MCTL

De wetenschap van ontwerpen bestaat in het voorkomen van de moeilijkheden in de uitvoering.

Hoofdstuk 6.2

Taakgebied Ontwerp

V1.19.1 / 01 februari 2019

Auteur: Ton van den Hoogen

Met dank aan alle bedrijven en personen die in de afgelopen jaren bewust en onbewust een bijdrage aan MCTL hebben geleverd.

Tekstredactie: TekstFontein

HOOFDSTUK 6.2. TAAKGEBIED ONTWERP

In dit taakgebied wordt voortgebouwd op de in de vorige taakgebied opgestelde requirements. Deze requirements beschrijven de behoefte aan wijzigingen, gezien *vanuit het bedrijfsproces*. In het taakgebied Ontwerp wordt de vertaling gemaakt naar de benodigde computertechnologie *op functioneel niveau.* Welke functionele aanpassingen aan de software zijn nodig? Wat moet er aan het datamodel worden aangepast? Verandert het gebruik van hardware zodanig, dat hieraan aanpassingen nodig zijn? Er wordt kortom een volledige, eenduidige en specifieke beschrijving van de wijzigingen in de systemen gemaakt.

Ontwerpen nog relevant in de huidige, dynamische wereld?

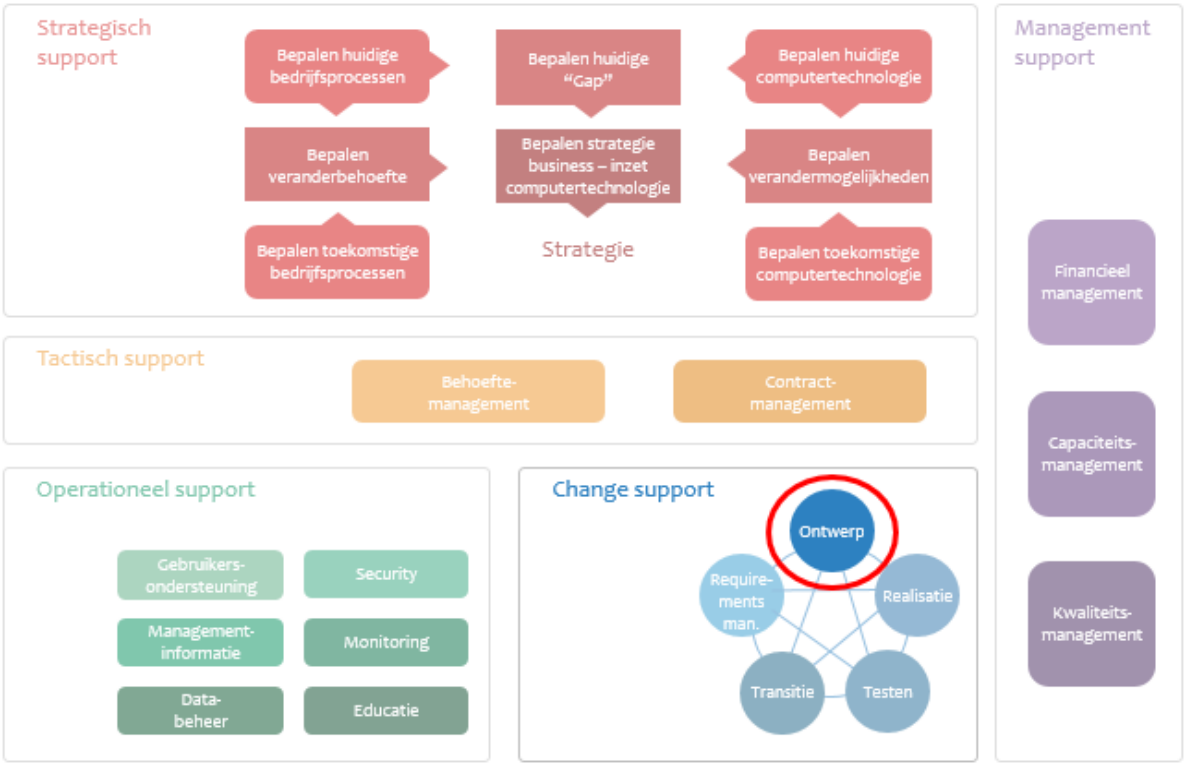
Een opmerking die nogal eens wordt gemaakt is dat ontwerpen in de huidige tijd van Agile werkwijzen niet meer relevant is. Immers, in Agile wordt juist niet alles vooraf geheel bedacht en doordacht, maar ontstaat gaandeweg de realisatie iteratief het gewenste systeem. Toch blijft ontwerpen ook in deze tijd van belang. Een goed ontwerp geeft structuur en samenhang waardoor elke requirement in de juiste context kan worden gezet. En mocht tijdens de realisatie aanpassing nodig zijn, dan kan letterlijk worden teruggegrepen op het “grote plaatje”.

Systeemontwikkeling valt onder engineering, zoals ook de bouw van huizen en bruggen dat is. In engineering geldt dat alles tweemaal wordt gemaakt: eenmaal in het hoofd en op papier (in een ontwerp), en eenmaal in het echt. Door de eerste stap over te slaan lijk je tijdwinst te behalen, maar in werkelijkheid betaal je daar uiteindelijk juist een enorme prijs voor.

Tot slot is een belangrijk onderdeel van ontwerpen ook het opdelen van een systeem in logische modules met een sterke interne samenhang. Een dergelijke opdeling levert subsystemen op die later eenvoudig onderhoudbaar en uitbreidbaar zijn. Een evolutionaire of organische werkwijze is hier vreselijk lastig en levert vaak veel rework op.

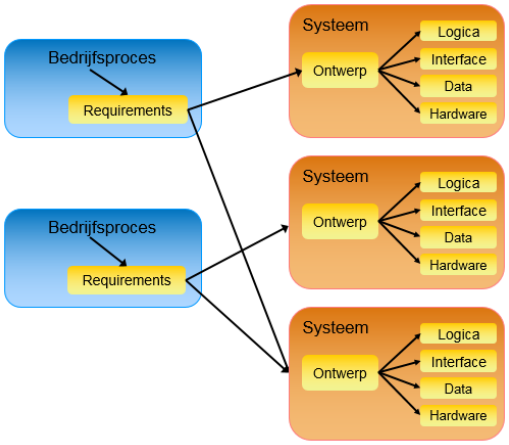
PLAATS IN HET MCTL-FRAMEWORK

Het taakgebied Ontwerp maakt deel uit van het taakcluster Change support:



ACHTERGROND

In Requirements management ligt de focus op het bedrijfsproces en de wijzigingen daarin. Bij Ontwerp ligt de focus op de onderliggende benodigde systemen. Deze matchen doorgaans niet een-op-een met het bedrijfsproces. Soms kan het handig zijn één systeem te bouwen dat meerdere bedrijfsprocessen ondersteunt (integratie), soms kan het beter zijn één bedrijfsproces te ondersteunen door meerdere, losstaande systemen. Het volgende schema illustreert deze situatie.



Data en functionaliteiten kunnen vaak op meerdere plaatsen gebruikt worden. Binnen Ontwerp wordt gestreefd naar het meervoudig gebruik van componenten. Hierdoor worden kosten verlaagd, fouten voorkomen (doordat data maar op een plaats vastgelegd worden) en wordt het onderling samenwerken in een organisatie gestimuleerd: er is immers waar mogelijk sprake van één gedeelde omgeving.

DEFINITIES

Binnen het taakgebied Ontwerp worden enige definities gebruikt. Die welke nog niet eerder ter sprake zijn gekomen, worden hieronder omschreven.

Definitie *entiteit*
Een entiteit is een op zichzelf staande eenheid, uitgedrukt in een zelfstandig naamwoord.

