# Leidraad VISI-systematiek versie 1.3

## Bijlage 5
## Logboek van wijzigingen en 'Frequently Asked Questions'

## Informatief

Documentversie: 1.0
Datum: 25 november 2011
Status: Definitief

# Inhoud

1 Logboek van wijzigingen in de VISI-systematiek ..... 5

*1.1 Wijzigingen in versie 1.3 t.o.v. 1.2* ..... 5
1.1.1 Vergroten bijlage capaciteit VISI-berichten ..... 5
1.1.2 Verplicht stellen onderwerpveld ..... 5
1.1.3 Uniek volgnummer voor iedere transactie ..... 5
1.1.4 Machtiging om namens een andere gebruiker berichten te versturen ..... 7
1.1.5 Traceerbaarheid van communicatie waarborgen na aanpassen raamwerk ..... 8
1.1.6 Appendixtypes koppelen aan transacties ..... 11
1.1.7 Standaard en richtlijn voor het archiveren van VISI-projecten ..... 11
1.1.8 Metaraamwerk ..... 11
1.1.9 Oneindige loops/successor ..... 11
1.1.10 Aanvullende functionele eisen ..... 12
1.1.11 Richtlijn Element Conditions ..... 12
1.1.12 Dubbele messagetypes ..... 12
1.1.13 Boolean waarden ..... 12
1.1.14 Optionele velden ..... 12

*1.2 Wijzigingen in versie 1.2 t.o.v. eerdere versies* ..... 13
1.2.1 Richtlijn voor transactiekoppeling; communicatie over meerdere transacties ..... 13
1.2.1.1 Inleiding ..... 13
1.2.1.2 Deel 1: Communiceren met meerdere rollen over één onderwerp ..... 13
1.2.1.3 Deel 2: Meerdere instanties van een secundaire transactie ..... 21
1.2.2 Hergebruik en blokkeren gegevenselementen ..... 26
1.2.2.2 Waarden en aanpasbaarheid van SimpleElements en ComplexElements ..... 26
1.2.3 DateTime format ..... 33
1.2.4 Successor (zie ook 1.1.9 oneindige loops) ..... 33
1.2.5 Toevoeging MITT aan bericht ..... 33

2 Frequently Asked Questions ..... 37

*1. Wat is het eerste MessageType in een TransactionType?* ..... 37
*2. Wat is het laatste MessageType in een TransactionType?* ..... 37

*3. Indien een bericht verstuurd is in een transacties, is het dan toegestaan om nog een bericht te versturen?* ..... 37

*4. Hoe wordt met het DATETIME datatype omgegaan op het moment dat geen tijdzone is gespecificeerd?* ..... 37

*5. Als ik een raamwerk heb gepromoot en een berichtenschema gegenereerd, heb ik dan de garantie dat dit raamwerk in de praktijk werkt?* ..... 38

*6. Als ik transacties aan elkaar relateer is de traceerbaarheid van communicatie altijd gegarandeerd?* ..... 38

*7. Hoe geeft ik in een raamwerk aan of een bericht een startbericht is, of niet?* ..... 39

# 1 Logboek van wijzigingen in de VISI-systematiek

De VISI-systematiek evolueert nog steeds. Voortschrijdend inzicht, technische foutjes, wensen van gebruikers of softwareleveranciers, kunnen allemaal leiden tot verbetervoorstellen. De Technische Commissie (VISI TC) behandelt alle verbetervoorstellen en houdt in een voortschrijdend status-overzicht bij welke verbeteringen in welke versie van de VISI systematiek worden doorgevoerd.

We zijn inmiddels aangeland bij versie 1.3. De technische uitwerking van elk verbetervoorstel is/wordt in de Leidraad opgenomen, direct in de systematiek (deel 1en/of 2) en/of in een van de bijlagen. Daarbij wordt tevens aangegeven of een verbetering normatief is, of informatief.

In deze bijlage wordt een inhoudelijk overzicht bijgehouden van de wijzigingen. De nummering – voor zover bekend – is overeenkomstig het statusoverzicht van de TC. In hoofdstuk 2 van de Leidraad zelf wordt een beknopt overzicht van wijzigingen gegeven dat naar de bijlagen verwijst.

