizing.org.

Het COSMIC Measurement Practices Committee (MPC)

INHOUDSOPGAVE

1

INTRODUCTIE

Softwareontwikkeling is een belangrijk onderdeel van vele bedrijfsbudgetten. Organisaties onderkennen het belang om de softwarekosten te beheersen. Zij doen dat door de bedragen die gekoppeld zijn aan softwareontwikkeling en onderhoud te analyseren en de resultaten te vergelijken met de beste in het veld (benchmarking). Vandaar dat meten nodig is voor zowel de kwaliteit als de productiviteit die gekoppeld zijn aan het ontwikkelen en onderhouden van software.

Ontwikkelaars hebben behoefte aan technische meetresultaten om de technische performance van een product of dienst te kunnen kwantificeren. Technische metingen kunnen bijvoorbeeld gebruikt worden voor “efficiency” analyses of om een ontwerp te verbeteren.

Hiernaast zijn functionele metingen noodzakelijk, bijvoorbeeld het schatten of het meten van de omvang van de software in een vroeg stadium van het project aan de hand van requirements als voornaamste variabele om projectinspanning te meten, of een productiviteitsanalyse om de performance van producten of diensten vast te stellen gezien vanuit het perspectief van een gebruiker of eigenaar.

Functionele metingen moeten onafhankelijk zijn van technische ontwikkeling en van implementatie besluiten. Ze kunnen dan gebruikt worden om de productiviteit, die verkregen is vanuit verschillende technieken en technologieën, met elkaar te vergelijken.

Functiepunt analyse (FPA)[2] is een voorbeeld van een functionele omvangsmeetmethode. Het is bruikbaar voor software in het MIS domein waarin het op uitgebreide schaal is gebruikt voor productiviteitsanalyse en het begroten van projecten (Abran, 1996; Desharnais, 1988; Jones, 1996; Kemerer, 1987). Deze methode kan de specifieke functionele kenmerken van MIS software aan op een succesvolle manier meten..

FPA is echter wel bekritiseerd omdat het niet universeel toepasbaar is voor alle typen software [Conte, 1986; Galea, 1995; Grady, 1992; Hetzel, 1993; Ince, 1991; Jones, 1988; Jones, 1991; Kan, 1993; Whitmire, 1992]. Met name de real-time software gemeenschap heeft FPA nooit goed geaccepteerd.

In de “Full Function Point” methode (versie 1.0) van 1997 werd als doel gesteld om FPA uit te breiden zodat het de functionele omvang van real-time, technische en systeem software kan omvatten. Veldtesten toonden aan dat FFP ook geschikt om de functionele omvang van MIS software te meten, dat wil zeggen: de metingen leiden tot dezelfde resultaten.

In 1998 fuseerde de FFP groep met de COSMIC groep die de principes van de tweede generatie functionele omvangsmeetmethode voorstelde. Deze inspanning resulteerde in de eerste, voor het publiek toegankelijke “field trials” versie 2.0 van de COSMIC-FFP meetmethode, gepubliceerd in oktober 1999.

Vanaf versie 2.0 is de COSMIC meetmethode ook ontworpen om volledig in overeenstemming te zijn met zowel ISO/IEC 14143-1: 1998 standaard (en vervolgens ISO/IEC 14143-1: 2007) als met de COSMIC principes.

---

[2] Albrecht A.J., Gaffney Jr. J.E., “Software function, source lines of code and development effort prediction: a software science validation”, IEEE Transactions on Software Engineering, Vol. SE-9, pp. 639-648, November 1983. ‘FPA’ s staat nu bekend als de‘IFPUG’ methode.

Over het COSMIC initiatief

Gezien de explosieve groei en de diversiteit aan software (outsourcing) contracten en hebben klanten en leveranciers behoefte aan meer accurate schattingsmethoden en meetprestaties. Deze methoden moeten voor alle typen software even betrouwbaar werken. “Eerste generatie” methoden voor omvangsmeting van software zijn niet altijd voldoende krachtig om aan de behoeften van de markt te voldoen of zijn slechts geschikt voor het meten van een beperkt aantal typen software.. De industrie heeft dringend behoefte aan software meetmethoden die aantoonbaar meer accuraat en op ruimere schaal bruikbaar zijn.

De COSMIC groep richt zich op deze behoeften voor:

- software leveranciers, die de requirements van de opdrachtgever vertalen naar de omvang van de software die gemaakt moet worden als een sleutelonderdeel van het schatten van de projectkosten,

- klanten die de functionele omvang van de geleverde software willen weten als een belangrijk onderdeel van het meten van de leverancierspresta tie.

COSMIC, het COmmon Software Measurement International Consortium, is een vrijwillig initiatief van een internationale groep software meetexperts, zowel mensen uit de praktijk als academici, van Azië, Oceanië, Europa en Noord-Amerika. Het oorspronkelijke doel van het COSMIC project was het ontwikkelen, testen, op de markt brengen en acceptatie zoeken van nieuwe software meetmethode ter ondersteuning van het begroten van softwareprojecten en het meten van prestaties van software ontwikkelingsprojecten.

Nadat de principes van de COSMIC methode voor het eerst in 1999 waren gepubliceerd, zijn er in 2000/01 door verschillende internationale bedrijven en academische instituten met succes “field trials” uitgevoerd. Papers die deze trial resultaten beschrijven en andere onderzoeksresultaten zijn op de site www.cosmic-sizing.org geplaatst. In 2001 startte het proces van het ontwikkelen van een Internationale Standaard voor de COSMIC methode. De Standaard werd in december 2002 goedgekeurd en gepubliceerd door ISO in begin 2003 als ISO/IEC 19761.

COSMIC gaat door met het verfijnen van de definities. Versie 3.0 van het meethandboek is de laatste stap in dit verfijningproces, dat doorgaat terwijl het altijd compatibel blijft met de ISO / IEC 19761 standaard. **Het moet worden benadrukt dat het Generieke Software Model, die de basis vormt van de omvangsmeting, sinds het voor de eerste keer is gepubliceerd, niet is veranderd.** Het ontwerp van versie 3.0 in vergelijking tot de voorgaande versie 2.2 toont aan dat v 3.0 een belangrijke stap voorwaarts is in de verfijning van de methode. Voor een volledige verantwoording van de wijzigingen die zijn aangebracht vanaf v2.2 en v3.0 wordt verwezen naar “Het COSMIC meethandboek v3.0”

Het Common Software Measurement International Consortium (COSMIC) ziet er op toe dat deze toevoegingen en verfijningen worden voorgelegd aan ISO voor het opnemen in ISO / IEC 19761 als het wordt gereviseerd in 2007/08

In 2006 introduceerde COSMIC het eerste “Entry-level” examencertificaat voor beoefenaars van de methode. Gebruikers van de COSMIC methode worden sterk aangemoedigd om de performance gegevens van hun projecten te sturen naar de International Software Benchmarking Standards Group (‘ISBSG’), om de bestaande benchmark uit te breiden met meetgegevens afkomstig van de COSMIC methode.

Voor meer informatie over COSMIC, publicaties, activiteiten en examens, kunt u terecht op de volgende site: www.cosmic-sizing.org. Voor de Nesma vertaling: www.nesma.org. Voor meer informatie over ISBSG zie: www.isbsg.org.

2

OVERZICHT VAN DE COSMIC MEETMETHODE

De COSMIC meetmethode definieert een gestandaardiseerde methode om de functionele omvang van software te meten. In dit hoofdstuk worden de volgende onderwerpen besproken:

- de types software waarvoor de COSMIC methode is ontworpen om de functionele omvang van te meten (bekend als het 'toepassingsdomein' van de methode) in paragraaf 2.1.

- een overzicht van de software modellen die worden gebruikt om te meten in paragraaf 2.2. Deze modellen introduceren alle basisconcepten van de COSMIC methode. Het begrijpen van deze concepten is belangrijk, omdat de analist bij het meten van de functionele omvang van een stuk software de software artefacten (bijvoorbeeld een basisontwerp of de fysieke implementatie van software) moet vertalen naar de concepten van de COSMIC modellen.