Voorbeelden:

- Docent
- Les
- Leslokaal

Let op! Een entiteit kan een representatie zijn van zowel een feit, een gegeven, informatie als kennis. De definities van deze entiteiten luiden als volgt.

- **Data:** De nullen en enen in een computersysteem, samen de enige vorm waarmee een digitale computer kan werken. Alle hierop volgende termen geven meer of minder betekenis aan deze nullen en enen.
- **Feiten:** werkelijke, waarneembare dingen, gebeurtenissen, omstandigheden of eigenschappen. Voorbeelden: een auto, een banksaldo, de zon schijnt in Amsterdam, het huis is tien meter hoog.
- **Gegevens:** registraties van feiten. Het getal 10 kan dan de registratie zijn van de hoogte van een huis in meters, maar ook de temperatuur in een stad op een bepaalde datum.
- **Informatie:** gegevens die betekenis hebben voor de *ontvanger* van de gegevens. Of iets informatie is hangt dus af van de ontvanger en niet van de zender. Zo zal of een trein op tijd vertrekt van een bepaald station voor slechts enkele mensen informatie zijn en de specificatie bij een factuur voor sommigen relevant (en dus informatief) en voor anderen niet.
- **Kennis:** ontstaat uit informatie samen met vaardigheden en ervaring. Deze drie elementen samen zorgen er bijvoorbeeld voor dat iemand weet hoe hij storingen aan een auto kan verhelpen.
- **Begrip:** geeft antwoord op de vraag: waarom? Begrip zorgt ervoor dat iemand weet waarom een storing op een bepaalde manier verholpen moet worden.

- **Wijsheid:** geeft houvast om precies de juiste dingen te doen. Wijsheid leidt ertoe dat iemand een auto zo gebruikt dat deze zo lang mogelijk en met zo min mogelijk storingen meegaat.

Er wordt wel onderscheid gemaakt in zwakke en sterke entiteiten: Een sterke entiteit is er een die op zichzelf kan bestaan zoals bijvoorbeeld een "docent" of een "klaslokaal". Een zwakke entiteit bestaat slechts met een referentie. Een "les", bijvoorbeeld, refereert aan een docent en een klaslokaal; immers, zonder docent of klaslokaal geen les.

Definitie *entiteitwoordenboek*
De verzameling van alle entiteiten die in de gehele organisatie gebruikt worden, inclusief hun definitie.

Definitie *attribuut*
Een attribuut is een eigenschap van een entiteit.

Voorbeelden:
- Docent: docentnummer, voornaam, achternaam, telefoonnummer, e-mailadres, datum indiensttreding.
- Les: lesnummer, naam, leslokaalnummer, docentnummer, datum, starttijd, eindtijd.
- Leslokaal: leslokaalnummer, locatie.

Definitie *relatie*
Een relatie geeft een verband tussen entiteiten aan.

Voorbeelden:
- Een docent geeft les.
- Een les vindt plaats in een bepaald leslokaal.

Definitie *constraint*
Een constraint is een beperking, regel of conditie waaraan voldaan moet worden. In de ontwerpfase worden constraints formeel vastgelegd. Bijvoorbeeld: er mag niet meer dan één les op hetzelfde moment in hetzelfde leslokaal worden gepland, of een docent mag maar op één datum/tijdsblok op één les worden ingepland.

DOEL VAN DIT TAAKGEBIED

Het doel van het taakgebied Ontwerp is het creëren van een samenhangend ontwerp van het complete conceptuele en logische datamodel, het beschrijven van alle bijbehorende functionaliteiten (logica en bijbehorende interfaces), de niet-functionele systeemeisen, de organisatie-inrichting en de rolbeschrijvingen.

HOE WEET JE DAT HET DOEL IS BEREIKT?

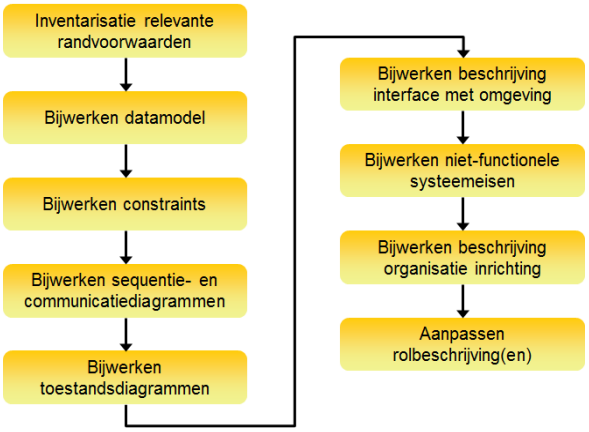
De volgende indicatoren zijn te benoemen om te weten of bovenstaand doel is bereikt.

- Het bijgewerkte ontwerp blijkt in het taakgebied Realisatie en Testen voor minstens 95% volledig (dekkend) en juist. Met andere woorden: er zijn minder dan 5% verdere aanpassingen nodig om de achterliggende doelstelling(en) van de wijziging te behalen.
- De realisatie kan op basis van het ontwerp zonder verdere informatie worden uitgevoerd (precies voldoende, volledig en gedetailleerd).
- Naast de binnen Ontwerp aangepaste elementen blijken geen andere elementen gewijzigd te moeten worden (dekkingsgraad 100%).
- Nadat de wijzigingen in productie zijn genomen, blijken zij ten opzichte van het ontwerp voor minimaal 90% volledig juist (en behoeven zij dus geen verdere aanpassing).

(Genoemde percentages zijn indicatief.)

TAKEN

De taken in dit taakgebied zijn als volgt schematisch weer te geven.



Niet altijd zal elke taak nodig zijn, er moet met dit schema dus met het nodige gezond verstand omgegaan worden. Het is wel aanbevelenswaardig alle taken elke keer na te lopen waarbij als default geldt dat elke taak wordt uitgevoerd. Er moet dus een reden zijn hier iets *niet* te doen en niet andersom.

Ook het detailleringsniveau kan een punt van discussie zijn. De stelregel is dat het oorspronkelijk (bij de nieuwbouw) gekozen detailleringsniveau in het ontwerp hier de richtlijn is. Het gaat hier immers doorgaans om aanpassingen/uitbreidingen op bestaande systemen.

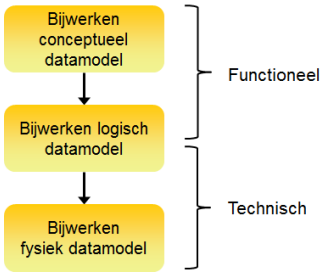
1. INVENTARISEREN RELEVANTE RANDVOORWAARDEN

In deze taak worden de relevante randvoorwaarden geïnventariseerd. Deze randvoorwaarden kunnen afkomstig zijn uit infra-, applicatiesupport of leveranciers, bijvoorbeeld vanwege een bepaalde architectuur op basis waarvan wordt gewerkt. Randvoorwaarden kunnen echter ook uit de gebruikersorganisatie afkomstig zijn.

Voorbeelden hiervan zijn randvoorwaarden in verband met juridische aspecten, vanuit ergonomisch oogpunt, vanuit beveiliging (functiescheiding) of bedrijfsstandaarden.

2. BIJWERKEN DATAMODEL

Het bijwerken van het datamodel bestaat schematisch weergeven uit de volgende stappen.



Hierna worden de activiteiten op functioneel terrein verder uitgewerkt.

1. Bijwerken conceptueel datamodel

Als eerste dient de betekenis van alle in de requirements genoemde entiteiten vast te staan. Dat lijkt een vanzelfsprekendheid, maar het voorbeeld in onderstaand kader geeft de problematiek goed weer.