## *1.1 Wijzigingen in versie 1.3 t.o.v. 1.2*

### 1.1.1 Vergroten bijlage capaciteit VISI-berichten

M0001. Dit betreft het vergroten van de bijlagecapaciteit van de VISI-berichten tot ca. 120 MB. Deze oplossing is gebaseerd op de MTOM-SOAP [1] implementatie van XOP (XML-binary Optimized Packaging) Zie ook: http://www.w3.org/TR/soap12-mtom/. Dit wordt voor verdere optimalisatie gecombineerd met ‘chunking’, een methode om bestanden in kleine stukken te versturen en vervolgens weer samen te voegen.

### 1.1.2 Verplicht stellen onderwerpveld

M0002. Het verplicht stellen van het onderwerpveld, zodat er geen berichten zonder onderwerpveld kunnen worden verzonden. Deze informele afspraak is door middel van een VISI-richtlijn geformaliseerd. Dit betekent dat de eindgebruikersinterface van een VISI-applicatie het invullen van een onderwerpsveld verplicht moet stellen. Als gevolg waarvan de applicatie deze tekst in het element ‘description’ van de ‘TransactionTemplate’ van Systematiek Deel 2 plaatst (zie bijlage 3).

### 1.1.3 Uniek volgnummer voor iedere transactie

M0004. Over transacties en inhoud daarvan wordt tevens gecommuniceerd via andere methoden dan VISI. Bijvoorbeeld: per telefoon, in een (bouw)overleg, per e-mail. In de genoemde communicatievormen kan alleen aan een transactie gerefereerd worden aan de hand van het onderwerp van de transactie. Iedere transactie heeft momenteel een unieke identificatie. Hiervoor wordt een

[1] ‘MTOM’ staat voor ‘Message Transmission Optimization Mechanism’. Dit is een W3C-standaardmethode om op efficiënte wijze binaire data naar en van webservices te versturen. Het maakt gebruik van XML-binary Optimized Packaging (XOP) om binaire data over te brengen en is bedoeld om zowel MIME- als DIME-bijlagen te vervangen (MIME staat voor ‘Multipurpose Internet Mail Extensions’, een internetstandaard voor e-mail die de structuur en codering van e-mailberichten vastlegt. DIME staat voor ‘Direct Internet Message Encapsulation). Binaire bestanden worden gewoonlijk gezien als een opeenvolging van bytes. Binaire bestanden bevatten typisch bytes die niet als tekstkarakters geïnterpreteerd moeten worden, maar die iets anders moeten voorstellen. In een SOAP-bericht wordt zo'n zogenaamde ‘array of bytes’ echter vaak gecodeerd als tekstdata. MTOM maakt het mogelijk om efficiënter de binary data van grote bestanden in a SOAP-request (aanvraag) of response (reactie) te plaatsen. Bron: http://nl.wikipedia.org/wiki/Message_Transmission_Optimization_Mechanism.

'Globally Unique Identifier' (GUID) gebruikt. Een voorbeeld van een GUID is 'ac0234d8-5c89-40c0-8afc-57cdbe9ac00c'. Deze identificatie is echter ongeschikt om per telefoon of in een overleg aan te geven welke transactie bedoeld wordt. Aanvullende voorwaarden zijn namelijk dat de identificatie kort is, snel mondeling over te dragen is, en herkenbaar is voor een mens. Systematiek Deel 2 (zie bijlage 3) is daartoe op twee plaatsen aangepast, te weten 1.13 en 1.7.

De oplossing behelst verder:

- Uitbreiden van de Entity 'TransactionTemplate' in VISI Systematiek Deel 2 met het verplichte attribuut 'number' van het type 'INTEGER'.

