Technical Sciences
Eemsgolaan 3
9727 DW Groningen
Postbus 1416
9701 BK Groningen

www.tno.nl

T +31 88 866 70 00
F +31 88 866 77 57
infodesk@tno.nl

TNO-rapport

TNO 2013 R11002 | versie 1.0

Aanpak IT architectuur en ontwerp voor ketentransparantie in de melkveesector

Datum	3 juli 2013
Auteur(s)	Matthijs Vonder, Edwin Jan Harmsma
Reviewer	Bram van der Waaij
Aantal pagina's	15
Opdrachtgever	LTO Noord
Projectnaam	BIC Meten is Weten
Projectnummer	052.03031/01.02 en 03

Inhoudsopgave

1 Inleiding 3
1.1 Achtergrond informatie tav SDF context 3
1.2 Overgedragen methode voor IT-architectuur en ontwerp 3
1.3 Doel van dit document 4
1.4 Opbouw document 5
1.5 Verklarende woordenlijst 5

2 “4+1 viewmodel” - methode 6
2.1 De 4+1 views 6
2.1.1 Samenhang tussen de gezichtspunten 7
2.2 Een voorbeeld 8
2.2.1 Logisch gezichtspunt 8
2.2.2 Proces gezichtspunt 9
2.2.3 Fysiek gezichtspunt 9
2.2.4 Development gezichtspunt 10

3 Quint - methode 11
3.1 Functionality 12
3.2 Reliability 12
3.3 Usability 13
3.4 Efficiency 13
3.5 Maintainability 14
3.6 Portability 14

4 Bereik van kennisoverdracht 15

1 Inleiding

De kennisoverdracht tav de “Aanpak IT architectuur en ontwerp voor ketentransparantie in de melkveesector” heeft plaatsgevonden binnen het *Branche Innovatie Contract (BIC) Meten is weten*. Dit BIC is uitgevoerd in de context van het project Smart Dairy Farming, maar bedoeld voor de gehele melkvee sector en is en wordt op verschillende plaatsen gedissemineerd.

1.1 Achtergrond informatie tav SDF context

Het doel van het project Smart Dairy Farming (SDF) is het ondersteunen van melkveehouders bij de verzorging van individuele dieren, om daarmee een goede gezondheid, een langer leven en een langere productietijd van de koe te bewerkstelligen. Daartoe ontwikkelt SDF sensoren en andere technologische hulpmiddelen, beslismodellen, procesbeschrijvingen en adviesproducten. Deze hulpmiddelen en diensten stellen melkveehouders in staat betere keuzes te maken op het gebied van gezondheid, vruchtbaarheid en voeding.

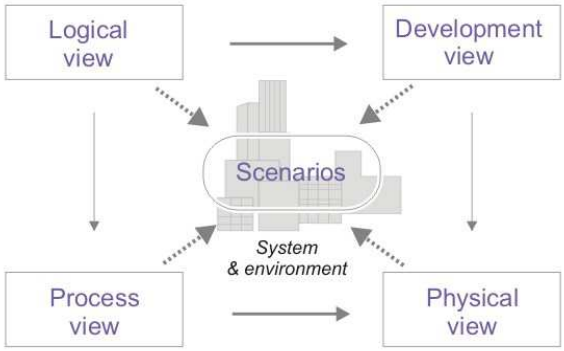
Als gevolg van deze verbeterde keuzes leven koeien langer en zijn ze langer productief. Bij de afronding van de pilot in 2014 is er bovendien sprake van een infrastructuur en een netwerk van relaties die een effectieve uitwisseling van gegevens, ervaringen en kennis tussen melkveebedrijven en partners mogelijk maakt. De diensten worden vervolgens op zoveel mogelijk melkveebedrijven in Nederland geïmplementeerd (bron: www.smartdairyfarming.nl).

Voor het realiseren van een versterkte duurzame melkveehouderijketen zoals hierboven beschreven, is een zogenaamde keteninfrastructuur nodig. In de praktijk betekent dit concreet dat een flexibele architectuur, ingebed in de melkveehouderijketen nodig is die makkelijk nieuwe modellen en diensten ontsluit. Hiertoe bestaat binnen het project Smart Dairy Farming het Werkpakket 4 *Ketentransparantie*, dat zich richt op samenwerking in de keten en het ontwikkelen van de architectuur die deze ketensamenwerking faciliteert.
In dit SDF Werkpakket 4, waarvan TNO de trekker is, zijn verder verschillende partijen vertegenwoordigd van het SDF project, namelijk FrieslandCampina, CRV, Agrifirm, Rovecom, WUR en S&S Systems.