> Voorbeeld: entiteit 'klant'.
>
> Elk bedrijf heeft klanten. En dus een klantenbestand. Maar wat is eigenlijk een klant?
>
> a) Is een klant een persoon die of een bedrijf dat een bestelling heeft gedaan?
> b) Of is een klant een persoon aan wie of een bedrijf waaraan werkelijk iets is geleverd?
> c) En wat dan als de levering is afgeleverd, maar teruggezonden?
> d) Of als een klant al jaren niets meer heeft besteld?
>
> Wie werkelijk als klant gezien moet worden, is nog niet meteen een uitgemaakte zaak. Stel dat de afdeling marketing een klantenactie op touw wil zetten. Moet die actie dan de 'echte', de potentiële, voormalige of de huidige klanten bereiken? Wanneer klantendata niet goed gedefinieerd en vastgelegd zijn, zijn ze niet goed bruikbaar.

Er moet een entiteitwoordenboek (ook wel 'data dictionary' genoemd) bestaan, waarin alle in de gehele organisatie gebruikte entiteiten vastgelegd zijn. Vervolgens worden alle in de requirements genoemde entiteiten hier getoetst aan de bestaande beschrijvingen. De volgende situaties zijn dan te onderscheiden.

a) Is er een match (zelfde term én zelfde betekenis), dan is geen verdere actie nodig.

b) Indien een term al bestaat, maar een andere betekenis heeft, is er meestal een aanzienlijk probleem. Eerst moet dan worden nagegaan waar de versie die nu in het entiteitwoordenboek is opgenomen gebruikt wordt en waardoor de afwijking is ontstaan. De oplossing kan bestaan uit het aanpassen van de entiteit in het entiteitwoordenboek (en dus aanpassing overal waar deze entiteit wordt gebruikt) of het conformeren van de entiteit uit de requirements aan de beschrijving ervan in het entiteitwoordenboek (ofwel het anders benoemen van de entiteit zodat een nieuwe ontstaat). Zie voor de laatstgenoemde oplossing verder onder c hierna. Een voorbeeld: 'les' zou kunnen refereren aan een les in een leslokaal met docent, maar ook aan een informatieblok in een elektronische leeromgeving. Oplossing is dan een nieuwe term te introduceren zoals 'e-les'.
c) Indien een term nog niet bestaat in het woordenboek, kan deze worden toegevoegd. De nodige terughoudendheid moet wel worden betracht. Zeker bij systemen die al wat langer bestaan is het zinvol de vraag te stellen waarom een entiteit niet al eerder opgenomen is. Het gevaar bestaat dat dezelfde entiteiten onder verschillende namen in het entiteitenwoordenboek en daarmee uiteindelijk in het systeem terechtkomen. Dat moet vanzelfsprekend worden voorkomen; synoniemen (bijvoorbeeld leraar - docent of klaslokaal - leslokaal) leiden doorgaans tot grote verwarring.

Samen met de vastlegging in het entiteitenwoordenboek moet ook het type van de entiteit worden vastgesteld en vastgelegd. Het type kan zijn:

- numeriek (alleen cijfers of eventueel met komma's of wetenschappelijke notatie);
- datum;
- datum + tijd (timestamp);
- alfabetisch (alleen letters uit alfabet);
- tekst (tekens uit alfabet, speciale tekens en cijfers);
- afbeelding;
- video;
- geluid;
- vector.

Ten derde dienen de aan de entiteit gestelde eisen te worden vastgelegd. Voorbeelden van dit soort eisen/restricties zijn:

- aantal tekens;
- limieten: waarde moet onder of boven een bepaalde waarde liggen;
- bepaalde tekens zijn bij invoer uitgesloten (zoals bijzondere tekens of tekens die niet in het Latijnse alfabet voorkomen);
- bij invoer zijn geen datums in het verleden of juist toekomstige datums toegestaan;
- de waarde van de entiteit moet binnen een bepaalde bandbreedte liggen;
- vereiste nauwkeurigheid;
- wel of niet atomair (samengestelde entiteit zoals naam of splitsen in voornaam, tussenvoegsels en achternaam);
- omvang;
- kwaliteit (zoals van geluid, video, afbeelding).

Ten vierde kan worden aangegeven hoe een bepaalde entiteit wordt gebruikt/mag worden gebruikt. De aanschafdatum van een apparaat kan eveneens de ingangsdatum van de garantietermijn zijn, bijvoorbeeld.

> Oppassen met meervoudig gebruiken van entiteiten.
>
> Er is voorzichtigheid geboden bij het meervoudig gebruiken van een entiteit. Een voorbeeld: wanneer de Belastingdienst of een autoleasebedrijf de registratiedatum van een voertuig koppelt aan de betaaltermijnen, kan dit problemen opleveren zodra het beleid rondom die termijnen wijzigt. Bijvoorbeeld als opeens betaling per jaar moet gaan plaatsvinden, of dat de betaling ingaat per 1e van de maand waarop de registratie van het voertuig plaatsvindt. Of de registratiedatum om allerlei redenen soms gaat afwijken van de werkelijke ingebruikname van de auto en dus ook de datum waarop de auto op de weg is gekomen. Met andere woorden: in dit voorbeeld is het vanaf het begin handiger om de registratiedatum los te zien van de betalingen en betaaltermijnen en dus ook als losse entiteiten te registreren.

De totale beschrijving op conceptueel niveau van alle wijzigingen in het entiteiten-woordenboek, geeft een goede basis voor het bijwerken van het logisch datamodel.

> Beschrijving entiteit voorzien van voorbeeld.
>
> Het is aanbevelenswaardig om elke entiteit te voorzien van een concreet voorbeeld. Dat maakt de beschrijving van de entiteiten helderder.

Een volledig, bijgewerkt entiteitenwoordenboek geeft een eenduidig referentiekader. Mede hierdoor worden misverstanden en foutief gebruik van systemen voorkomen.

2. Bijwerken logisch datamodel

Het logisch datamodel legt de attributen bij alle entiteiten en alle relaties tussen de entiteiten vast. Dit leidt uiteindelijk tot een bedrijfsbreed, overkoepelend datamodel. Mogelijke relaties zijn:

1. 1:1, waarbij de ene entiteit precies eenmaal verbonden is met een andere entiteit (één persoon heeft één woonadres);
2. 1:n, waarbij een entiteit verbonden kan zijn met 0 tot meerdere andere entiteiten (een persoon heeft 0 tot meerdere banen);
3. n:m, waarbij meerdere entiteiten verbonden zijn met meerdere andere entiteiten (een werknemer heeft 0 tot meerdere werkgevers, maar andersom heeft een werkgever ook 0 tot meerdere werknemers);

Tot slot nog eenmaal de definities van zwakke en sterke entiteiten.

- Een sterke entiteit is een entiteit die op zichzelf kan bestaan.
- Een zwakke entiteit bestaat slechts met een referentie.

3. Bijwerken fysiek datamodel

Het bijwerken van het fysiek datamodel is werk van DBA'ers en dus geen functionele activiteit. Vanuit functioneel perspectief wordt hier wel de uitvoering daarvan voor zover mogelijk gecontroleerd. Kent het systeem de mogelijkheid aan de functionele kant extra velden/tabellen te definiëren, dan is het natuurlijk ook mogelijk de daadwerkelijke wijzigingen in de database(s) zelf uit te voeren. Overleg met de DBA'er is aan te bevelen. Het is in dit geval vrij eenvoudig om de structuur van het datamodel te ruïneren.