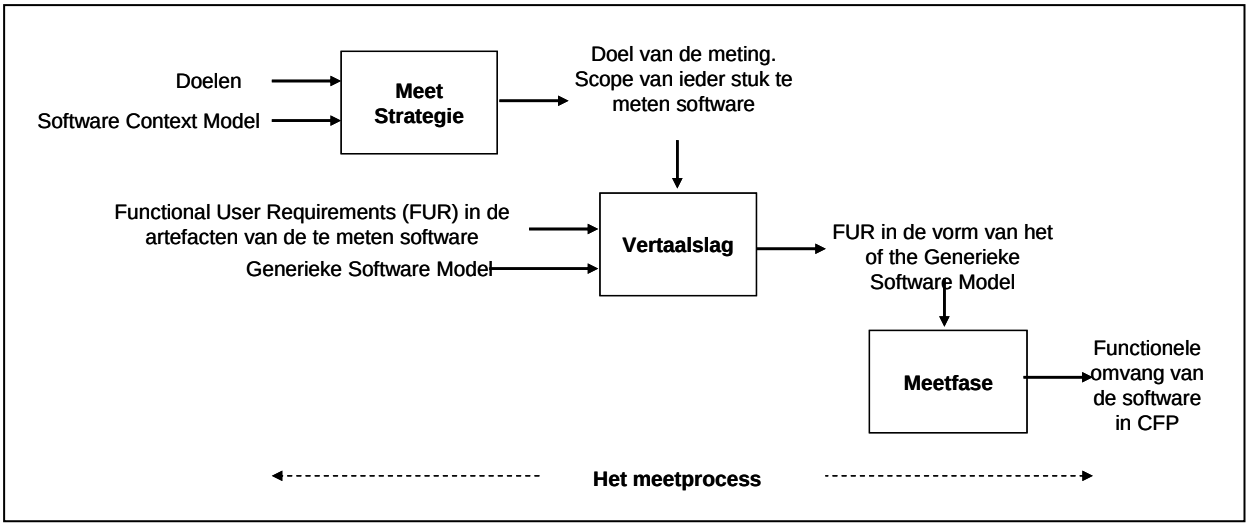
- een overzicht van het algemene COSMIC meetproces, dat uit drie fasen bestaat:

    - De Meetstrategie, vastgesteld voordat met de meting wordt begonnen (paragraaf 2.3.1).

    - De Vertaalslag (paragraaf 2.3.2).

    - De Meetfase (paragraaf 2.3.3).

Het resultaat van het meetproces is een omvang uitgedrukt in 'COSMIC Functie Punten' ('CFP'). Deze fasen vind je terug in figuur 2.0.



Figuur 2.0 De structuur van de COSMIC methode

De paragrafen 2.3.1, 2.3.2 en 2.3.3 worden verder uitgewerkt in een hoofdstuk van het Meethandboek. Daar worden de complete en gedetailleerde definities, uitgangspunten en regels van de methode gegeven samen met een aantal voorbeelden.

2.1 De toepasbaarheid van de COSMIC methode

2.1.1 Toepasbaar op diverse domeinen

De COSMIC meetmethode is ontworpen om toepasbaar te zijn op de functionaliteit van software in de volgende domeinen:

- Administratieve informatiesystemen ('Business Application Software') ter ondersteuning van de bedrijfsadministratie, zoals bij banken, verzekeringsbedrijven, accountants, personeelssystemen, inkoop, distributie of productie. Deze software wordt vaak gekarakteriseerd als 'rijk aan data'. Haar complexiteit wordt met name bepaald door de behoefte om grote hoeveelheden data te beheersen over gebeurtenissen in de echte wereld.
- Real-time software, die als taak heeft om gebeurtenissen in de echte wereld bij te houden of te sturen. Voorbeelden hiervan zijn: software voor telefoonswitches, software die is ingebouwd in apparaten, zoals huishoudelijke apparaten, liften, automotoren, en vliegtuigen, met als doel het proces te sturen en het automatisch verzamelen van data en binnen het operating system van computers.
- Hybriden van de bovenstaande types software zoals een real-time reserveringssysteem van een vliegmaatschappij of van een hotel.

2.1.2 Niet-toepasbaar

De COSMIC meetmethode is (nog) niet ontworpen om de functionele omvang te meten van rekenintensieve software, dus software die wordt gekarakteriseerd door complexe rekenkundige algoritmen en andere gespecialiseerde en complexe regels, zoals in expertsystemen, simulatie software, zelflerende software, weersvoorspelling systemen, etc., of systemen die continue variabelen verwerken zoals audio geluiden of video beelden zoals je die tegen komt binnen computer games, muziekinstrumenten en dergelijke. Ook meet de COSMIC methode geen aspecten van de functionaliteit zoals 'complexiteit'.

Voor dit soort software is het echter mogelijk om lokale uitbreidingen van de COSMIC meetmethode te definiëren. In het Meethandboek wordt uitgelegd binnen welke context deze lokale uitbreidingen worden gebruikt

2.2 De COSMIC software modellen

Deze paragraaf geeft een overzicht van de COSMIC methode en alle basisconcepten daarvan. De definities van al deze concepten staan in het document 'COSMIC Documentatie Overzicht en Begrippenlijst'. In deze paragraaf wordt ieder van deze concepten die voor de eerste keer wordt genoemd **vet** weergegeven.

Het is essentieel dat analisten bij het uitvoeren van een meting ALLE COSMIC modellen en basisconcepten begrijpen die hieronder worden gepresenteerd en dat zij ALLE uitgangspunten en regels toepassen die in de hoofdstukken van het meethandboek beschreven zijn. Alleen met deze discipline kan de analist er zeker van zijn dat metingen betekenisvol zijn en deze metingen herhaald kunnen worden door andere analisten en/of vergelijkbaar zijn met metingen die door andere analisten in andere software omgevingen zijn gemaakt.

2.2.1 Functional User Requirements

De COSMIC meetmethode bestaat uit het toepassen van een verzameling modellen, uitgangspunten en processen op de **Functional User Requirements** (of **FUR**) van een gegeven stuk software. Het resultaat is een numerieke waarde, die de **functionele omvang** van de software weergeeft conform de COSMIC methode in eenheden 'COSMIC Functiepunten' (of 'CFP').

De functionele omvang die door de COSMIC meetmethode verkregen wordt, is ontworpen om onafhankelijk te zijn van de manier waarop de software is geïmplementeerd in de operationele artefacten van de te meten software. 'Functionaliteit' houdt zich bezig met de 'informatieverwerking die de software moet uitvoeren voor haar **gebruikers**.

Meer specifiek, de FUR beschrijven 'wat' de software moet doen voor de **functionele gebruikers**. Dit zijn 'de verzenders en beoogde ontvangers van gegevens van en naar de gewenste functionaliteit'. De FUR negeren alle technische of kwaliteitseisen die aangeven 'hoe' de software zich moet gedragen. Alleen de FUR worden in beschouwing genomen bij het meten van een functionele omvang.

Het afleiden van de FUR uit software artefacten in de praktijk

In de dagelijkse praktijk van de software ontwikkeling is het erg zeldzaam om artefacten te vinden waarin de FUR zeer duidelijk kunnen worden onderscheiden ten opzichte van andere types requirements en waarin de FUR zodanig zijn beschreven dat ze direct meetbaar zijn zonder de noodzaak om te moeten interpreteren. Dit betekent dat de analist normaal gesproken de aangeleverde of geïmpliceerde FUR moet afleiden uit de werkelijke artefacten en deze moet vertalen naar de concepten van de COSMIC software modellen.

FUR kunnen worden afgeleid uit de software engineering artefacten die worden gemaakt voordat de software is gerealiseerd, zoals requirements definitiedocumenten, de resultaten van gegevens- of functionele analyse van de requirements, etc. Dit betekent dat de functionele omvang van software kan worden gemeten voordat de software wordt geïmplementeerd op een computersysteem.

In andere gevallen kan het nodig zijn een bestaand stuk software te meten zonder dat er enige, of slechts zeer weinig, architectuur- of ontwerp artefacten voorhanden zijn en de FUR misschien niet gedocumenteerd zijn (bijvoorbeeld bij 'legacy' software). In dit soort gevallen is het nog steeds mogelijk om de FUR af te leiden van de artefacten die op de computer zijn geïnstalleerd na de implementatie, zoals fysieke schermen of rapporten of door de gegevensstromen te analyseren.

Het afleiden van de FUR uit software artefacten

Het proces om de FUR af te leiden uit verschillende types software engineering artefacten en deze weer te geven in de vorm van de COSMIC software modellen zal uiteraard variëren afhankelijk van het type artefact. Dergelijke processen zijn domeinspecifiek en variëren dusdanig dat dit niet binnen de COSMIC methode kan worden behandeld. De methode gaat er vanuit dat de FUR van de te meten software ofwel bestaan ofwel kunnen worden afgeleid uit zijn artefacten. Niettemin publiceert COSMIC ook domeinspecifieke 'Handleidingen' die een aantal aspecten van het afleiden van de FUR beschrijven.[3]

De COSMIC methode beperkt zich daarom tot het beschrijven en het definiëren van de concepten van de COSMIC software modellen. Deze concepten zijn gevat in twee COSMIC software modellen: het 'Software Context Model' en het 'Generieke Software Model'.

[3] De 'Guideline for Sizing Business Application Software using COSMIC' helpt bij het vertalen van verschillende data analyse en methoden om eisen vast te stellen die worden gebruikt in het domein van de administratieve software.

2.2.2 Het COSMIC Software Context Model

Een stuk software dat moet worden gemeten, moet zorgvuldig worden gedefinieerd (binnen de meetscope) en deze definitie moet rekening houden met de omgeving van andere software en/of hardware waarmee het communiceert. Dit software context model beschrijft de uitgangspunten en concepten die nodig zijn voor deze definitie.