1.2 Overgedragen methode voor IT-architectuur en ontwerp

Het doel van het traject is kennis over te dragen waarmee een IT architectuur en ontwerp gemaakt kan worden tbv ketentransparantie in de melkveesector. De hiertoe gebruikte en overgedragen methode is voornamelijk gebaseerd op het “4+1 viewmodel”. Samen met branche organisatie LTO en de SDF-projectleider is afgesproken dat in dit document ook kort de “Quint” methode beschreven wordt, zodat er desgewenst vanuit SDF mee gewerkt kan worden .

Het **"4+1 viewmodel**" is ontworpen door Philippe Kruchten[1,2] voor software-intensieve systemen, gebaseerd op het idee van meerdere parallelle aanzichten zoals weergegeven in het onderstaande diagram.



Figuur 1 Principe van het 4+1 viewmodel

De views worden gebruikt om het system te beschrijven vanuit gezichtspunten van de verschillende belanghebbenden, zoals eindgebruikers, ontwikkelaars en projectmanagers. De vier gezichtspunten van het model zijn het: logische-, ontwikkelings-, proces- en fysieke gezichtspunt. De additionele geselecteerde use cases of scenario's worden gebruikt om de architectuur te illustreren als het "+1" gezichtspunt. De 4+1 methode is toegelicht en gebruikt tijdens diverse werksessies van SDF Werkpakket 4 tussen januari 2013 en juli 2013. Gedetailleerde uitleg over de 4+1 methode kan gevonden worden in het volgende hoofdstuk

Quint staat voor Quality in Information Technology, wat een uitbreiding is van het kwaliteitsmodel beschreven in de ISO/IEC 9126 standaard[3]. Het gaat daarbij om het specificeren van gebruikersgerichte eisen die gesteld worden aan een software product, daarnaast wordt beschreven hoe deze geverifieerd kunnen worden. De Quint methode is in een eerste werksessie kort aangestipt en in dit document in hoofdstuk 3 kort toegelicht, maar ivm de beschikbare tijd en geld niet meer binnen het BIC in workshops behandeld.

1.3 Doel van dit document

Dit document beschrijft de aanpak en kennisoverdracht van de methoden om tot een IT-architectuur en ontwerp te komen voor ketentransparantie in de melkvee sector.
De concrete resultaten van het toepassen van de 4+1 methode worden in het SDF project uitgewerkt en komt beschikbaar in een apart document: *"IT architectuur en ontwerp voor SDF"*.

[1] http://www.cs.ubc.ca/~gregor/teaching/papers/4+1view-architecture.pdf
[2] http://en.wikipedia.org/wiki/4%2B1
[3] De eerste versie is geschreven door CIBIT Adviseurs in 1996, in een document genaamd "Kwaliteit van softwareprodukten".

1.4 Opbouw document

In het volgende hoofdstuk wordt de 4+1 viewmodel methode beschreven. Hoofdstuk 3 beschrijft kort de Quint methode. Tot slot beschrijft hoofdstuk 4 het bereik van de kennisoverdracht.

1.5 Verklarende woordenlijst

Begrip	Uitleg
Architectuur	Een bouwtekening op hoofdlijnen die recht doet aan de eisen, wensen en prioriteiten van direct betrokkenen
Persona	Een persona is een archetype van een gebruiker, ofwel een karakterisering van een bepaald type van gebruiker. Dit fictieve karakter representeert een doelgroep
Requirements	Eisen en wensen
SOP	Standard Operating Procedure: een (geschreven) werkinstructie die (in detail) voorschrijft welke handeling uitgevoerd dient te worden
Use case	Een use Case beschrijft ‘wie’ met het betreffende systeem ‘wat’ kan doen. Uit deze use cases worden eisen en wensen gedestilleerd om de use cases te kunnen uitvoeren

2 “4+1 viewmodel” - methode

In dit hoofdstuk wordt een gedetailleerd overzicht van de verschillende views binnen de 4+1 viewmodel methode gegeven. Daarbij worden er een aantal SDF gerelateerde voorbeeld diagrammen weergegeven die typisch binnen een bepaalde view worden gebruikt.