Vanuit functioneel oogpunt kan er ook aandacht zijn voor de wijze waarop integriteit technisch wordt gewaarborgd en voor de mogelijkheden tot optimalisatie. Vooral een bepaalde wijze van opvragen van data (met name query's) kan leiden tot het in fysieke zin aanpassen van het logische datamodel, bijvoorbeeld door indexering, views of redundancy. Datamodellering en de manier waarop data functioneel worden gebruikt, staan niet geheel los van elkaar. Dit komt bij het ontwerp van het fysieke datamodel verder ter sprake.

3. BIJWERKEN CONSTRAINTS

Het bijwerken van constraints omvat het formeel vastleggen van de beperkingen, condities en regels. Deze kunnen worden afgeleid uit het onderdeel *Regels/restricties* in het taakgebied Requirements management.

Er zijn verschillende vormen van constraints te onderscheiden.

1. De preconditie: een constraint waaraan moet zijn voldaan voor een operatie wordt uitgevoerd.
2. De postconditie: een constraint waaraan moet zijn voldaan direct na uitvoering van een operatie.
3. Een invariant: een constraint die altijd waar moet zijn voor alle betreffende entiteiten (instanties van een klasse).
4. Een guard: een constraint op een toestandsovergang.

Een uit UML afkomstige wijze om constraints te beschrijven is OCL: Object Constraint Language. Deze geeft de mogelijkheid alle constraints eenduidig vast te leggen. Elke constraint is verbonden met een object.

Voorbeeld: klasse *bestelling*.

Bij een bestelling willen we vastleggen dat er een tafelnummer moet zijn ingevuld. Dat is dan een invariant, die als volgt kan worden uitgewerkt:

Context bestelling **inv:**
tafelNummer > 0

Hierbij is de klasse *bestelling* de naam van het contextobject, vandaar het woord Context.

Wanneer de constraints niet (allemaal) op een dergelijke wijze eenduidig vast te leggen zijn, dient er in het taakgebied Requirements management verduidelijking te worden verkregen. Doorgaans is er sprake van een iteratieve werkwijze tussen de taakgebieden Ontwerp en Requirements management.

Constraints kunnen betrekking hebben op:

- de individuele attributen (eigenschappen) van de klassen, zoals eisen aan waarden en limieten.
- relaties tussen attributen: indien de ene aan een bepaalde eis voldoet, dan geldt ook iets voor het andere attribuut (bijvoorbeeld wederzijdse uitsluiting, de ene moet een hogere waarde hebben dan de andere).

Naast condities zijn er ook andere zaken in OCL vast te leggen, zoals initiële waarden van attributen en afleidingsregels (de vluchtduur van een vliegtuig is af te leiden uit de vertrek- en aankomsttijd).

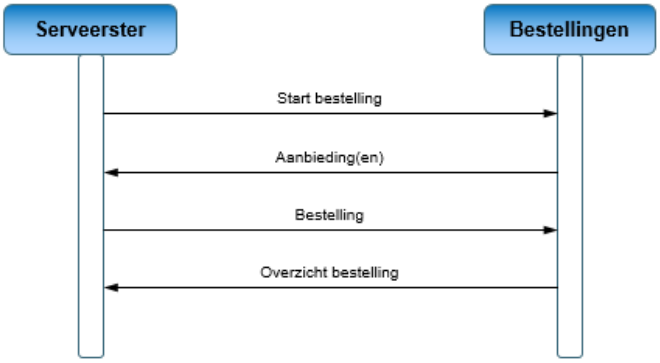
> De studentenkroeg.
>
> Een studentenkroeg kent een simpele regel: zodra het totaal van de bestellingen van een groep studenten boven de 100 euro uitkomt, moet er eerst betaald worden. Een naastgelegen kroeg hanteert een ander bedrag en een derde kroeg kent dit hele fenomeen niet. Om een computersysteem te creëren waarin alle genoemde werkwijzen eenvoudig in te passen zijn, is het werken met parameters of bedrijfsregels in de software de aangewezen weg. Daarmee wordt de software instelbaar op basis van de wensen per individuele kroeg. Bovendien wordt het systeem daardoor flexibel. Stel dat een kroeg in de loop van de tijd een ander bedrag wil hanteren of ook bij evenementen/particuliere feesten zo wil gaan werken, is dat mogelijk zonder dat er softwareaanpassingen nodig zijn.

Voor een verdere beschrijving van de mogelijkheden van OCL verwijzen wij u naar de documentatie op dit gebied die buiten MCTL al beschikbaar is.

4. BIJWERKEN SEQUENTIE- EN COMMUNICATIEDIAGRAMMEN

De requirements worden hier verder uitgewerkt, voornamelijk in sequentiediagrammen. Daarin worden gedetailleerd en in chronologische volgorde de interacties tussen de verschillende actoren beschreven.

Een voorbeeld van een dergelijk diagram:



Een sequentiediagram geeft een grafische weergave van de volgorde van gebeurtenissen. De tijd loopt in het schema van boven naar beneden. Ter verduidelijking van het verloop van de gebeurtenissen kan een diagram worden gebruikt.

Communicatiediagrammen leggen de nadruk op de communicatie tussen de verschillende objecten van een systeem en visualiseren welke verbanden er tussen die objecten bestaan. Communicatie hoeft niet lineair in de tijd te verlopen en dat is in dit type diagrammen wat eenvoudiger vorm te geven.

In de praktijk worden sequentiediagrammen veel meer gebruikt dan communicatie-diagrammen. Beide uitwerken leidt tot overkill en daarom zijn sequentiediagrammen de meest voorkomende.

Controleer tot slot de dekking: zijn alle in de diagrammen genoemde activiteiten terug te vinden in de beschreven requirements?

5. BIJWERKEN TOESTANDSDIAGRAMMEN

Het kan nuttig zijn voor objecten, voor deelsystemen of zelfs voor een systeem als geheel toestandsdiagrammen te maken. In een toestandsdiagram worden alle mogelijke toestanden van een object, een deelsysteem of systeem beschreven. Het is afhankelijk van de gebruikte methode op welk niveau toestandsdiagrammen worden beschreven. Wordt een methode als RUP gebruikt (met als notatiewijze UML) dan worden per keer alleen toestanden van objecten van één klasse beschreven. Zeker in real-timesystemen kan het zinvol zijn ook op deelsysteem- en systeemniveau toestandsdiagrammen te maken.

Nu volgt een voorbeeld van een eenvoudig toestandsdiagram van een bestelling op een terras.

Behalve de toestanden waarin iets zich kan bevinden, moeten hier ook de transities (toestandsovergangen) worden beschreven. Een transitie zorgt voor overgang van de ene naar de andere toestand. Naast de begin- en eindtoestand zijn er nog twee bijzondere toestanden te onderkennen.

6. BIJWERKEN BESCHRIJVING INTERFACING MET OMGEVING

Centraal in deze taak staat het bijwerken van de beschrijving van de vorm waarin het systeem interacteert met de omgeving. Hierbij kan worden gedacht aan menuschermen, printwerk, koppeling met randapparatuur en aansturing van apparaten. Dit gedeelte omvat dus niet alleen wat wel 'Human Computer Interaction' wordt genoemd: de interface tussen computertechnologie en mens. Ook interfaces met andere apparaten worden hier beschreven. Dit laatste wordt overigens steeds interessanter en omvangrijker. Computertechnologie wordt steeds meer direct aangesloten op andere apparaten om iets te meten, besturen, beïnvloeden etc. Voorbeelden hiervan zijn meet- en regelsystemen en end-to-end geautomatiseerde systemen. Ook het Internet of Things, waarbij talloze apparaten via internet aan computers worden gekoppeld, hoort in deze categorie thuis.