UITGANGSPUNTEN – Het COSMIC Software Context Model
a) Software wordt begrensd door hardware
b) software is normaal gesproken gestructureerd in lagen
c) Een laag kan één of meer afzonderlijke 'peer' stukken software bevatten en verder kan elk stuk software bestaan uit afzonderlijke 'peer componenten'
d) Elk stuk software dat moet worden gemeten wordt gedefinieerd door zijn meetscope, die zich geheel binnen een enkele laag moet bevinden.
e) De scope van een stuk software dat moet worden gemeten hangt af van het doel van de meting
f) De functionele gebruikers van een stuk software worden afgeleid uit de FUR van het stuk software dat moet worden gemeten als de verzenders en/of beoogde ontvangers van gegevens
g) Een stuk software interacteert met zijn functionele gebruikers door middel van data movements over een grens en het stuk software kan gegevens van en naar persistente opslag binnen de grens verplaatsen
h) De FUR van software kan worden uitgedrukt in verschillende granulariteit
i) De granulariteit waarop metingen normaal gesproken moeten worden uitgevoerd is die van de functionele processen
j) Als het niet mogelijk is om te meten op de granulariteit van de functionele processen, dan moet de FUR gemeten worden met een benaderingsmethode en geschaald worden naar de granulariteit van de functionele processen[4].

Deze uitgangspunten en concepten worden nu uitgewerkt en geïllustreerd met een aantal eenvoudige voorbeelden.

Om dit te kunnen doen moeten we onderscheid maken tussen twee manieren om een computer hardware/software systeem te beschouwen als de context van een stuk software dat moet worden gemeten, namelijk:

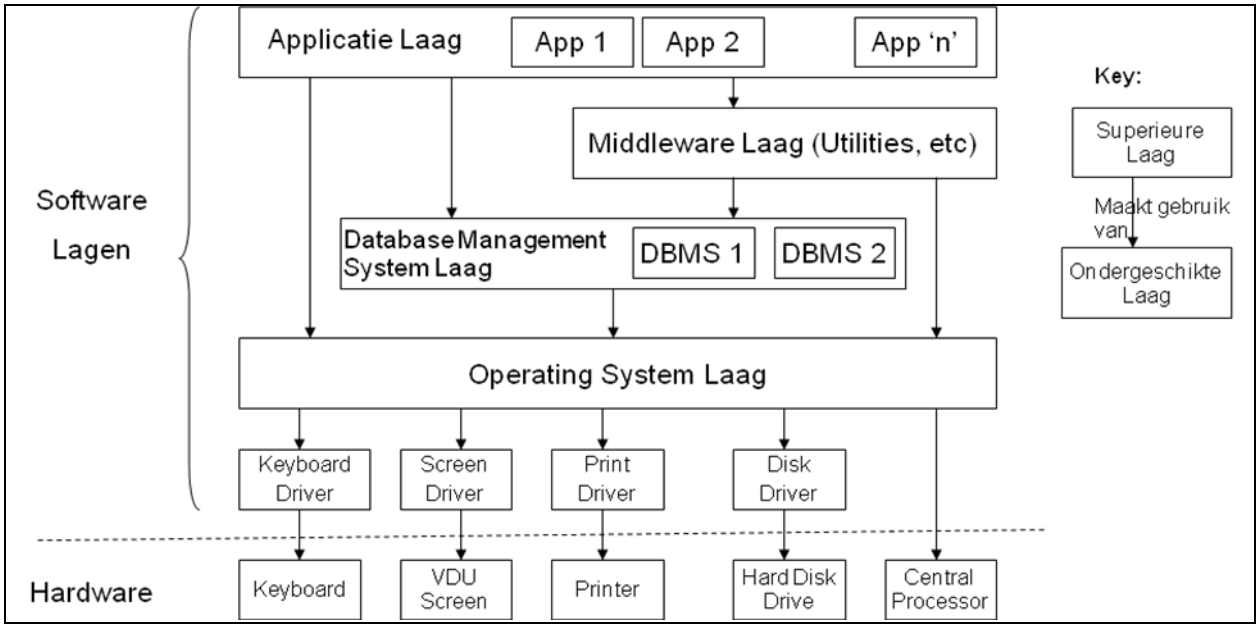
- De 'fysieke' beschouwing, die laat zien hoe in de praktijk de software normaal gesproken gestructureerd is in een hiërarchie van lagen, met elk zijn eigen specifieke functie. Deze beschouwing laat zien dat in werkelijkheid alle communicatie met een stuk software plaats vindt via hardware en (wellicht) via andere tussenliggende software lagen.

- De 'logische' beschouwing, die een abstractie is van de fysieke beschouwing voor het maken van een functionele omvangmeting. Deze beschouwing laat zien dat de functionele gebruikers van een stuk software dat moet worden gemeten (de verzenders en/of beoogde ontvangers van gegevens) over een grens met de software interacteren en dat de software gegevens verplaatst van en naar persistente opslag. In deze abstractie worden alle hardware en software die deze interactie mogelijk maken genegeerd.

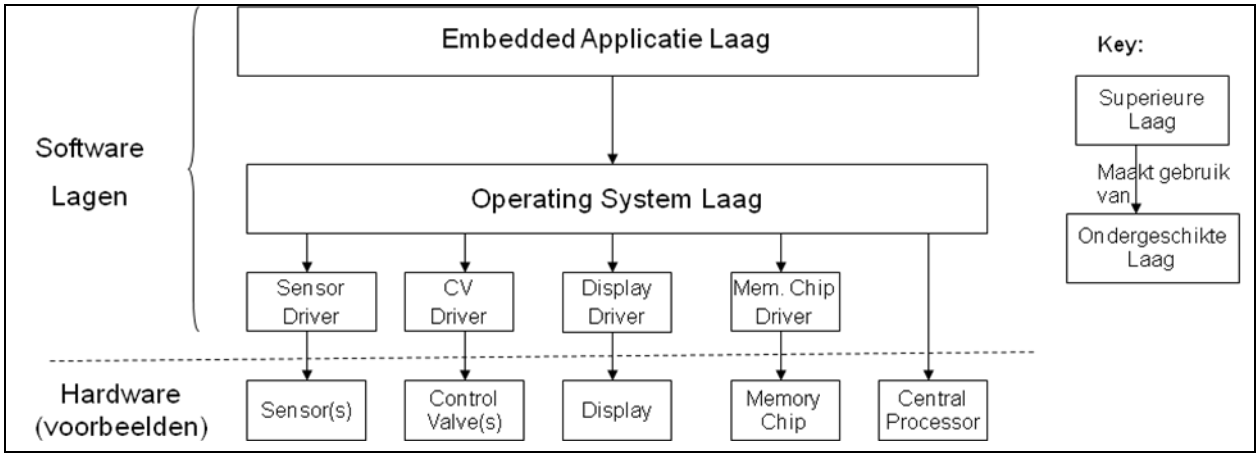
[4] Het onderwerp van het schalen tussen verschillende granulariteit wordt behandeld in het COSMIC document 'Advanced & Related Topics'.

Hoewel alleen de tweede beschouwing nodig is voor het maken van een functionele omvangmeting, is het nuttig om beide beschouwingen toe te lichten, omdat analisten onderscheid moeten kunnen maken tussen de twee beschouwingen. Verder gebruikt de COSMIC methode termen zoals 'laag' en 'peer component' op zeer specifieke wijze die goed begrepen moet worden. (Binnen de software industrie worden deze termen met zeer verschillende betekenis gebruikt.)

Figuur 2.2.2.1 illustreert de fysieke beschouwing van een doorsnee stuk bedrijfsinformatiesysteem in zijn context van een gelaagde software architectuur die bestaat uit het operating system, device drivers etc. en de hardware. Figuur 2.2.2.2 illustreert dezelfde fysieke beschouwing voor een eenvoudig voorbeeld van real-time embedded software.



Figuur 2.2.2.1 Typische gelaagde software architectuur voor een administratief/MIS systeem



Figuur 2.2.2.2 Typische gelaagde architectuur voor een real-time embedded software systeem

Uitgangspunt (a).

Software die gebruikt wordt door een menselijke gebruiker wordt begrensd door I/O hardware, zoals een muis, een toetsenbord, een printer of een beeldscherm; real-time embedded software wordt normaal gesproken begrensd door apparaten, zoals sensoren of relais. Software wordt ook begrensd door 'persistente opslag' hardware zoals een harde schijf of andere media die gebruikt kunnen worden om gegevens vast te houden.

Uitgangspunt (b)

Als het stuk software onderdeel is van een ontworpen gelaagde architectuur, zou het eenvoudig moeten zijn om vast te stellen tot welke laag dit stuk behoort. Wanneer een software omgeving echter is gegroeid en geëvolueerd door de tijd heen, kan het zijn dat de lagen (als ze er al zijn) niet duidelijk bepaald kunnen worden. Voor dit soort gevallen bevat de COSMIC methode een aantal regels om lagen te onderscheiden.