Deze template krijgt dan de volgende inhoud:

```
ENTITY TransactionTemplate;
 number : INTEGER;
 name : STRING;
 description : STRING;
 startDate : DATETIME;
 endDate : DATETIME;
 state : STRING;
 dateLamu : DATETIME;
 userLamu : STRING;
 result : OPTIONAL STRING;
 initiator : PersonInRole;
 executor : PersonInRole;
 project : ProjectTypeInstance;
END_ENTITY;
```

- Het number-attribuut bevat een volgnummer dat gecombineerd met de afkorting van de organisatie die de transactie start als referentie gebruikt kan worden om deze transactie mee aan te duiden. Dit geldt zowel voor hoofd- als subtransacties.
- De VISI implementatie toont in de gebruikersinterface bij de betreffende transactie een veld 'referentie' in het formaat <organisatie-afkorting> + <volgnummer>.
- Het transactievolgnummer (number) wordt eenmalig bepaald bij het starten van de transactie. Daarna blijft dit nummer gelijk.
- De organisatie-afkorting wordt vooraf via het projectspecifiek bericht gedistribueerd maar is ook in ieder verzonden bericht via Transaction > Initiator > PersonInRole > Organisation terug te vinden.
- Daarom moet ook de Entity 'OrganisationTemplate' van de VISI Systematiek Deel 2 worden uitgebreid met een 'abbreviation' attribuut:

```
ENTITY OrganisationTemplate;
 name : STRING;
 abbreviation: STRING;
 state : STRING;
 dateLamu : DATETIME;
 userLamu : STRING;
  contactPerson : PersonTemplate;
 template : ComplexElementTemplate;
END_ENTITY;
```

- Het abbreviation-attribuut bevat geen spaties, tabs of andere opmaaksymbolen.
- Voorgesteld wordt maximaal 5 bovenkastletters en/of cijfers.
- Een voorbeeld van een transactiereferentie: 'ABCD000036' wat staat voor een transactie met volgnummer 36 gestart door de organisatie aangeduid met 'ABCD'.
  In dit voorbeeld wordt gebruikgemaakt van voorloopnullen om alle referenties dezelfde lengte te geven en tekstuele sortering te faciliteren. Het daadwerkelijk toegepaste formaat wordt aan de

VISI implementerende partij overgelaten.

### 1.1.4 Machtiging om namens een andere gebruiker berichten te versturen

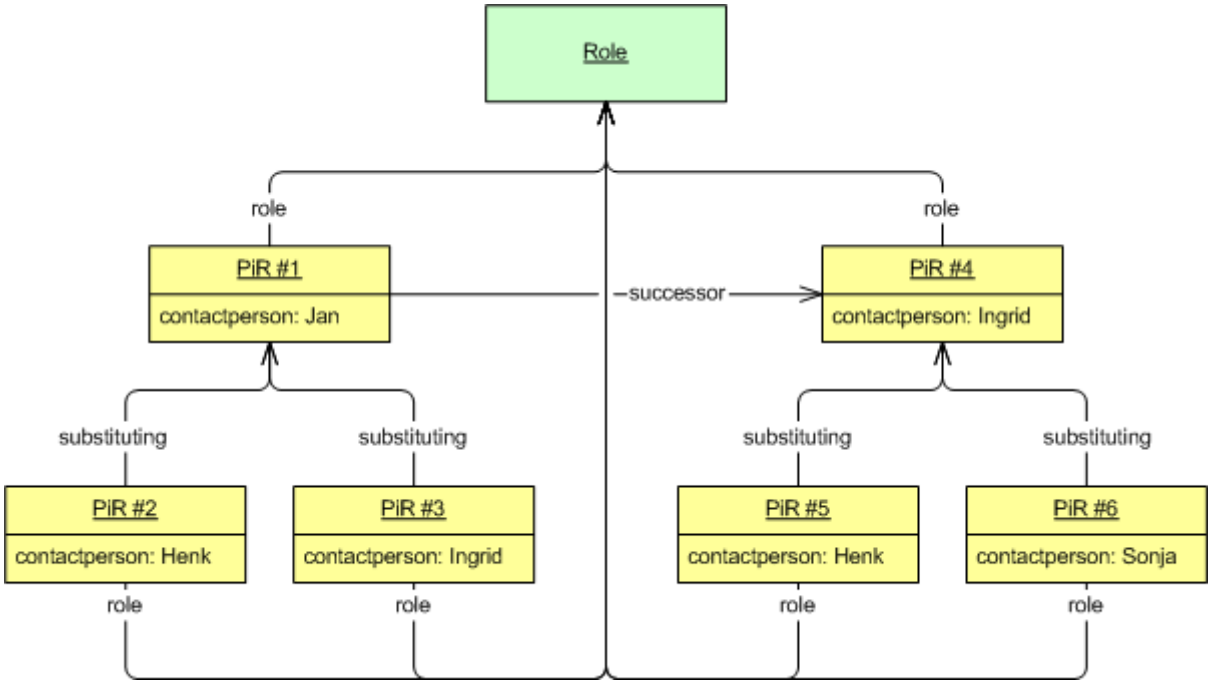
M0006. Dit betreft de transparantie bij tijdelijke vervanging van een persoon, oftewel het mogelijk maken om mensen te ‘machtigen’ om namens hem/haar berichten te verzenden. De Entity ‘PersonInRole’ in VISI Systematiek Deel 2 is daartoe uitgebreid met de optionele relatie ‘substituting’ van het type ‘PersonInRole’ (zie bijlage 3, paragraaf 1.8) . Hiermee kan een eenduidige manier van kortdurende vervanging geregeld worden. Tevens kan hiermee geregeld worden dat er ‘centrale’ boxen aangemaakt kunnen worden, waarmee meerdere mensen uit naam van de formele rolvervuller (bijvoorbeeld Directievoerder UAV) berichten kunnen verzenden.