Aldus zijn interfaces te verdelen in:

1. mens <> computersysteem <> mens (bij in- en uitvoer is een mens betrokken).
2. mens <> computersysteem <> apparaat (bij invoer is en mens betrokken, bij uitvoer een apparaat, zoals een verfmengmachine).
3. apparaat <> computersysteem <> mens (bij invoer is een apparaat betrokken, bij uitvoer een mens, bijvoorbeeld een arts die de behandeling in gang zet).
4. computersysteem <> computersysteem (computersystemen die van elkaar gebruik maken, zoals een computersysteem dat een ander raadpleegt i.v.m. een controle).
5. apparaat <> computersysteem <> apparaat (bij invoer is bijvoorbeeld een sensor betrokken, waarna metingen in een computersysteem worden geregistreerd, waarop een apparaat anders gaat functioneren, zoals bij luchtbehandeling in een gebouw, afhankelijk van het gebruik van de ruimten).

De beschrijving van de interfaces geeft meteen de begrenzing van het systeem ten opzichte van de omgeving aan. Bij een juiste beschrijving van alle interfaces kan de interne werking van het systeem voor die omgeving verborgen gehouden worden (black-box-benadering). Hoewel in de praktijk een echte black box benadering meestal niet voor 100% haalbaar is, is een goede beschrijving van alle interfaces een belangrijk ingrediënt.

Het ontwerpen van interfaces tussen mensen en computers is inmiddels een discipline op zichzelf geworden. Buiten MCTL is hier heel veel over geschreven. Onderstaande poster belicht een belangrijk functioneel aspect van userinterfaces.

(Bron: behappy.me)

Elke userinterface die niet voor zich spreekt, is functioneel voor verbetering vatbaar.

7. BIJWERKEN NIET-FUNCTIONELE SYSTEEMEISEN

Aan systemen moeten naast functionele ook niet-functionele eisen worden gesteld. Die laatste worden hier omschreven. Meestal worden op bedrijfsniveau globale eisen vastgesteld/-gelegd. Indien die vastlegging/-stelling gebeurd is, hoeven hier alleen maar de systeem-specifieke eisen die afwijken van de globale eisen te worden beschreven. Schematisch weergegeven ziet de opbouw er dan als volgt uit.

De niet-functionele systeemeisen vallen uiteen in de volgende vijf.

1. Beschikbaarheidseisen.
   Een systeem moet een bepaalde beschikbaarheid binnen bepaalde openingstijden hebben. Steeds vaker zijn de openingstijden 24/7. Vanwege onderhoud en storingen is binnen de openingstijden een beschikbaarheid van 100% echter onhaalbaar. Het is wel mogelijk de 100% dicht te benaderen. Zo zijn er genoeg systemen met een (werkelijke) beschikbaarheid van 99,999% (5 x 9). Maar hoe hoger het beschikbaarheidspercentage is, hoe hoger de kosten en hoe groter de consequenties zijn. Zo is een hoog beschikbaarheidspercentage alleen haalbaar na zeer grondig en uitgebreid testen en een heel zorgvuldig voorbereide inproductiename. Een hoog beschikbaarheidspercentage kan bovendien ten koste gaan van bijvoorbeeld de aanpassingssnelheid en flexibiliteit van een systeem. Hier dient dus een bedrijfsmatige afweging te worden gemaakt, waarbij vanuit infra-, applicatiesupport en leveranciers de mogelijkheden, consequenties en kosten op technisch gebied kunnen worden aangereikt. Aan de andere kant zijn in de gebruikersorganisatie de opbrengsten te calculeren in de vorm van minder productieverlies, minder kans op klantverlies (denk bijvoorbeeld aan een webshop), hoger werkgenot en meer vertrouwen in de technologie.

2. Performance-eisen.
   Performance ligt in het verlengde van beschikbaarheid, omdat bij een te lage performance het systeem niet bruikbaar (beschikbaar) is. Er dient gespecificeerd te zijn welke responsetijden gemiddeld en bij piekbelastingen moeten worden gehaald. Ook moet worden omschreven hoe het systeem moet reageren bij piekbelastingen. Dan kan bijvoorbeeld gekozen worden voor limitering van het aantal gebruikers (‘Het is erg druk, probeert u het later nog eens.’). Limitering wordt vooral ingezet wanneer het aantal gebruikers moeilijk voorspelbaar is, bijvoorbeeld op internet. Een bepaalde traagheid (slechte responsetijden) kan in bepaalde omstandigheden worden geaccepteerd, maar dat moet dan wel worden gespecificeerd.

3. Capaciteitseisen.
   De benodigde hoeveelheid (systeem)capaciteit moet worden berekend, vanzelfsprekend vanuit gebruikersperspectief. Er mag hier niet in termen als *giga*- en *terabytes* gedacht worden. Het bedrijfsproces en de wijzigingen daarin moeten altijd het uitgangspunt zijn. Door te kijken naar de aantallen (simultane) gebruikers, het aantal transacties per tijdsperiode en de gemiddelde load die een transactie op het systeem veroorzaakt, kunnen infra-, applicatiesupport en/of leveranciers berekenen welke technische middelen nodig zijn. Daarnaast moet worden meegerekend dat bepaalde data langere tijd bewaard moeten blijven. Aldus is goed te berekenen hoeveel data initieel in een systeem aanwezig zullen zijn, hoe het verloop in een opbouwperiode zal zijn en hoe een en ander zich uiteindelijk zal stabiliseren.

4. Continuïteitseisen.
   Continuïteit ligt in het verlengde van beschikbaarheid, maar wordt in het algemeen gezien als het weer beschikbaar maken van systemen na een grootschalige

verstoring. Bij grootschalige verstoringen, waarbij (een groot deel van) de gebruikersorganisatie niet kan werken als gevolg van het uitvallen van alle systemen of een bedrijfskritisch systeem, komen andere aspecten in beeld dan de hierboven onder *Beschikbaarheidseisen* genoemde. Bij continuïteitseisen wordt gedefinieerd wat de maximale downtime is per systeem en/of voor het geheel, wat de onderlinge prioriteiten zijn (welk systeem wordt als eerste weer in de lucht gebracht), de mogelijkheden tot uitwijk op andere locaties, mogelijkheden voor aanpassingen van bedrijfsprocessen (het werk op een andere manier voortzetten, minimaliseren tot de kernactiviteiten, beperking aantallen gebruikers etc.) en het maximale dataverlies dat de organisatie zich kan veroorloven. Er wordt hier een bedrijfsmatige afweging gemaakt. De kosten en gevolgen van technische en organisatorische maatregelen om continuïteit te waarborgen (calamiteiten voorkomen/gevolgen van calamiteiten beperken) moeten opwegen tegen de te verwachten opbrengsten.

5. Security-eisen.
   Op het gebied van security kunnen eisen bestaan zoals encryptie. Ook kan hier worden nagedacht over de gelaagdheid van security. In het algemeen is het niet verstandig slechts één laag van beveiliging te ontwerpen. Door security in meer lagen onder te brengen wordt het geheel veiliger. Daarnaast moet worden bepaald in welke lagen beveiliging moet worden gerealiseerd. Voorheen werd beveiliging in de hardwarelaag ondergebracht. Er waren bijvoorbeeld geen koppelingen tussen een intern bedrijfsnetwerk en de buitenwereld. Doordat steeds meer hardware aan elkaar verbonden is, wordt dat steeds minder een optie. Daardoor is het gros van de beveiliging tegenwoordig in de softwarelaag aangebracht. Het is ook mogelijk de beveiliging op dataniveau te implementeren. Bij het benaderen van data, op welke manier dan ook, wordt dan eerst gecheckt of het wel om een geautoriseerde aanvraag gaat.
   Security wordt op basis van de requirements en algemene security-eisen in het ontwerp meegenomen en uiteindelijk in Realisatie verwezenlijkt.

Hiermee zijn alle niet-functionele systeemeisen benoemd.