Uitgangspunt (c)

De verschillende hoofdcomponenten van bijvoorbeeld een informatiesysteem (bijvoorbeeld een 'drie lagen' architectuur van een 'front end / user interface' component, een 'bedrijfsregels' component en een 'data services' component) zijn peer componenten. Deze peer componenten op het hoogste niveau kunnen apart worden gemeten en de COSMIC methode geeft regels om op deze manier te meten. Deze mogelijkheid om de hoofdcomponenten van een applicatie apart te meten is erg belangrijk bij het meten van de inspanning en het begroten, met name als deze componenten zich op verschillende technische platforms bevinden.

Ieder stuk software in iedere laag kan worden opgedeeld in componenten op verschillende niveaus (bijvoorbeeld tot op het niveau van individuele modules of object classes) en de COSMIC methode kan worden gebruikt om de functionele omvang te meten op ieder van deze niveaus. Om verschillende metingen te kunnen vergelijken, zullen analisten echter (samen met software- of systeemarchitecten) lokale standaard niveaus van decompositie moeten bepalen onder het niveau van de hoofdcomponenten in iedere laag als zij er zeker van willen zijn dat metingen uit verschillende bronnen vergelijkbaar zijn.

Uitgangspunt (d)

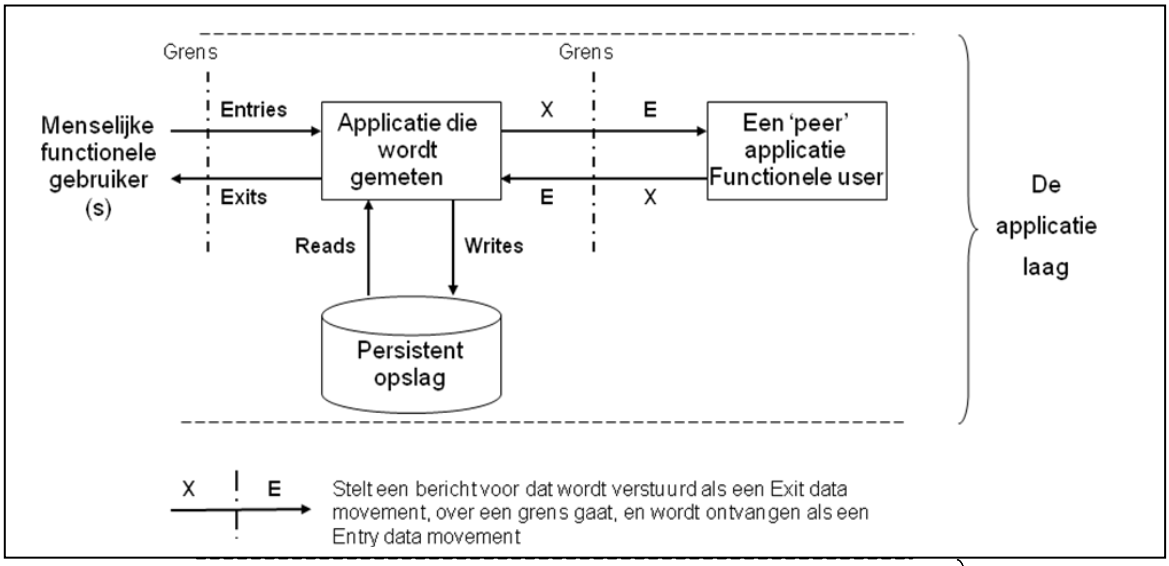
Uitgangspunt (d) schrijft voor dat de scope van een stuk software dat moet worden gemeten zich geheel binnen één enkele laag bevindt. De reden hiervoor is dat iedere laag een specifieke functie heeft en kan worden ontwikkeld met behulp van andere technologie van andere lagen. Het is dus wel of niet zinnig om de omvang van stukken software die zich in twee of meer lagen bevinden te meten en deze resultaten samen te nemen alsof het de omvang van een enkele entiteit weergeeft. Deze omvang kan, net als de som van de omvang van appels en peren, lastig te interpreteren en/of te vergelijken zijn met andere functionele omvangmetingen. Voor regels over het samennemen van de omvang van stukken software in verschillende lagen wordt verwezen naar de paragraaf over het samennemen van meetresultaten in het Meethandboek.

Uitgangspunt (e)

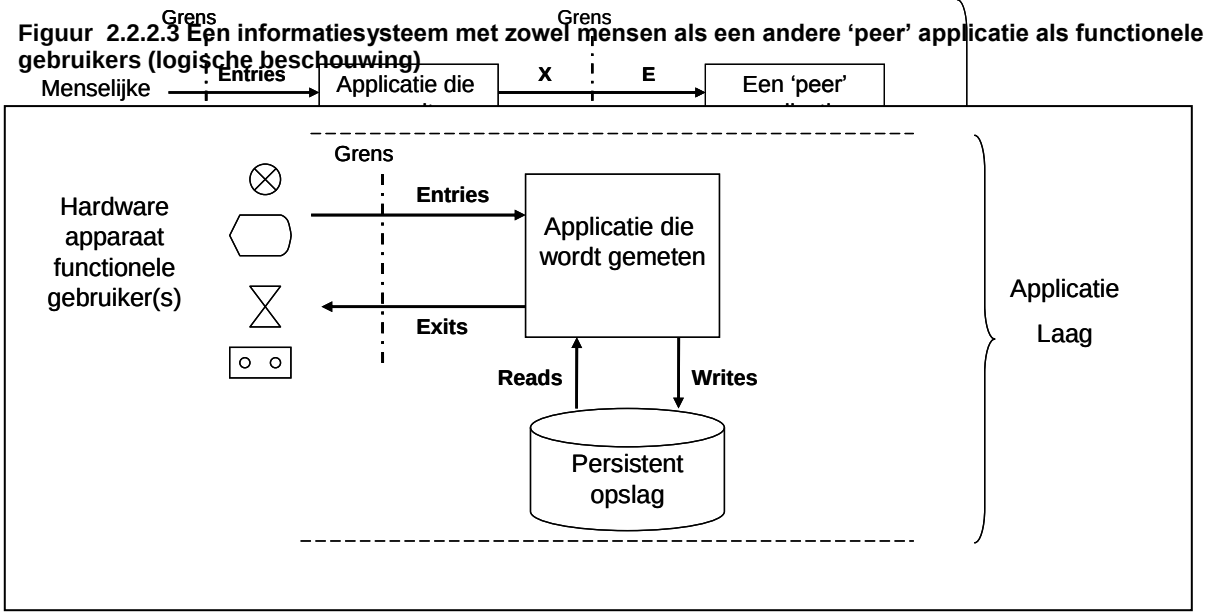
Als we bijvoorbeeld aannemen dat de stukken software in de figuren 2.2.2.1 en 2.2.2.2 elk afzonderlijke stukken software zijn die ieder door een eigen projectteam zijn ontwikkeld dan ligt het voor de meest gangbare meetdoeleinden voor de hand om de hele applicatie als meetscope te definiëren. Als de applicatie echter ontwikkeld is als drie peer componenten (zoals genoemd in relatie tot uitgangspunt (c) hierboven) die ieder gebruik maken van verschillende technologie en/of door verschillende teams worden ontwikkeld, ligt het meer voor de hand om drie afzonderlijke meetscopes te definiëren, één voor iedere peer component. De omvangmetingen van elk van de drie afzonderlijke componenten kan dan worden gebruikt als basis voor een begrotingsformule die rekening houdt met verschillende technologieën en/of karakteristieken van het projectteam voor iedere component.

Uitgangspunt (f)

Om dit uitgangspunt en het volgende uitgangspunt (g) toe te kunnen lichten is het noodzakelijk om eens nader te kijken naar de logische beschouwing van de software die gemeten moet worden. De figuren 2.2.2.3 en 2.2.2.4 illustreren deze logische beschouwing en tonen respectievelijk de interactie van de functionele gebruikers met een stuk bedrijfsinformatiesysteem en met een stuk real-time embedded software.



Figuur 2.2.2.3 Een informatiesysteem met zowel mensen als een andere 'peer' applicatie als functionele gebruikers (logische beschouwing)



Figuur 2.2.2.4 Real-time embedded software met verschillende hardware apparaten als functionele gebruikers (logische beschouwing)

Laten we eerst kijken naar het voorbeeld van een stuk bedrijfsinformatiesysteem dat moet worden gemeten. Figuur 2.2.2.1 laat zien dat zowel het operating system, de hardware (zoals het toetsenbord, de printer, etc.) als de menselijke gebruikers tot de 'gebruikers' kunnen worden gerekend, omdat ze allemaal, direct of indirect, interacteren met de applicatie. Maar niet al deze (typen) gebruikers zullen in de FUR gespecificeerd worden als verzenders en beoogde ontvangers van gegevens van en naar de applicatie. Het operating system en de hardware maken deze gegevensuitwisseling mogelijk en zijn zelf geen verzenders of beoogde ontvangers.

Voor een bedrijfsinformatiesysteem zal de FUR normaal gesproken alleen de gewenste functionaliteit beschrijven vanuit het gezichtspunt van de menselijke gebruikers van de applicatie en wellicht vanuit het gezichtspunt van andere 'peer applicaties' die gegevens verzenden naar of ontvangen van de applicatie. Deze mensen en 'peer applicaties' zullen daarom de functionele gebruikers zijn van de applicatie, zoals aangegeven in de logische beschouwing in figuur 2.2.2.3.