2.1 De 4+1 views

De onderstaande tabel heeft als doel om elk gezichtspunt concreet en in detail beschrijven. Het is een korte technische samenvatting van de oorspronkelijke definities in het 4+1 paper[1]. Meer uitleg en voorbeelden van de gezichtspunten staan in de volgende paragraaf.

Gezichtspunt	Beschrijving
Logisch	Binnen deze view staan de technische bouwblokken centraal. Het gaat om de objecten die uit het probleem domein geabstraheerd worden. Typisch omvat dit gezichtspunt klasse diagrammen en/of E-R diagrammen. Een bijkomend aspect van dit gezichtspunt zijn primaire technische interacties tussen de ontworpen bouwblokken. Denk hierbij aan uitwerkingen in de zin van technische tijdvolgorde diagrammen (sequence diagrams) die de interactie en samenhang tonen van een groep objecten. Beschrijvingen van bericht-types en standaarden passen ook in deze view. Denk hierbij bijvoorbeeld aan XML Document Type Definitions die een communicatie bericht definiëren.
Proces	Waar de logische view de technische bouwblokken en bijbehorende interacties bevat, belicht de proces view de aanwezige processen op een hoger abstractie niveau. Dit abstractie niveau sluit in het beginsel nauw aan bij de gedefinieerde scenario’s. In dit geval geeft de proces view de exacte stappen en keuzes uit de use cases schematisch weer. Naarmate de architectuur verder uitgewerkt wordt komen de bouwblokken en objecten uit de logical view meer in zicht. De keuzes en activiteiten worden veelal weergegeven in een Activity diagram. De proces view laat dus wel berichten en informatie stromen zien, maar zoomt niet in op de daadwerkelijke bericht inhoud.
Fysiek	Het doel van het fysieke gezichtspunt is om de gedefinieerde processen (en bouwblokken) daadwerkelijk te “mappen” naar fysieke machines. Denk hierbij aan besluiten om geen “kennis” of “rekencapaciteit” op een bepaalde locatie te plaatsen, maar dit op een centraal punt te doen. Een ander concreet aspect wat aan bod komt in deze view zijn de organisatie en management vraagstukken. Het fysieke gezichtspunt brengt scheidingslijnen aan in verantwoordelijkheden tussen organisaties met betrekking tot het beheer en onderhoud van de fysiek draaiende systemen.

Gezichtspunt	Beschrijving
Development	Het development gezichtspunt heeft als doel het beschrijven en structureren van de gehele ontwikkel omgeving. Het meest belangrijke onderdeel hiervan is het structureren van de te implementeren onderdelen door het ontwerpen van een package indeling. Een package structuur zorgt ervoor dat verschillende ontwikkel teams zo onafhankelijk mogelijk kunnen opereren, en moet voorkomen dat de zelfde functionaliteit op verschillende plaatsen opnieuw geprogrammeerd worden. “Software reuse” wordt bijvoorbeeld bevorderd door een overzichtelijke en simpele package indeling en voorschriften voor het gebruik van al beschikbare software bibliotheken. Een ander onderdeel van de development view zijn de project planning, resource toekenning en voortgang bewaking. Deze onderwerpen vallen doorgaans onder de verantwoordelijkheid van een of meerdere project leiders, maar ook hier moet in de software architectuur rekenschappen mee gehouden worden. Tot slot kunnen in een zeer gedetailleerde uitwerking van het development gezichtspunt ook afspraken worden gemaakt over testbaarheid, het afhandelen van errors en excepties, en bijvoorbeeld logging en configuratie eisen.
Scenario’s	De “+1” view voegt de informatie uit de 4 andere views samen tot één geheel. Het doel hiervan is om zowel nieuwe ontbrekende architectuur aspecten te ontdekken als het valideren van de architectuur nadat een versie van het ontwerp af is. Scenario’s kunnen gezien worden als een generieke abstractie van een set use cases. De selectie van de architectuur relevante scenario’s speelt hierbij een belangrijke rol.

2.1.1 Samenhang tussen de gezichtspunten

De beschreven gezichtspunten staan niet op zich zelf maar hebben interactie en overlap met elkaar. Onderstaand worden drie belangrijke interacties beschreven.

Van logisch naar proces

In de samenhang tussen het logische en proces gezichtspunt komen vooral concurrency aspecten aan bod. In het logische gezichtspunt wordt vooral ontworpen in termen van data modellen en klasse definities, waarbij het proces gezichtspunt de informatie stromen en uitvoer gedrag beschrijft.