8. BIJWERKEN BESCHRIJVING ORGANISATIE-INRICHTING

Behalve de functionele en niet-functionele aspecten van het systeem moet er aandacht zijn voor de benodigde organisatorische aanpassingen. Deze organisatorische aanpassingen bestaan uit de volgende twee componenten.

1. Aanpassingen in de organisatie van de gebruikersorganisatie.
   Functionele aanpassingen aan systemen leiden soms tot aanpassingen in de organisatie of zijn het gevolg van een organisatorische aanpassing. Deze organisatorische aanpassingen worden nu verder uitgewerkt. Er kan sprake zijn van een andere invulling van een taak, de verschuiving van taken en het samenvoegen van meerdere afdelingen.

2. Aanpassingen in de organisatie van functioneel support.
   Naast organisatorische consequenties voor de gebruikersorganisatie kunnen er consequenties zijn voor functioneel support. Dat betekent concreet dat functioneel specialisten naar zichzelf en hun eigen werk moeten kijken: gaat daar iets veranderen? Ook de werkverdeling binnen infra- en applicatiesupport of de afspraken met leveranciers kunnen veranderen. Het is zelfs mogelijk dat de organisatorische afspraken in de met leveranciers afgesloten contracten moeten worden aangepast. In dat geval worden in taakgebied Ontwerp de inhoudelijke aspecten uitgewerkt en vervolgens in het taakgebied Contractmanagement de benodigde contractaanpassingen uitgevoerd.

9. AANPASSEN ROLBESCHRIJVING(EN)

Uit de functionele en organisatorische aanpassingen is af te leiden of er op het gebied van beveiliging wijzigingen nodig zijn. Met name is dan te denken aan de aanpassing van rollen in de gebruikersorganisatie. Veranderen taken of de workflow, dan is het mogelijk dat de rolbeschrijvingen niet meer voldoen. In uitzonderlijke gevallen kan het zijn dat een nieuwe rolbeschrijving moet worden gemaakt of dat een rol komt te vervallen. Uiteindelijk leiden veranderingen in rollen tot andere menustructuren en autorisaties in het systeem. Soms worden hier wijzigingen aangebracht omdat er nieuwe technologische mogelijkheden zijn ontstaan die in het betreffende bedrijfsproces gebruikt gaan worden.

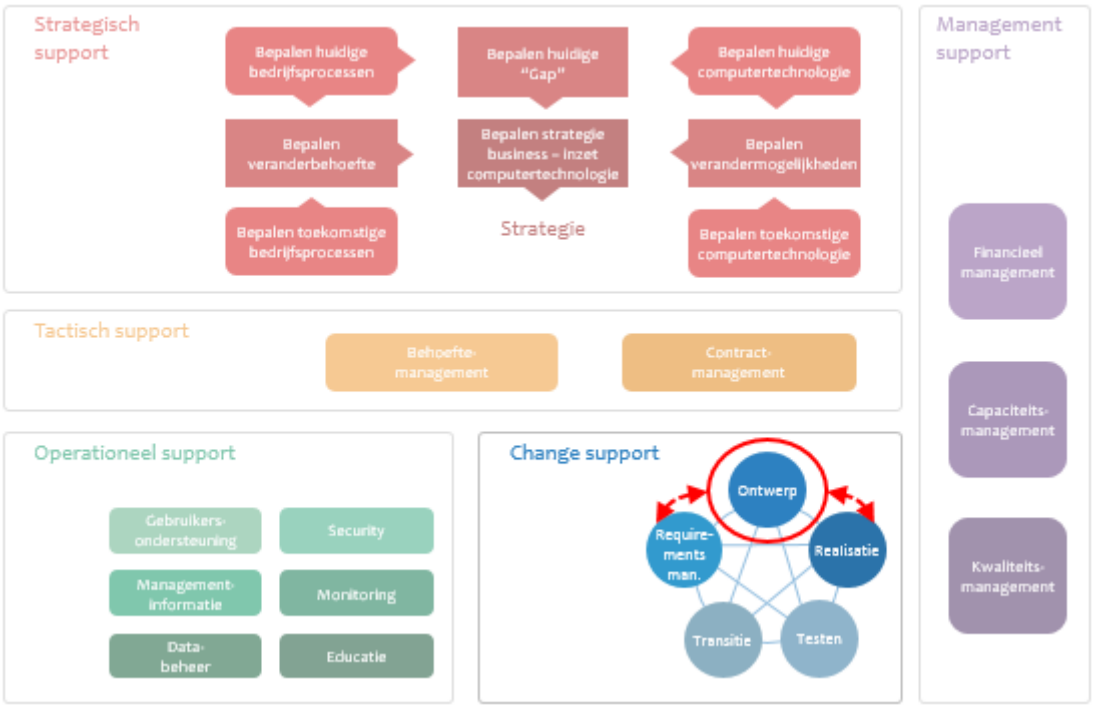
Vastlegging van welke rol wat mag, vindt plaats in een autorisatiematrix. Daarin worden op de ene as de rollen weergegeven en op de andere de functionaliteiten en data(groepen). Vervolgens wordt aangegeven welke rol welke functionaliteiten mag gebruiken. Bij data moet gedetailleerd worden aangegeven of data mogen worden ingezien, gebruikt, gewijzigd en/of verwijderd. Ook het eigenaarschap kan bij zowel data als functionaliteiten worden aangegeven. Het eigenaarschap hangt doorgaans nauw samen met de bevoegdheid functionaliteiten en/of data te (laten) aanpassen.

Functiescheiding is een belangrijk organisatorisch aandachtspunt. Functiescheiding hoort op organisatieniveau bepaald en beschreven te zijn. Deze moet bij wijzigingen in rollen en autorisaties aantoonbaar in stand blijven. Er zijn overigens veel verschillende uitgangspunten voor functiescheiding te vinden. Het ene bedrijf vindt dat iemand alleen mag zien/gebruiken wat nodig is voor de juiste uitvoering van het werk, terwijl andere bedrijven een tegenovergesteld uitgangspunt hanteren: alles is voor iedereen beschikbaar tenzij uitdrukkelijk niet toegestaan. Binnen Ontwerp worden deze uitgangspunten vertaald naar concrete scheidingen in het bedrijfsproces en het bijbehorende systeem.

Bij de uitvoering van deze taak wordt vaak de Security Officer betrokken. Hij/zij is immers degene die overall verantwoordelijk is voor de gehele beveiliging van alle gebruikte systemen.

RELATIES MET ANDERE ONDERDELEN VAN MCTL

Dit taakgebied kent de volgende belangrijke relaties.



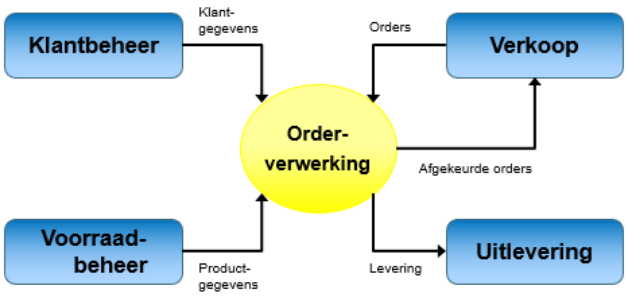
De belangrijkste relaties zijn die met taakgebied **Requirements management** en **Realisatie**. Het eerstgenoemde zal de uitgewerkte requirements aanleveren op basis waarvan het ontwerp kan worden bijgewerkt. Vervolgens zal dit bijgewerkte ontwerp aan Realisatie worden opgeleverd. Een andere duidelijke relatie is die met taakgebied Testen omdat daar (onder andere) zal worden beoordeeld of het gerealiseerde conform het ontwerp is.