*Probleemomschrijving*
Bij het op vakantie gaan van medewerkers is er op dit moment geen mogelijkheid om een vervangingsregeling op te maken, waarbij ook duidelijk is, wie de VISI-berichten verzonden heeft. Dit is zeer gewenst.
Hiermee kan een eenduidige manier van kortdurende vervanging geregeld worden. Tevens kan hiermee geregeld worden dat er "centrale" boxen aangemaakt kunnen worden, waarmee meerdere mensen uit naam van (bijvoorbeeld Directievoerder UAV) berichten kunnen verzenden.

*Oplossing*
De Entity ‘PersonInRole’ in VISI Systematiek II uitbreiden met de optionele relatie ‘substituting’ van het type ‘PersonInRole’.
Deze template krijgt dan de volgende inhoud:

```
ENTITY PersonInRole;
 state : STRING;
 dateLamu : DATETIME;
 userLamu : STRING;
 successor : OPTIONAL PersonInRole;
 substituting : OPTIONAL PersonInRole;
 contactPerson : PersonTemplate;
 organisation : OrganisationTemplate;
 role : RoleTemplate;
END_ENTITY;
```

*Voorbeeld van een mogelijke structuur*
In dit voorbeeld …
- Is in eerste instantie Jan degene die invulling geeft aan de rol.
- Hij kan daarbij vervangen worden door Henk of Ingrid.
- Later volgt Ingrid Jan op.
- Zij kan vervangen worden door Henk of Sonja.




De volgende regels gelden hierbij:

- De PersonInRole die fungeert als vervanger van een andere PersonInRole kan zelf niet naar een successor verwijzen. Alleen de PersonInRole die niet via een substituting relatie naar een andere PiR verwijst kan naar een successor verwijzen.
  Deze regel is vooral bedoeld om hele complexe structuren in te dammen. Om dezelfde reden moet ook nesting (vervanging van PiR die zelf ook al iemand in dezelfde rol vervangt) niet worden toegestaan.
- De PersonInRole die niet via een substituting relatie naar een andere PiR verwijst is automatisch de primaire rolvervuller. Namens haar of hem kunnen de vervangers berichten versturen.
- Bij de vaststelling of een PiR fungeert als substitute moet worden uitgegaan van de geldende rolvervuller zoals vastgelegd in de successor-structuur. In het voorbeeld is Ingrid in eerste instantie een vervanger van Henk maar volgt zij hem later op als de primaire rolvervuller.
- Uitgangspunt is dat een PersonInRole element nooit wordt "gerecycled". In het voorbeeld had dit gekund door bij de successor van Jan rechtstreeks naar het al bestaande PiR element van Ingrid te verwijzen (en met verwijdering van de substituting relatie bij Ingrid). Nadeel is dat daarmee de historie van rolvervulling wordt doorkruist, daarom wordt in principe altijd een nieuw PiR element toegevoegd.
- De getoonde structuur is bedoeld voor het projectspecifieke bericht. In een gewoon bericht worden beide PiR elementen (de echte afzender en degene namens wie dat gebeurt) meegestuurd en in de user-interface kan bij de specificatie van de afzender de primaire rolvervuller worden toegevoegd, bijvoorbeeld “Afzender: Ingrid namens/bij afwezigheid van Jan”.