Een voorbeeld van een beslissing die tussen deze twee views in hangt is het besluit om een reeks aanvragen *stateless* te maken. Het stateless maken van een bepaald soort aanvraag heeft invloed op zowel het logische datamodel als het proces zelf. Vanuit de het logische gezichtspunt komt nu de eis naar voren dat iedere aanvraag zelf-omschrijvend moet zijn, waarbij het proces gezichtspunt de extra eis krijgt dat de aanvraag niet mag afhangen op eerdere of latere aanvraag.

Van logisch naar development
Het logische gezichtspunt ligt heel dicht tegen het development gezichtspunt aan. De functionele blokken uit logische gezichtspunt worden immers ingedeeld zoals beschreven wordt door het development gezichtspunt. Toch zijn de belangen en afwegingen die spelen binnen beide views verschillend. Het logische gezichtspunt definieert de bouwblokken vanuit een functioneel perspectief, waarbij de development puur let op hoe deze bouwblokken uiteindelijk geïmplementeerd moeten gaan worden op een zo kosteneffectief mogelijke manier.

Doorgaans groeit het gat tussen de logische en development view als projecten groter worden[1].

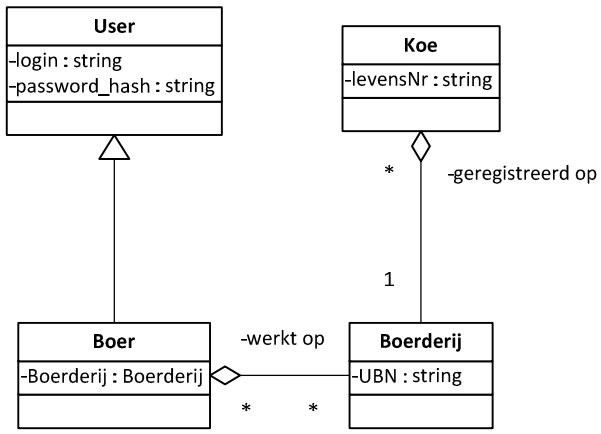
Van proces naar fysiek
De relatie tussen de proces view en de fysieke view moet er voor zorgen dat elk proces de juiste toekenning krijgt naar een fysieke of virtuele machine. Denk hierbij bijvoorbeeld aan vragen over disaster recovery en fault-tollerance. Een concreet voorbeeld is het toevoegen van een “hot-spare” component als fail-over mechanisme.

2.2 Een voorbeeld

Deze sectie geeft voor elke view een beknopt en SDF gerelateerd voorbeeld. Het doel hiervan is om na de view beschrijvingen in de vorige sectie een concrete uitwerking te laten zien in de vorm van een “typisch” diagram voor de desbetreffende view. De getoonde voorbeeld diagrammen geven echter geen volledig beeld en zijn puur bedoeld om een beter inzicht te krijgen in de 4+1 ontwerp methode zelf. De echte diagrammen zijn te vinden in de architectuur beschrijving in het document: *“IT architectuur en ontwerp voor SDF”*.