OPMERKINGEN

De volgende opmerkingen zijn te maken betreffende dit taakgebied.

1. ANDERE ONTWERPTECHNIEKEN

Een voorbeeld van een andere specificatietechniek is SCD (Systeemcontextdiagram). Deze is afkomstig uit de IT en functioneel vaak minder bruikbaar. Bij deze techniek wordt het systeem in het midden gezet en de systemen of domeinen waarmee directe interactie is eromheen gepositioneerd, zoals in het volgende schema is te zien:



Deze specificatietechniek wordt in MCTL niet gebruikt.

2. AARD VAN DATA

Veel data zijn redelijk statisch: ze worden betrekkelijk veel gebruikt, maar slechts incidenteel geüpdatet. Indien wordt gezocht naar (verdere) optimalisatie van bedrijfsprocessen en systemen kan dit zeker een rol spelen. Bedrijfsbreed kunnen bijvoorbeeld veel data worden vrijgegeven voor gebruik, terwijl het muteren wordt geconcentreerd op één plaats in één systeem. Aldus ontstaat al snel een eenvoudiger beheer van datasets met tegelijkertijd een zo breed mogelijk gebruik ervan.

3. HERGEBRUIK

Een belangrijk thema in het taakgebied Ontwerp is hergebruik. Met hergebruik wordt bedoeld dat de datastructuur, maar ook functionaliteiten en interfaces zo moeten zijn ontworpen dat deze op zoveel mogelijk plaatsen gebruikt en hergebruikt kunnen worden. Daarmee zijn kosten te besparen en wordt de eenduidigheid van systemen aanmerkelijk verhoogd. Wanneer daarnaast bijvoorbeeld een 'Windows look & feel' gehanteerd wordt, gaan userinterfaces er min of meer hetzelfde uitzien. Vanuit functionele optiek zijn beide argumenten interessant. Het nadeel van ontwerpen op basis van hergebruik is vaak wel dat de kosten in eerste instantie hoger kunnen zijn en de complexiteit van systemen groter wordt. Toch blijft hergebruik vanuit functioneel oogpunt een fraai streven.

> Autobouw
>
> Hergebruik en standaardisatie wordt ook bij de bouw van auto's op grote schaal toegepast. Zo zijn het chassis, maar ook grote delen van motoren en het interieur van veel auto's identiek. Probleem is natuurlijk wel dat auto's daardoor steeds meer op elkaar gaan lijken.

4. MODULAIRE OPBOUW VAN SYSTEMEN

Naast hergebruik is een ander belangrijk thema binnen het taakgebied Ontwerp de modulaire opbouw van systemen. Een modulaire opbouw geeft de mogelijkheid onderdelen

van systemen flexibel te combineren. Die flexibiliteit is van functioneel belang bij de inzet in bedrijfsprocessen die zelf flexibel moeten zijn. Hoewel de essentie van een modulaire opbouw vrij eenvoudig is, kan het in het taakgebied Ontwerp nog lastig zijn de juiste opdeling vast te stellen. De volgende drie vragen kunnen hierbij gesteld worden.

1. Welke onderdelen hebben onderling de sterkste samenhang en horen dus in één module thuis?
2. Op welk niveau willen we modules creëren ofwel wat is de gewenste omvang van modules? (Vaak wordt gedacht aan modulaire opbouw à la LEGO, maar dat geeft (te) veel losse modules, wat het gevaar van extra complexiteit met zich meebrengt. Werken met iets grotere 'bouwblokken' kan beter hanteerbare modules opleveren.)
3. Worden modules samengesteld op technische gronden (platforms, programmeertalen e.d.) of op functionele gronden (samenhang gerelateerd aan bedrijfsprocessen en -taken)?

CERTIFICERING/PROEFEXAMENVRAGEN

Voor MCTL kunt u zich certificeren op *foundation*, *advanced* en *expert* niveau. Het foundationniveau toetst uw kennis van MCTL. Het advanced en expert level toetsen uw vaardigheid in het toepassen van MCTL. In een apart onderdeel, 'MCTL Certificering', vindt u alle informatie over de drie niveaus. Hierna vindt u proefexamenvragen op foundationniveau. Aansluitend treft u een aantal vragen aan op advanced–basisniveau.

1. MCTL FOUNDATION - PROEFEXAMENVRAGEN

Voor dit hoofdstuk zijn de volgende proefexamenvragen beschikbaar. Maak deze zonder terug te bladeren. De correcte antwoorden en uitleg vindt u direct hierna.

14-1. Binnen het taakgebied Ontwerp is er ook aandacht voor de niet-functionele systeemeisen. Deze wordt in MCTL onderverdeeld in:
a. beschikbaarheid, capaciteit en continuïteit;
b. beschikbaarheid, performance, capaciteit en continuïteit;
c. beschikbaarheid, performance, capaciteit, continuïteit en security;
d. beschikbaarheid, performance, capaciteit, continuïteit, security en userinterface.

14-2. Binnen het taakgebied Ontwerp worden ook op datagebied verschillende modellen bijgewerkt. De volgende bevinden zich op het functionele terrein.
a. Het conceptueel en logisch datamodel.
b. Het conceptueel en fysiek datamodel.
c. Het logisch en fysiek datamodel.
d. Het logisch datamodel en het implementatiemodel.

14-3. In het taakgebied Ontwerp wordt in het conceptueel datamodel het entiteit-woordenboek bijgewerkt. Voorbeelden van een entiteit zijn *klant*, *leerling* en

***factuur*. Waarom is het zo belangrijk een eenduidige beschrijving van alle entiteiten te hebben?**
a. Het geeft iedereen op het functionele terrein hetzelfde referentiekader.
b. Het wordt geëist door de softwareleverancier.
c. Het is van belang voor een goede onderlinge communicatie.
d. Een dergelijk woordenboek kan ook worden gebruikt om het systeem te vertalen, zodat het ook in andere landen kan worden gebruikt.

14-4. In het taakgebied Ontwerp wordt ook aandacht besteed aan het bijwerken van de beschrijving van de interfacing. Dit betreft:
a. de interface tussen het computersysteem en de gebruikers ervan;
b. de interface tussen het computersysteem en de randapparatuur;
c. de interface tussen computersystemen onderling;
d. de interface tussen computersystemen onderling, computersystemen met mensen en computersystemen met randapparatuur.

14-5. Binnen het taakgebied Ontwerp is ook aandacht voor beveiliging. Dit uit zich onder andere in:
a. aanpassingen in rolbeschrijvingen en de autorisatiematrix;
b. de uitgifte van nieuwe accounts;
c. het uitwerken van een functioneel ontwerp van een firewall;
d. de discussie over het al dan niet mogen gebruiken van USB-sticks voor gevoelige informatie.

2. MCTL FOUNDATION – PROEFEXAMENVRAGEN MET ANTWOORDEN EN UITLEG

Hierna vindt u de proefexamenvragen met direct daarachter de antwoorden en uitleg.

14-1. Binnen het taakgebied Ontwerp is er ook aandacht voor de niet-functionele systeemeisen. Deze wordt in MCTL onderverdeeld in:
a. beschikbaarheid, capaciteit en continuïteit;
b. beschikbaarheid, performance, capaciteit en continuïteit;
c. beschikbaarheid, performance, capaciteit, continuïteit en security;
d. beschikbaarheid, performance, capaciteit, continuïteit, security en userinterface.

a. Onjuist. Performance en security horen er ook bij.
b. Onjuist. Security hoort er ook bij.
c. Juist. Dit zijn precies de vijf onderdelen die binnen MCTL worden benoemd bij de niet-functionele systeemeisen. Zie hoofdstuk 14.
d. Onjuist. Userinterface-ontwerp hoort bij de functionele systeemeisen.