### 1.1.5 Traceerbaarheid van communicatie waarborgen na aanpassen raamwerk

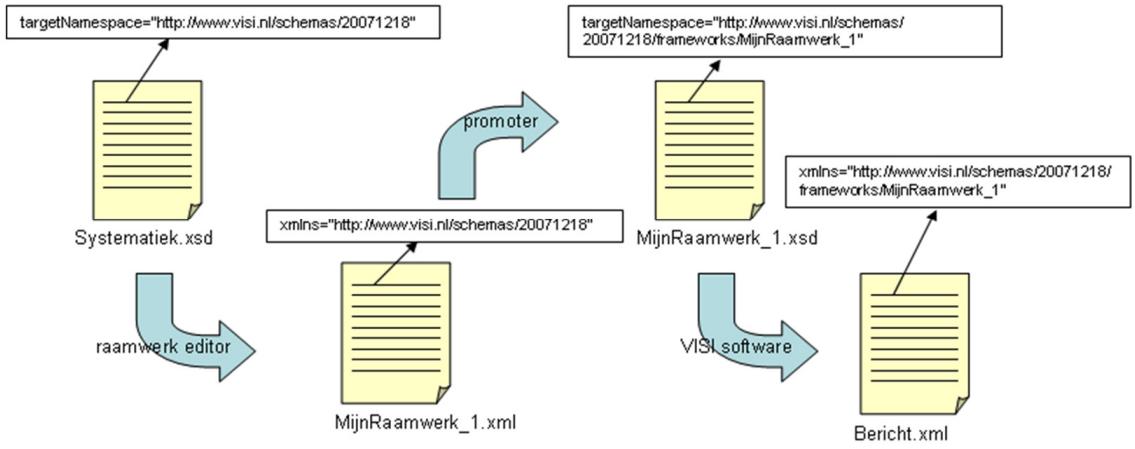
TC001. Dit is een verbetering in Systematiek Deel 1 (zie bijlage 2) van de bestaande functionaliteit om de traceerbaarheid van de communicatie na het aanpassen van een raamwerk te waarborgen. De huidige ‘namespace’ van het VISI berichtenverkeer wordt gewijzigd in een raamwerkspecifieke ‘namespace’ die is ontleend aan het nieuwe ‘namespace’ attribuut van de ProjectType entiteit van het raamwerk (zie paragraaf 1.9). De promotor software gebruikt dit gegeven om de correcte target ‘namespace’ te genereren.

Opmerking: VISI-software zal na invoering van VISI Systematiek versie 1.3 rekening moeten houden dat er meerdere versies van een raamwerk naast elkaar actief kunnen zijn.

*Motivatie*
Bij (langlopende) projecten is het wenselijk om gedurende het project het raamwerk aan te kunnen passen. Dit kan bijvoorbeeld bestaan uit:
- het toevoegen van transacties, maar ook
- het wijzigen van transacties, of
- het verwijderen van transacties.