Vanwege de strikte scheiding van functionaliteit in lagen zoals in figuur 2.2.2.1 kan de FUR van een informatiesysteem normaal gesproken iedere softwarelaag of hardware zoals het operating system of het beeldscherm negeren die de interactie van de functionele gebruikers met de applicatie mogelijk maken. Normaal gesproken zal er geen twijfel zijn over wie de functionele gebruikers zijn.

In het voorbeeld van de real-time embedded software zal de FUR normaal gesproken de gewenste functionaliteit beschrijven vanuit het gezichtspunt van de hardware (sensoren, kleppen, etc.) die de software moet ondersteunen. Deze apparaten zullen daarom de functionele gebruikers zijn van de embedded software, zoals in figuur 2.2.2.4 wordt getoond. In het meethandboek zal worden ingegaan op het, zelden voorkomende, geval dat de FUR van sommige typen software meer dan één type functionele gebruiker beschrijft, waardoor er verschillende functionaliteit zichtbaar wordt en er dus een andere functionele omvang wordt gemeten. [5]

Uitgangspunt (g)

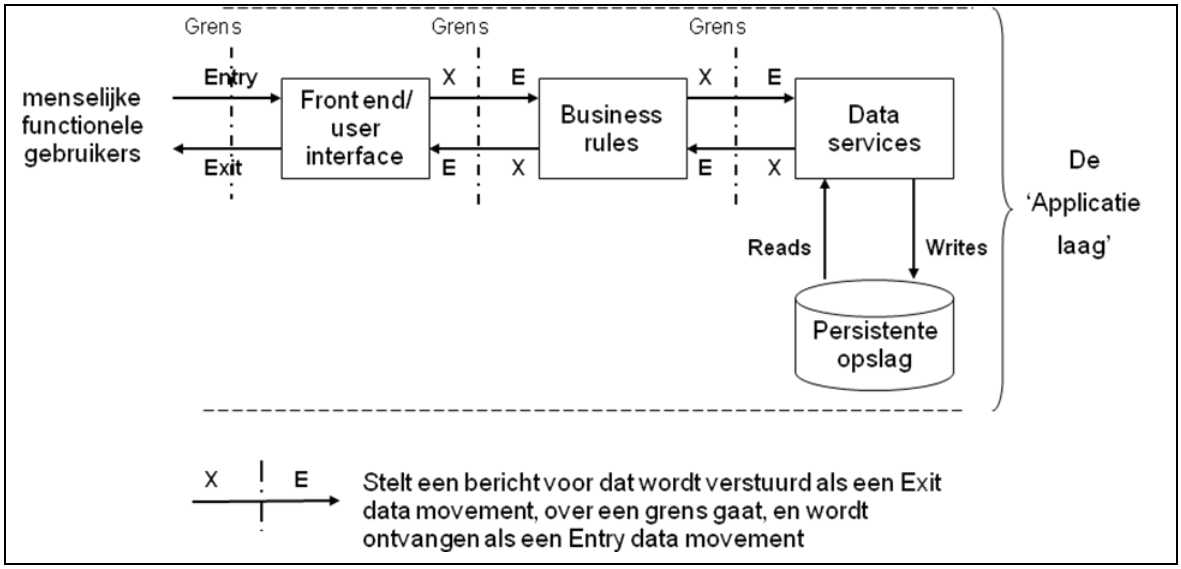
De figuren 2.2.2.3 en 2.2.2.4 laten ook zien dat functionele gebruikers interacteren met de software over een grens door middel van twee typen data movements (**Entries** en **Exits**). De software wisselt ook gegevens uit met persistente opslag hardware door middel van twee type data movements (**Reads** en **Writes**). Deze data movements worden verder gedefinieerd in paragraaf 2.2.3.

De 'grens' wordt gedefinieerd als 'een conceptuele interface tussen de te bestuderen software en zijn functionele gebruikers'. Deze grens moet niet worden verward met een lijn die om een diagram getrokken kan worden om de scope van een stuk software dat moet worden gemeten aan te geven, of om een software laag.

De grens maakt een duidelijk onderscheid mogelijk tussen alles dat onderdeel is van het stuk software dat moet worden gemeten (*alles dat zich aan de software kant van de grens bevindt*) en alles dat onderdeel is van de omgeving van de functionele gebruikers (*alles dat zich aan de functionele gebruikerskant van de grens bevindt*). Persistente opslag wordt niet gezien als een gebruiker van de software en bevindt zich daarom aan de software kant van de grens.

Figuur 2.2.2.5 laat nu de logische beschouwing zien van een bedrijfsinformatiesysteem dat ontwikkeld is als drie peer componenten zoals beschreven in de uitgangspunten (c) en (e) hierboven. Aangenomen dat de peer componenten ontwikkeld zijn met behulp van verschillende technologieën, dan ligt het voor de hand dat het doel van de meting voorschrijft dat er een afzonderlijke meetscope moet worden gedefinieerd voor iedere peer component.

[5] Een uitzondering waarin het operating system een functionele gebruiker van een applicatie kan zijn is als het operating system volgens de FUR moet voorzien in bijvoorbeeld een 'klok tik' om een functioneel proces in de applicatie te starten.



Figuur 2.2.2.5 – Een informatiesysteem waarvan de 'peer' componenten afzonderlijke gemeten moeten worden (logische beschouwing)

In de logische beschouwing van figuur 2.2.2.5 zien we dat de 'front end / user interface' component mensen en de 'bedrijfsregels' component als functionele gebruikers heeft, die elk met de component interacteren over een grens heen door middel van Entries en Exits. Deze figuur laat ook zien dat alleen de 'data services' component interacteert met de persistente opslag en dat de functionele gebruiker de 'business rules' component is. Met deze logische beschouwingen kan de FUR van iedere component afzonderlijk worden gemeten.

Uitgangspunten (h) en (i)

De FUR van de software kan op verschillende granulariteit beschreven worden. (Bedenk dat het concept van granulariteit te maken heeft met de mate van detail waarmee een stuk software beschreven is. Dit moet worden onderscheiden van het 'niveau van decompositie' die te maken heeft met het opdelen van de software in componenten.) De granulariteit waarop normaal gesproken metingen worden uitgevoerd is dat van de functionele processen (zie paragraaf 2.2.3). Wanneer de ontwikkeling van nieuwe software begint, start het proces om de FUR te bepalen normaal gesproken met het definiëren van en overeenstemming krijgen over eisen en wensen op een 'hoog niveau'. Deze worden vervolgens verfijnd en in meer detail uitgewerkt. De FUR van een stuk software met een gegeven scope kan daardoor met verschillende granulariteit bestaan. Een veel voorkomend voorbeeld van het gebruik van een functionele analyse techniek om de FUR van een stuk software te bepalen zou kunnen resulteren in de volgende hiërarchie van granulariteit van de FUR.

Een 'niveau 1 hoofdfunctie' blijkt bij een meer gedetailleerde analyse te bestaan uit een aantal 'niveau 2 functies'. Elk daarvan bestaat uit 'niveau 3 subfuncties', die weer 'niveau 4 sub-sub-functies' bevatten etc. Op enig punt in deze hiërarchie zal de analyse de individuele functionele processen tonen. (Paragraaf 2.2.3 gaat hier verder op in. Voor nu is het alleen van belang om deze te zien als standaard stukken functionaliteit die we kunnen meten.)

Als de analyse verder 'inzoomt' op meer en meer detail, zal de gemeten functionaliteit waarschijnlijk toe lijken te nemen omdat er meer details in beschouwing worden genomen. (Bedenk dat dit fenomeen iets anders is als 'scope creep' waarbij de omvang toeneemt omdat de scope van de software toeneemt).

Om metingen uit verschillende bronnen zinvol te kunnen vergelijken of de metingen in een ander proces te kunnen gebruiken, moeten alle metingen op een standaard granulariteit worden gemaakt (of daarnaar geschaald worden) dat we de 'functioneel proces granulariteit' noemen. In de meeste gevallen waarin het doel is om de functionaliteit van een volledig gespecificeerd of van een bestaand stuk software te meten zal de functioneel proces granulariteit zich zelf wijzen.

Iedereen is bekend met het idee om afstanden te meten op kaarten die verschillende schalen hebben, bijvoorbeeld waarop 1 km wordt weergegeven door 1 cm of 1 mm op de kaart en deze van de ene naar de andere schaal te vertalen. Maar wanneer we de functionele omvang van een stuk software meten met behulp van COSMIC, hebben we maar één standaard 'functioneel proces granulariteit' en maar één meeteenheid. Als we dus metingen op verschillende granulariteit moeten vergelijken, moeten we daarbij uitvinden wat onze lokale schaalfactoren zijn om deze omvang te vertalen naar de omvang op het standaard 'functioneel proces granulariteit'. Deze concepten worden verder uitgediept in de paragraaf over de standaard granulariteit in het meethandboek.