14-2. Binnen het taakgebied Ontwerp worden ook op datagebied verschillende modellen bijgewerkt. De volgende bevinden zich op het functionele terrein.

a. Het conceptueel en logisch datamodel.
b. Het conceptueel en fysiek datamodel.
c. Het logisch en fysiek datamodel.
d. Het logisch datamodel en het implementatiemodel.

a. Juist. Het conceptueel en logisch datamodel worden in het taakgebied Ontwerp bijgewerkt. Zie hoofdstuk 14.
b. Onjuist. Het fysiek datamodel wordt door DBA'ers bijgewerkt, en die rol/functie valt niet onder MCTL.
c. Onjuist. Het fysiek datamodel wordt door DBA'ers bijgewerkt, en die rol/functie valt niet onder MCTL.
d. Onjuist. Het implementatiemodel is geen term die in MCTL wordt gebruikt.

14-3. In het taakgebied Ontwerp wordt in het conceptueel datamodel het entiteitwoordenboek bijgewerkt. Voorbeelden van een entiteit zijn *klant*, *leerling* en *factuur*. Waarom is het zo belangrijk een eenduidige beschrijving van alle entiteiten te hebben?
a. Het geeft iedereen op het functionele terrein hetzelfde referentiekader.
b. Het wordt geëist door de softwareleverancier.
c. Het is van belang voor een goede onderlinge communicatie.
d. Een dergelijk woordenboek kan ook worden gebruikt om het systeem te vertalen, zodat het ook in andere landen kan worden gebruikt.

a. Juist. Een eenduidige beschrijving van alle gebruikte entiteiten geeft iedereen hetzelfde referentiekader. Daardoor ontstaan dus geen misverstanden. Verkeerd gebruik van systemen wordt mede hierdoor voorkomen. Zie hoofdstuk 14.
b. Onjuist. Het kan zijn dat het (ook) door een softwareleverancier wordt geëist, maar dat is binnen MCTL geen doel.
c. Onjuist. Het entiteitenwoordenboek geeft een eenduidig referentiekader, en de communicatie wordt daar zeker ook eenvoudiger door. Het is echter niet het eerste doel.
d. Onjuist. Het zou in bijzondere gevallen daarvoor kunnen worden gebruikt. Het is binnen MCTL niet als doel geïdentificeerd.

14-4. In het taakgebied Ontwerp wordt ook aandacht besteed aan het bijwerken van de beschrijving van de interfacing. Dit betreft:
a. de interface tussen het computersysteem en de gebruikers ervan;
b. de interface tussen het computersysteem en de randapparatuur;
c. de interface tussen computersystemen onderling;
d. de interface tussen computersystemen onderling, computersystemen met mensen en computersystemen met randapparatuur.

a. Onjuist. Dit is een onderdeel, antwoord d is volledi ger.
b. Onjuist. Dit is een onderdeel, antwoord d is volledi ger.
c. Onjuist. Dit is een onderdeel, antwoord d is volledi ger.

d. Juist. Dit dekt de gehele activiteit van het bijwerken van de interfacing af. Zie hoofdstuk 14.

14-5. Binnen het taakgebied Ontwerp is ook aandacht voor beveiliging. Dit uit zich onder andere in:
a. aanpassingen in rolbeschrijvingen en de autorisatiematrix;
b. de uitgifte van nieuwe accounts;
c. het uitwerken van een functioneel ontwerp van een firewall;
d. de discussie over het al dan niet mogen gebruiken van USB-sticks voor gevoelige informatie.

a. Juist. Dit zijn inderdaad ontwerpaspecten die binnen het taakgebied Ontwerp ter sprake komen. Zie hoofdstuk 14.
b. Onjuist. De uitgifte van nieuwe accounts is een operationele taak binnen taakgebied Security.
c. Onjuist. Een firewall is een technische oplossing voor een functionele wens. Een firewall wordt doorgaans door een externe leverancier ontworpen. Het functioneel ontwerp ligt daarmee ook bij die externe leverancier.
d. Onjuist. Er moet ook aan de functionele kant worden nagedacht over het gebruik van opslagmedia. Dit is echter geen systeemontwerpkwestie maar een organisatiekwestie.

3. MCTL ADVANCED-BASIS - PROEFEXAMENVRAGEN

Voor dit hoofdstuk zijn de volgende proefexamenvragen op advanced-basisniveau beschikbaar. Het zijn open vragen waarop u de antwoorden in de tekst van dit hoofdstuk kunt terugvinden. Om veel herhaling te voorkomen is daarom hier geen aparte uitleg per vraag opgenomen.

Vraag 1 (5 punten): Welke datamodellen worden op functioneel gebied beheerd en eventueel bijgewerkt in het taakgebied Ontwerp? Let op: foutieve namen leiden bij deze vraag tot puntenaftrek.

Vraag 2 (5 punten): In het taakgebied Ontwerp wordt ook aandacht besteed aan de beschrijving van de interfaces van het systeem met de omgeving. Hoe wordt de interface tussen computer en mens ook wel genoemd? Geef een voorbeeld uit de eigen werksituatie.

Vraag 3 (5 punten): Een term die binnen taakgebied Ontwerp bij de beschrijving van interfaces wordt genoemd is *Internet of Things*. Wat voor soort interface is dit in essentie?

Vraag 4 (5 punten): Benoem alle vijf in taakgebied Ontwerp uitgewerkte, niet-functionele systeemeisen. Licht elk van de vijf in maximaal 15 woorden per eis toe.

Vraag 5 (5 punten): In zowel taakgebied Ontwerp als Security wordt gesproken over rollen en rolbeschrijvingen. Geef in maximaal 30 woorden aan wat het verschil is tussen deze twee taakgebieden op het vlak van rollen en rolbeschrijvingen.

NUTTIGE WEBSITES EN BOEKEN

Vanuit functioneel perspectief zijn de volgende websites interessant voor taakgebied Ontwerp.

- www.mctl.nl
  MCTL.nl – Website met alle informatie over MCTL; de achtergrond, een beschrijving van het model, video's, artikelen, etc. etc. Alle documenten, waaronder dit document, zijn vanaf deze website te downloaden.
- www.bisl.nl
  BiSL.nl – Website met alle informatie over BiSL. BiSL is, als voorganger van MCTL, interessant vanwege de verzameling Best Practices, whitepapers en artikelen die op deze website zijn te vinden.
- www.ismportal.nl/nl/fsm-procesmodel
  FSM – Website met alle informatie over FSM. FSM is een compacte out-of-the-box versie van BiSL. De praktische vertaling in dit model is absoluut de moeite waard.

Vanuit functioneel perspectief zijn de volgende boeken interessant voor taakgebied Ontwerp.

- Chonoles, M. J. & Schardt, J. A. (2003). *UML 2 for dummies*. New Jersey: John Wiley & Sons.
- Hoogendoorn, S. (2004). *Pragmatisch modelleren met UML 2.0.* Amsterdam: Pearson.
- Warmer. J. & Kleppe, A. (2011). *Praktisch UML*. Amsterdam: Pearson.
- Langendoen, J.W.M. (1990). *De praktijk van datamodellering*. Deventer: Kluwer Bedrijfswetenschappen.
- Coenen, A. Hermans, L., Roosmalen, M. van & Spreeuwenberg, S. (2008). *Uw bedrijf geregeld met Business Rule Management*. Den Haag: Sdu Uitgevers.
- Ross. R.G. (2003). *Principles of the Business Rule Approach*. Boston: Pearson Education Inc.
- Haas, M. de, Noordzij, M. (2002). *XML integratie en standaardisatie.* Den Haag: ten Hagen & Stam Uitgevers.