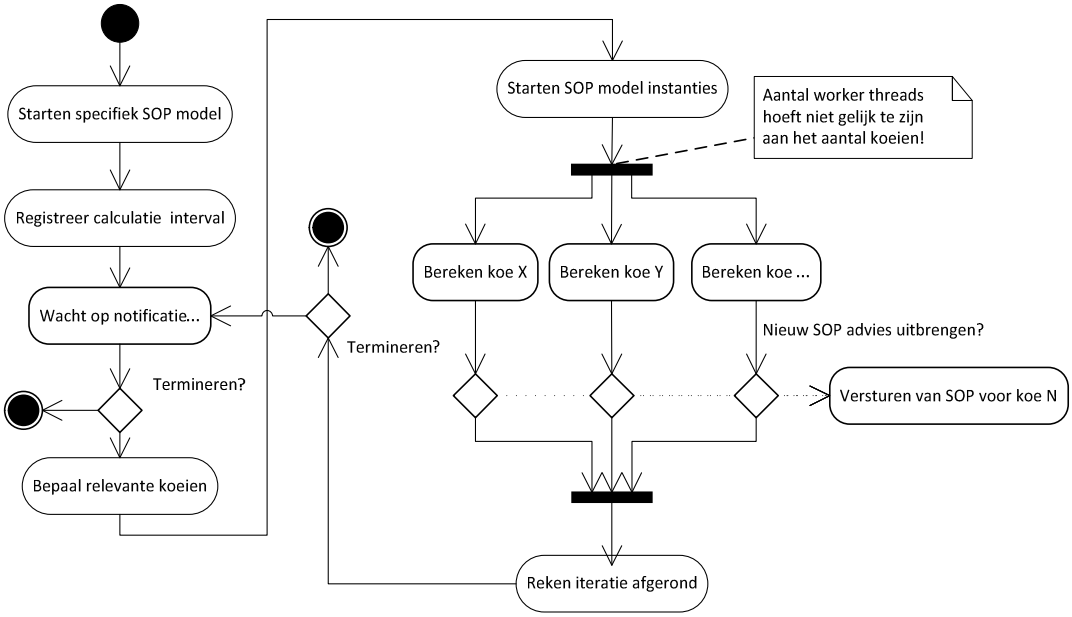
2.2.1 Logisch gezichtspunt

Een UML klasse diagram met daarin de evidente relaties tussen boeren, boerderijen en koeien:



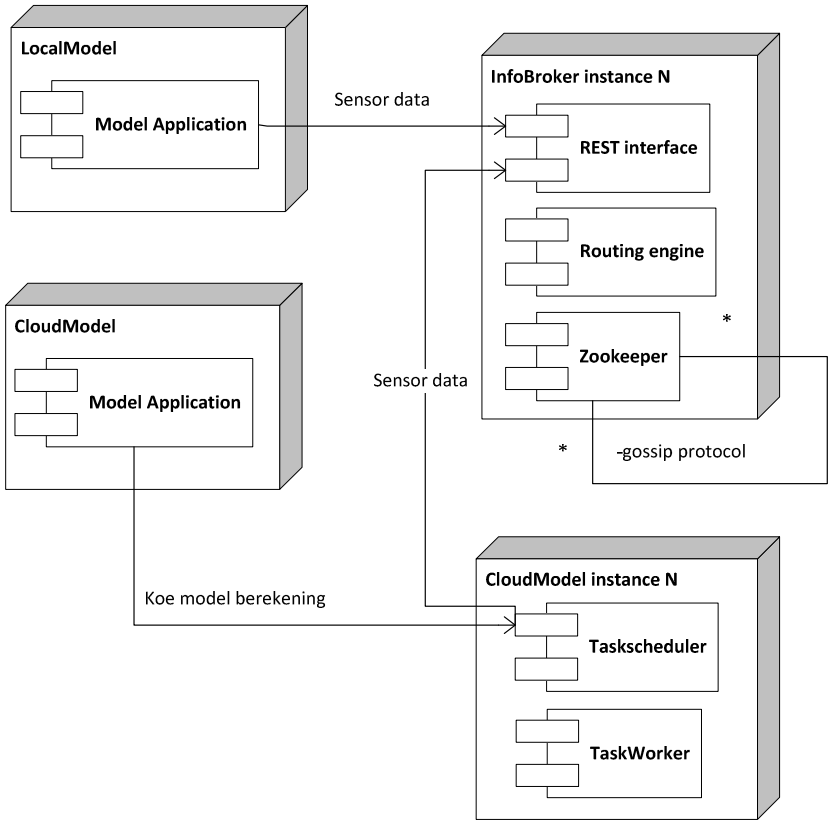
2.2.2 Proces gezichtspunt

Een UML activiteiten diagram met daarin het proces van het draaien van een SOP model:



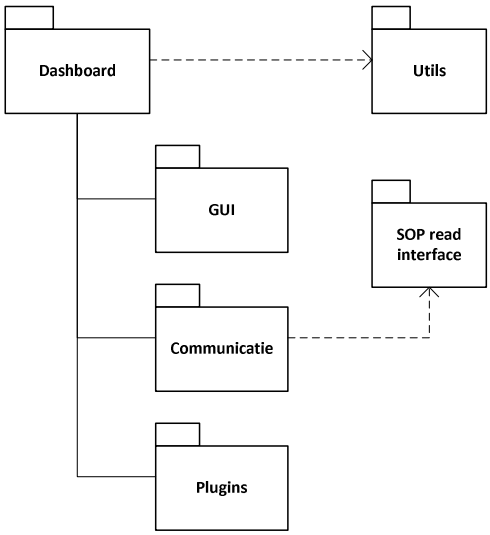
2.2.3 Fysiek gezichtspunt

Een UML deployment diagram dat de processen toekent aan fysieke machines. Distributie en op afstand uitgevoerde processen worden zichtbaar.