Uitgangspunt (j)

Het probleem dat een stuk software moet worden gemeten op een granulariteit die hoger is dan van de functionele processen doet zich normaal gesproken alleen voor in het voorstadium van de ontwikkeling van nieuwe software als de eisen en wensen zich nog ontwikkelen. In die omstandigheden waar het niet mogelijk is om op de granulariteit van de functionele processen te meten moet de FUR worden gemeten met een benaderingsaanpak en geschaald worden naar de granulariteit van de functionele processen. (zie het hoofdstuk over benaderingsmethoden in het document 'COSMIC method v3.0: Advanced & Related topics).

Samenvattend geeft het Software Context Model van de COSMIC meetmethode een verzameling concepten en uitgangspunten, te weten software lagen en peer componenten, de scope van een stuk software dat moet worden gemeten, zijn functionele gebruikers, data movements en een grens die helpt bij het meten van de FUR, die op verschillende granulariteit beschreven kan zijn. In het meetstrategie proces (beschreven in paragraaf 2.2.4) passen we deze concepten en uitgangspunten toe op de FUR van de software die gemeten moet worden om vragen als 'wat voor meting is er nodig?' of 'hoe moet deze meting worden geïnterpreteerd?' te beantwoorden.

2.2.3 Het COSMIC Generieke Software Model

Nadat de FUR van de te meten software geïnterpreteerd is naar de begrippen van het Software Context Model, kan het Generieke Software Model op de FUR worden toegepast om vast te stellen welke componenten van de functionaliteit gemeten zullen worden. Dit Generieke Software Model gaat er van uit dat de volgende algemene uitgangspunten waar zijn voor alle software die met de methode gemeten kan worden. Zie de begrippenlijst voor de definitie van de gebruikte begrippen. [6]

UITGANGSPUNTEN – Het COSMIC Generieke Software Model
a) Software ontvangt input gegevens van zijn functionele gebruikers en produceert output en/of een andere uitkomst voor de functionele gebruikers.
b) FUR van een stuk te meten software kunnen worden vertaald naar unieke functionele processen.
c) Ieder functioneel proces bestaat uit subprocessen.
d) Een subproces kan ofwel een data movement ofwel een gegevensverwerking zijn.
e) Ieder functioneel proces wordt getriggerd door een Entry data movement vanuit een functionele gebruiker die het functionele proces informeert dat de functionele gebruiker een event waargenomen heeft.
f) Een data movement verplaatst een enkele gegevensgroep.
g) Een gegevensgroep bestaat uit een unieke verzameling gegevensattributen die een enkel object of interest beschrijven.

[6] Zoals in de begrippenlijst is beschreven gaat het in iedere functionele omvangsmeetmethode om het identificeren van 'typen' en niet over 'voorkomens' van gegevens of functies. In de verdere tekst zal daarom de toevoeging 'type' worden weggelaten als het gaat om de basisprincipes van COSMIC, tenzij het essentieel is om onderscheid te maken tussen 'typen' en 'voorkomens'.

h) Er zijn vier typen data movements. Een Entry verplaatst een gegevensgroep naar de software vanuit een functionele gebruiker. Een Exit verplaatst een gegevensgroep vanuit de software naar een functionele gebruiker. Een Write verplaatst een gegevensgroep vanuit de software naar persistente opslag. Een Read verplaatst een gegevensgroep vanuit de persistente opslag naar de software.

i) Een functioneel proces dient tenminste één Entry data movement en ofwel een Write, ofwel een Exit data movement te bevatten, zodat het minimaal bestaat uit twee data movements.

j) Als een benadering voor meetdoeleinden worden gegevensverwerking subprocessen niet afzonderlijk gemeten. Aangenomen wordt dat de functionaliteit van alle gegevensverwerking wordt meegeteld bij het data movement waar het mee verbonden is.

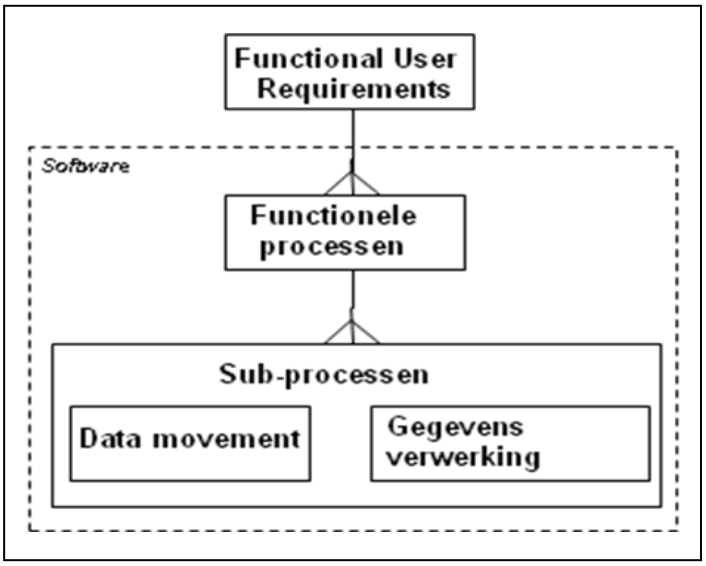
Deze uitgangspunten komen voort uit het volgende algemene beeld van software.

Uitgangspunten (a) tot en met (e)

De taak van software is om te reageren op events die plaatsvinden aan de kant van de grens van de functionele gebruikers, dus in de wereld van de functionele gebruikers. Een functionele gebruiker informeert de software over het plaatsvinden van het event en kan gegevens verzenden over het event. De software moet iets nuttigs doen voor de functionele gebruiker als reactie op het event. Binnen COSMIC wordt dit 'iets nuttigs' een 'functioneel proces' genoemd. Alle FUR voor software kunnen daarom worden uitgedrukt in de vorm van een lijst met events en de bijbehorende functionele processen die de reactie van de software op ieder event uitvoeren.

De uitgangspunten (c) en (d) vertellen ons dat functionele processen kunnen worden gezien als bestaande uit twee typen subprocessen, te weten data movements en gegevensverwerking (data manipulatie). Software kan alleen gegevens verplaatsen en/of manipuleren.

Figuur 2.2.3.1 illustreert de uitgangspunten (b) tot en met (d) van het Generieke Software Model.



Figuur 2.2.3.1 – De structuur van de functional user requirements

Uitgangspunten (f) en (g)

Iedere data movement verplaatst slechts één gegevensgroep, met andere woorden gegevens over een enkel object of interest (een ding dat van belang is voor een functionele gebruiker). Als voorbeeld vanuit het domein van de bedrijfsinformatiesystemen, een relatief eenvoudig functioneel proces om een order in te voeren zou de volgende data movements kunnen bevatten ('objects of interest' staan tussen aanhalingstekens):

- Twee Entries van gegevensgroepen over de 'order' en de 'orderregel' (uitgaande van een order met meerdere items). De eerste van deze Entries met gegevens die het 'order' object of interest beschrijven is degene die het functioneel proces triggert of start.
- Twee Reads van gegevensgroepen over 'klant' en 'product' om te valideren dat de klant mag bestellen en dat de gevraagde producten bestaan en beschikbaar zijn.
- Twee Writes van gegevensgroepen over de 'order' en de 'orderregel' om de ingevoerde gegevens naar persistente opslag te verplaatsen.
- Eén of meer Exits van gegevensgroepen die bijvoorbeeld een 'orderbevestiging' bericht bevatten samen met de totale waarde van de order bevat, een instructie voor het pakhuis om iedere 'order-item' te kunnen vinden, etc.

Elk van deze 'objects of interest' zijn echte of conceptuele dingen in de echte wereld van de functionele gebruikers (menselijke, in dit geval), waarover het stuk software gegevens moet verwerken. Zij moeten worden geïdentificeerd en onderscheiden om de data movements te kunnen identificeren.

Als real-time of embedded software gemeten wordt zijn exact dezelfde uitgangspunten van toepassing, hoewel de functionele gebruiker en het object of interest in de praktijk nauwelijks van elkaar te onderscheiden zijn. Neem bijvoorbeeld een functioneel proces dat de huidige temperatuur van een sensor moet aflezen en neem aan dat de sensor direct met de software kan communiceren. In dit geval is de sensor dus een functionele gebruiker en stuurt de sensor een Entry data movement dat waarschijnlijk maar twee gegevensattributen heeft (het sensor ID en de temperatuur). Deze twee attributen dragen gegevens over de sensor (als object of interest) – hoewel men met evenveel recht kan stellen dat het object of interest het 'ding' is waarvan de sensor de temperatuur heeft gemeten.