2.2.4 Development gezichtspunt

Een zeer versimpeld UML package diagram van de Dashboard applicatie:



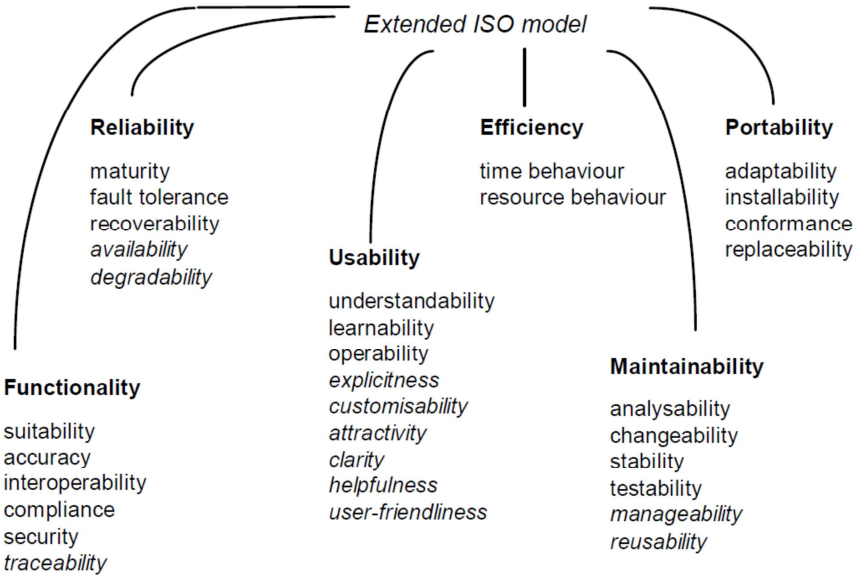
Een ander voorbeeld van een afspraak in de development architectuur kan zijn: “Er wordt binnen de implementatie gebruikt gemaakt van de PEP 257 conventie[4].” Rationale: De driver *maintainability* stelt als eis dat er geautomatiseerd documentatie gegenereerd kan worden.

[4] http://www.python.org/dev/peps/pep-0257/

3 Quint - methode

Quint staat voor Quality in Information Technology, wat een uitbreiding is van het kwaliteitsmodel beschreven in de ISO/IEC 9126 standaard[5]. Het gaat daarbij om het specificeren van gebruikersgerichte eisen die gesteld worden aan een software product, daarnaast wordt beschreven hoe deze geverifieerd kunnen worden.

Het overzicht in de volgende figuur is overgenomen uit bovengenoemde boek en geeft de "characteristics" van het Extended ISO-model weer in overeenkomst met de ISO 9126 hiërarchie. De eigenschappen die toegevoegd zijn aan ISO 9126 worden cursief weergegeven.



Figuur 2: Hiërarchisch overzicht van het Extended ISO model

Characteristics:

- **functionality**: a set of attributes that bear on the existence of a set of functions and their specified properties. The functions are those that satisfy stated or implied needs;
- **reliability**: a set of attributes that bear on the capability of software to maintain its level of performance under stated conditions for a stated period of time;
- **usability**: a set of attributes that bear on the effort needed for use, and on the individual assessment of such use, by a stated or implied set of users;
- **efficiency**: a set of attributes that bear on the relationship between the level of performance of the software and the amount of resources used, under stated conditions;
- **maintainability**: a set of attributes that bear on the effort needed to make specified modifications;
- **portability**: a set of attributes that bear on the ability of software to be transferred from one environment to another.

[5] De eerste versie is geschreven door CIBIT Adviseurs in 1996, in een document genaamd “Kwaliteit van softwareprodukten”.

In de vollende paragrafen worden de bijbehorende “sub characteristics” kort beschreven.

3.1 Functionality

Functionality: a set of attributes that bear on the existence of a set of functions and their specified properties. The functions are those that satisfy stated or implied needs.

Subcharacteristic	**Description**
Suitability	Attribute of software that bears on the presence and appropriateness of a set of functions for specified tasks.
Accuracy	Attributes of software that bear on the provision of right or agreed results or effects.
Interoperability	Attributes of software that bear on its ability to interact with specified systems.
Compliance	Attributes of software that make the software adhere to application related standards, conventions or regulations in laws and similar prescriptions.
Security	Attributes of software that bear on its ability to prevent unauthorized access, whether accidental or deliberate, to programs and data.
Traceability	Attributes of software that bear on the effort needed to verify correctness of data processing on required points.

3.2 Reliability

Reliability: a set of attributes that bear on the capability of software to maintain its level of performance under stated conditions for a stated period of time.

Subcharacteristic	**Description**
Maturity	Attributes of software that bear on the frequency of failure by faults in the software.
Fault tolerance	Attributes of software that bear on its ability to maintain a specified level of performance in cases of software faults or of infringement of its specified interface.
Recoverability	Attributes of software that bear on the capability to re-establish its level of performance and recover the data directly affected in case of a failure and on the time and effort needed for it.
Availability	Attributes of software that bear on the amount of time the product is available to the user at the time it is needed.
Degradability	Attributes of software that bear on the effort needed to reestablish the essential functionality after a breakdown.

3.3 Usability

Usability: a set of attributes that bear on the effort needed for use, and on the individual assessment of such use, by a stated or implied set of users.

Subcharacteristic	Description
Understandability	Attributes of software that bear on the users' effort for recognizing the logical concept and its applicability.
Learnability	Attributes of software that bear on the users' effort for learning its application (for example, control, input, output).
Operability	Attributes of software that bear on the users' effort for operation and operation control.
Explicitness	Attributes that bear on the clarity of the software product with regard to its status (progression bars, etc.).
Customisability	Attributes of software that enable the software to be customized by the user to reduce the effort required for use and increase satisfaction with the software.
Attractivity	Attributes of software that bear on the satisfaction of latent user desires and preferences, through services, behaviour and presentation beyond actual demand.
Clarity	Attributes of software that bear on the clarity of making the user aware of the functions it can perform.
Helpfulness	Attributes of software that bear on the availability of instructions for the user on how to interact with it.
User-friendliness	Attributes of software that bear on the users’ satisfaction.

3.4 Efficiency

Efficiency: a set of attributes that bear on the relationship between the level of performance of the software and the amount of resources used, under stated conditions.

Subcharacteristic	Description
Time behaviour	Attributes of software that bear on response and processing times and on throughput rates in performing its function.
Resource behaviour	Attributes of software that bear on the amount of resources used and the duration of such use in performing its function.

3.5 Maintainability

Maintainability: a set of attributes that bear on the effort needed to make specified modifications.

Subcharacteristic	Description
Analysability	Attributes of software that bear on the effort needed for diagnosis of deficiencies or causes of failures, or for identification of parts to be modified.
Changeability	Attributes of software that bear on the effort needed for modification, fault removal or for environmental change.
Stability	Attributes of software that bear on the risk of unexpected effect of modifications.
Testability	Attributes of software that bear on the effort needed for validating the modified software.
Manageability	Attributes of software that bear on the effort needed to (re)establish its running status.
Reusability	Attributes of software that bear on its potential for complete or partial reuse in another software product.

3.6 Portability

Portability: a set of attributes that bear on the ability of software to be transferred from one environment to another.

Subcharacteristic	Description
Adaptability	Attributes of software that bear on the opportunity for its adaptation to different specified environments without applying other actions or means than those provided for this purpose for the software in question.
Installability	Attributes of software that bear on the effort needed to install the software in a specified environment.
Conformance	Attributes of software that make the software adhere to standards or conventions relating to portability.
Replaceability	Attributes of software that bear on the opportunity and effort of using it in the place of specified other software in the environment of that software.

4 Bereik van kennisoverdracht

De 7 deelnemers aan Werkpakket 4 hebben de “4+1 viewmodel” methode in een 5 tal workshops overgedragen gekregen en deze actief toegepast om tot de resultaten te komen.

Op het AgroConnect zomersymposium van 6-6-2013 is de aanpak en het tussenresultaat op hoofdlijnen toegelicht aan de 80-100 deelnemers, voornamelijk IT experts van toeleverende partijen (de gebruikte presentatie is aldaar in te zien).

Ook op het SDF startsymposium van 26-9-2013 zal het ook kort aan de orde komen.