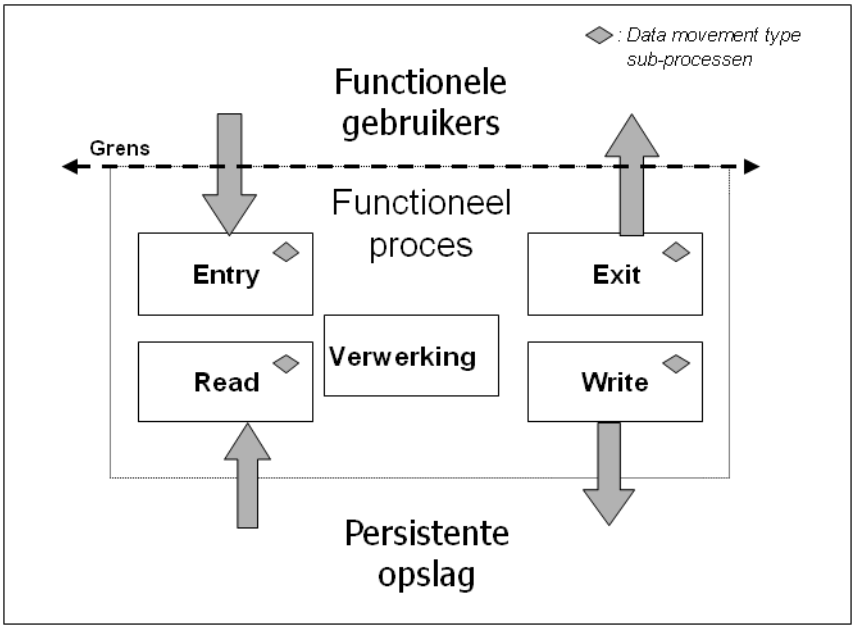
Neem een zeer eenvoudig real-time functioneel proces om met behulp van een sensor de temperatuur te meten en die te vergelijken met een doeltemperatuur. Dit functionele proces wordt met regelmatige tussenpozen getriggered door een kloksignaal en zou kunnen bestaan uit de volgende data movements.

- Een Entry van de klok die het functioneel proces triggert (start)
- Een Entry van de temperatuursensor die het sensor ID en de huidige temperatuur bevat
- Een Read van de doeltemperatuur vanuit persistente opslag (aangenomen dat de doeltemperatuur vastgelegd en gewijzigd kan worden door een ander functioneel proces)
- Een Exit naar het verwarmingselement dat een signaal bevat om het verwarmingselement aan of uit te zetten als de stand van het verwarmingselement gewijzigd moet worden

Merk op dat dit een heel eenvoudig voorbeeld is, waarin alle gegevensgroepen op één na slechts één gegevensattribuut bevatten.

Uitgangspunt (h)

Dit uitgangspunt vertelt ons dat de vier typen data movements onderscheiden worden door hun oorsprong en hun bestemming. Data movement overschrijden ofwel de grens tussen de te meten software en zijn functionele gebruikers (Entries en Exits) ofwel verplaatsen gegevens tussen de software en de persistente opslag (Reads en Writes). Deze verhouding is weergegeven in figuur 2.2.3.2.



Figuur 2.2.3.2 – De componenten van een functioneel proces en sommige van hun verhoudingen

Uitgangspunt (i)

Dit uitgangspunt bepaalt dat een functioneel proces tenminste twee data movements dient te bevatten. Dit volgt uit de voorgaande uitgangspunten. Een functioneel proces dat alleen één data movement ontvangt en er niets mee doet is praktisch nutteloos. Daarom dienen alle functionele processen tenminste één data movement te bevatten die ze informeert over het optreden van een event (een Entry) en tenminste één ander data movement als antwoord (of een nuttig resultaat), ofwel naar een functionele gebruiker (een Exit) ofwel naar persistente opslag (een Write).

Uitgangspunt (j)

Voor meetdoeleinden, en gegeven het domein waarvoor de methode ontwikkeld is, gaat de COSMIC methode uit van een simplificatie van het Generieke Software Model.

De eerste benadering in deze versie van de meetmethode is dat gegevensverwerking subprocessen, zoals geïllustreerd in figuur 2.2.3.2, niet afzonderlijk onderkend worden, maar geacht worden geassocieerd te zijn met of deel uit te maken van specifieke data movement subprocessen. (Vanaf hier zal daarom kortheidshalve worden gesproken over data movements in plaats van data movement subprocessen). De reden voor deze benadering is dat de noodzakelijke concepten en definities om gegevensverwerking te kunnen meten nog steeds onderwerp van veel discussie zijn. De specifieke typen gegevensverwerking functionaliteit die geacht worden onderdeel uit te maken van ieder type data movement wordt beschreven in de subsecties over de gegevensverwerking in het Meethandboek.

Gegeven deze aanname kunnen we zien waarom de standaard COSMIC methode toepasbaar is bij het meten van ‘movement-rijke’ typen software, zoals de meeste bedrijfsinformatiesystemen en veel real-time software, maar niet geschikt is voor het meten van ‘manipulatie-rijke’ (of ‘algoritme-rijke’) software. Het Meethandboek wijst ook op de noodzaak voorzichtig te zijn met het meten van zeer kleine stukken software en met name kleine wijzigingen in de software waar de aanname van uitgangspunt (j) mogelijk niet meer valide is.

Het Meethandboek voorziet ook in een mechanisme om een lokale uitbreiding op de COSMIC methode te definiëren waarmee een organisatie in staat gesteld wordt om gegevensverwerking

functionaliteit expliciet te waarderen, mocht men dit willen. Voor dergelijke gevallen zijn ook speciale rapportage conventies beschreven.

Door de concepten en hun definities en de uitgangspunten en regels van de COSMIC meetmethode te gebruiken, kunnen de FUR zoals die vanuit de artefacten van een stuk software geëxtraheerd zijn, gebruikt worden om een instantie te maken van het Generieke Software Model. Dit geïnstantieerde model bevat alle elementen die nodig zijn om de functionele omvang te meten, terwijl informatie, die niet relevant is voor het bepalen van de functionele omvang, verborgen wordt.

De meetregels en processen worden vervolgens op dit geïnstantieerde Generieke Software Model toegepast om de functionele omvang van het stuk software te bepalen – zie 2.3.3 voor de meetregels.

2.3 Overzicht van het COSMIC meetproces

Drie afzonderlijke en gerelateerde fasen zijn nodig om de functionele omvang van een stuk software te meten:

1. Het vaststellen van de meetstrategie, gebruik makende van de principes van het Software Context Model.
2. Het afbeelden van de artefacten van de te meten software op het Generieke Software Model.
3. Het meten van de specifieke elementen van dit model.

2.3.1 De meetstrategie fase

Voordat de meting wordt gestart moet de analist het met de sponsors van de meting eens zijn over (a) het doel van de meting, (b) de scope van ieder stuk software dat gemeten moet worden (c) de functionele gebruikers van ieder stuk software en dus de grens van ieder stuk software en (d) de granulariteit waarop de meting moet worden uitgevoerd. Het vaststellen van een duidelijk doel (a) van de meting is belangrijk omdat het de drie parameters (b), (c) en (d) bepaalt. Het vaststellen van deze parameters zal meestal een iteratief proces zijn.

(a) Het doel van de meting

Het doel van de meting geeft aan waarom de meting wordt uitgevoerd en waarvoor de resultaten gebruikt gaan worden. Dit zal op zijn beurt niet alleen helpen om de overige drie parameters van de meetstrategie vast te stellen, maar ook, bijvoorbeeld, om de vereiste nauwkeurigheid van de meting vast te stellen. Zoals in het hoofdstuk over vroegtijdig meten in het document 'Advanced & Related Topics' kan worden gelezen, is het mogelijk om de functionele omvang vast te stellen met behulp van een indicatieve variant van de COSMIC methode. Deze variant kan vroegtijdig in de project levenscyclus worden toegepast voordat alle requirements zijn uitgewerkt tot een detailniveau waarop het meten met de exacte regels van de methode mogelijk is.

(b) De scope van de software die gemeten moet worden

De totale scope van de software die gemeten moet worden volgt uit het doel van de meting. De totale scope beschrijft welke software functionaliteit in de meting moet worden meegenomen (en welke wordt uitgesloten). Afhankelijk van het doel kan de totale scope worden opgedeeld in een aantal afzonderlijk stukken software functionaliteit waarbij elke van deze afzonderlijke stukken binnen zijn eigen scope wordt gemeten.

Een dergelijke onderverdeling van de totale scope zou ten eerste nodig zijn als de totale scope software bevat die zich binnen meer dan één laag bevindt. Dit omdat ieder stuk te meten software zich volledig binnen één laag moet bevinden. Ten tweede zou een onderverdeling ook nodig kunnen zijn als het doel van de meting bijvoorbeeld het begroten van een software project betreft en de totale te meten software bestaat uit een aantal verschillende componenten die met verschillende technieken en/of technologieën worden ontwikkeld en/of op verschillende technische platforms worden geïmplementeerd, en/of door verschillende teams worden ontwikkeld.

(c) De functionele gebruikers en de grens van ieder stuk te meten software

De functionele gebruikers van ieder stuk software kunnen worden geïdentificeerd door middel van het onderzoeken van de gegevensstromen die de software in en uit gaan, zoals impliciet of expliciet beschreven in de FUR, waarbij met het doel van de meting rekening gehouden moet worden. De functionele gebruikers zijn de zenders en/of de beoogde ontvangers van de gegevens.

In de meeste gevallen is het identificeren van de functionele gebruikers vanuit het doel van de meting en vanuit de FUR een eenduidige zaak. In uitzonderingsgevallen kunnen de functionele gebruikers variëren, afhankelijk van het doel van de meting. Een voorbeeld wordt gegeven in de paragraaf over functionele gebruikers van het meethandboek, waarbij een geval wordt beschreven waar er een keuze mogelijk zou kunnen zijn.

Als de functionele gebruikers bekend zijn, dan kan de grens – de conceptuele interface tussen de functionele gebruikers en het stuk software dat wordt gemeten – eenvoudig worden vastgesteld.

(d) De granulariteit van de metingen

De granulariteit van de FUR van een stuk software waarop metingen normaal gesproken moeten worden uitgevoerd is de wijze waarop de functionele processen zijn vastgesteld en de onderverdeling in data movements is gedefinieerd.

Als het doel is om de FUR te meten van een volledig gespecificeerd of een bestaand stuk software, dan is de granulariteit normaal gesproken duidelijk wanneer de functionele processen zijn geïdentificeerd.

Aan de andere kant kan het zijn dat de FUR gemeten moeten worden in de beginfase van de softwareontwikkeling, d.w.z. voordat alle individuele functionele processen zijn onderkend en de data movements zijn gedefinieerd. Om de zaken verder te compliceren kunnen de verschillende onderdelen van de FUR op het moment dat de meting moet plaatsvinden van granulariteit verschillen. In deze omstandigheden zullen soms lokale ‘functionele units’ moeten worden geïdentificeerd en worden gemeten, waarbij een methode wordt gebruikt om deze units te schalen naar de granulariteit waarop de functionele processen worden geïdentificeerd en gemeten.

Nadat de stappen (a) tot en met (d) zijn uitgevoerd, is er enige iteratie benodigd. Als sommige requirements bijvoorbeeld verder worden gedetailleerd, dan kan dit leiden tot een verfijning van de scope van het stuk software dat wordt gemeten.

Voor een volledige discussie met definities en meer voorbeelden van doel, scope, functionele gebruikers en granulariteit wordt verwezen naar het hoofdstuk over de Meetstrategie van het Meethandboek.

2.3.2 De Vertaalslag

De input voor de Vertaalslag bestaat uit de FUR van ieder te meten stuk software die zijn afgeleid uit de artefacten (zoals ze zijn gevonden of gedocumenteerd binnen de organisatie of zoals ze uit de bestaande fysieke software zijn afgeleid) en rekening houden met het Software Context Model. De output van de Vertaalslag is een instantie van het Generic Software Model.

De stappen om een Generic Software Model in de Vertaalslag te initiëren zijn:

- identificeer de events in de wereld van de functionele gebruiker waarop de software moet reageren en identificeer op deze manier de functionele processen.

- Identificeer de data movements (Entries, Exits, Reads en Writes) van ieder functioneel proces, die afhankelijk zijn van de geïdentificeerde gegevensgroepen die worden verplaatst.

Een volledige discussie met definities en voorbeelden van de concepten en stappen binnen de Vertaalslag wordt gegeven in het bijbehorende hoofdstuk van het Meethandboek.

2.3.3 De meetfase

De meetfase heeft als invoer een instantie van het Generic Software Model en produceert een numerieke waarde, waarbij gebruik wordt gemaakt van een gedefinieerde verzameling regels en processen. De omvang van deze numerieke waarde is recht evenredig met de functionele omvang van het model, gebaseerd op het volgende uitgangspunt:

UITGANGSPUNT – Het COSMIC meet uitgangspunt
De functionele omvang van een stuk software is rechtevenredig met het aantal aanwezige data movements.

De kenmerken van de verzameling regels en processen leiden tot die deze numerieke waarden, zijn:

Kenmerk 1 – Eenheid van meten

De meetstandaard, te weten 1 CFP (COSMIC Functie Punt) is bij conventie gedefinieerd als equivalent van één enkele data movement.

Kenmerk 2 – Optelbaarheid van omvang binnen een gegeven meet scope

De functionele omvang van een functioneel proces is gedefinieerd als de rekenkundige som van het aantal betrokken data movements. Uitbreiding van het begrip: de functionele omvang van ieder stuk software met een specifieke meetscope in iedere laag van het software model is de rekenkundige som van de functionele omvang van de functionele processen binnen dat stuk software.

Kenmerk 3 – Omvang van wijziging(en) van een stuk software

De functionele omvang van iedere benodigde functionele wijziging(en) van een stuk software is bij conventie de rekenkundige som van alle data movements die moeten worden toegevoegd, gewijzigd en verwijderd als consequentie van de benodigde wijziging.

Kenmerk 4 – De minimum en maximum omvang van een functioneel proces

Volgens deze kenmerken, zoals al bepaald in uitgangspunt (g) van het generieke software model, is de minimum omvang van een enkel functioneel proces 2 CFP, omdat het kleinste functioneel proces tenminste één Entry moet hebben (als input) en ofl één Exit (als output) of één Write (als een alternatief nuttige uitkomst).

Omdat een wijziging slechts één data movement kan betreffen, is de minimum omvang van een wijziging op een functioneel proces 1 CFP.

Verder is er volgens deze kenmerken geen bovengrens aan de functionele omvang van een functioneel proces en dus ook niet aan de functionele omvang van een stuk software.

Verdere uitgangspunten en gedetailleerde regels en processen van de Meetfase voor het vaststellen van de functionele omvang van de FUR van een stuk software, weergegeven in het Generieke Software Model, staan in de gerelateerde hoofdstukken van het Meethandboek en zijn samengevat in de daar bijhorende appendices B en C.

Appendix A

COSMIC WIJZIGINGSVERZOEK EN COMMENTAAR PROCEDURE

De COSMIC Measurement Practices Committee (MPC) wil graag feedback, opmerkingen en indien nodig wijzigingsverzoeken ontvangen voor de COSMIC methode. Deze appendix beschrijft hoe met de COSMIC MPC gecommuniceerd kan worden.

Alle communicatie met de COSMIC MPC moet worden verstuurd in het engels per e-mail aan het volgende adres:

mpc-chair@cosmic-sizing.org

Al het commentaar met betrekking tot de Nederlandstalige versie kan worden verstuurd per e-mail aan

cosmic@nesma.org

Informele algemene feedback en opmerkingen

Informele opmerkingen en/of feedback met betrekking tot de COSMIC documentatie, zoals de moeilijkheden om de methode te begrijpen of toe te passen, aanbevelingen voor algemene verbetering, etc kunnen aan bovenstaande adressen worden gestuurd. De meldingen worden vastgelegd en worden binnen twee weken na ontvangst bevestigd. De MPC kan niet garanderen dat naar aanleiding van algemene opmerkingen actie wordt ondernomen.

Formele change requests

Daar waar de COSMIC lezer denkt dat er een fout in de tekst staat of er behoefte aan verduidelijking is of dat een deel van de tekst zou moeten worden verbeterd, dan kan een formele wijzigingsverzoek (Change Request - CR) worden ingediend.

Formele CR's worden gelogd en binnen twee weken na ontvangst bevestigd. Aan iedere CR zal dan een serienummer worden toegekend en het zal worden verspreid onder de leden van de COSMIC MPC, een wereldwijde groep met experts op het gebied van de COSMIC methode. De normale review cyclus duurt minimaal een maand en kan langer duren als de CR moeilijk op te lossen is.

De uitkomst van de review kan zijn dat de CR wordt geaccepteerd, of wordt afgewezen, of 'dat er eerst verder gediscussieerd zal moeten worden. Dit laatste geval bijvoorbeeld als er een afhankelijkheid is op een andere CR. De uitkomst wordt zo spoedig mogelijk gecommuniceerd naar de indiener van het CR.

Een formele CR zal alleen geaccepteerd worden als het is gedocumenteerd met de volgende informatie:

- Naam, functie en organisatie van de persoon die het CR indient
- De contact informatie van de persoon die het CR indient
- De datum van indiening

- Een algemene verklaring van het doel van het CR (bv. ‘tekst zou verbeterd moeten worden’)
- De daadwerkelijke tekst die verbeterd, die vervangen of die verwijderd moet worden. (of een duidelijke verwijzing hiernaar)
- Voorgestelde nieuwe of gewijzigde tekst
- Volledige uitleg van waarom de wijziging nodig is.

Een formulier om een CR in te dienen is verkrijgbaar van de site www.cosmic-sizing.org.

De beslissing van de COSMIC MPC betreffende de uitkomst van een CR review en, indien geaccepteerd in welke versie van de COSMIC documentatie de CR wordt toegepast, is definitief.

Vragen over de toepassing van de COSMIC methode

De COSMIC MPC betreurt het dat het niet in staat is om vragen te beantwoorden met betrekking tot het gebruik of de toepassing van de COSMIC methode. Er bestaan commerciële organisaties die training, tools en consultancy bieden voor de methode. Bezoekt u alstublieft de site www.cosmic-sizing.org voor meer details.