Volgens versie 1.2 van de VISI systematiek kon het raamwerk van een project alleen gewijzigd worden indien er geen openstaande transacties zijn. Dit is een (te) zware eis die in veel praktijksituaties niet haalbaar is. Een raamwerk moet ook kunnen worden gewijzigd indien er nog *wel* openstaande transacties zijn. Het gevolg hiervan is dat er op een gegeven moment meer dan één versie van het raamwerk in een project actief kan zijn.
Onduidelijkheid over welke versie van het raamwerk toe te passen kan worden weggenomen door als regel te stellen dat een transactie die is gestart volgens een bepaald raamwerk ook verder volgens datzelfde raamwerk moet worden afgehandeld. Om deze regel te kunnen handhaven moet vanuit een bericht kunnen worden getraceerd volgens welke versie van het raamwerk de afhandeling moet plaatsvinden. Als dat eenduidig is vast te stellen, is de traceerbaarheid van de communicatie, in combinatie met het raamwerk op basis waarvan de communicatie heeft plaatsgevonden, gewaarborgd.

*Probleemanalyse*

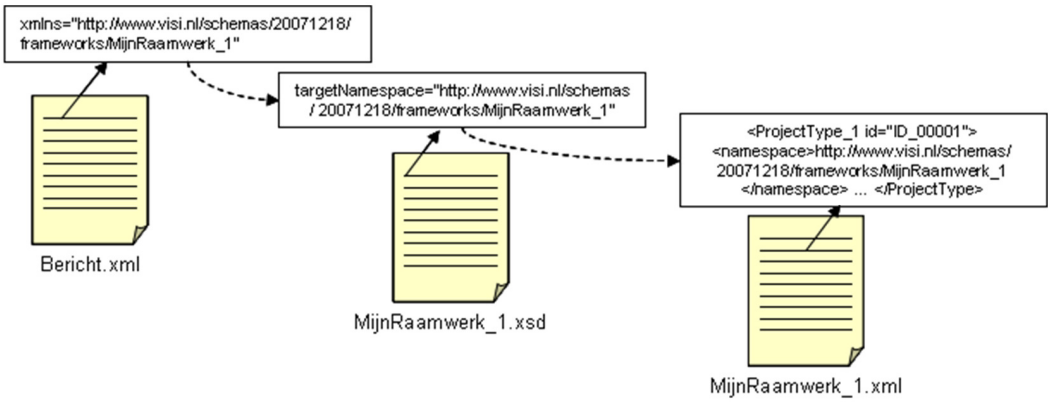



Volgens systematiek 1.2 was er gedurende de levenscyclus van een transactie maar één raamwerk actief. Er is sprake van één raamwerk en één projectspecifiek bericht en impliciet is de traceerbaarheid daarmee automatisch gewaarborgd. Kan deze relatie ook expliciet worden vastgesteld? Uit een eerdere analyse leek het gebruik van het attribuut ‘xsi:schemaLocation’ uitkomst te bieden. Echter bij een nog weer latere discussie is gebleken dat VISI bij het verzenden van berichten een verkeerde ‘namespace’ hanteert. Deze blijkt namelijk identiek te zijn aan de namespace van de systematiek zelf (waarin raamwerken gespecificeerd worden). Het berichtenverkeer moet volgens deze redenering een eigen namespace hebben die direct verbonden is met het toegepaste raamwerk. Daarmee zou dan ook automatisch de relatie bericht en toegepast raamwerk zijn vastgelegd.
De namespace van het raamwerk zou automatisch gegenereerd kunnen worden bijvoorbeeld met een hashing algoritme. Dit heeft als voordeel dat wanneer er iets aan het raamwerk wordt veranderd daarmee ook een andere namespace ontstaat. Dit geeft een garantie dat de berichten en het bijbehorende XML schema altijd met elkaar in overeenstemming zijn. Er kleven echter nogal wat praktische bezwaren tegen deze werkwijze. Zo leidt een iets andere opmaak van de XML of een niet

functionele wijziging in een description tekst al tot een een andere hash. In dit voorstel wordt de verantwoordelijkheid voor een nieuwe versie van het raamwerk en daarmee een andere namespace bij de raamwerkontwerper gelegd. De namespace specificatie van het raamwerk wordt onderdeel van het raamwerk zelf (opdat de promoter weet wat de target namespace van het gegenereerde XML schema zal worden).
Daarom is in versie 1.3 hiervoor een extra veld (namespace) aan de ProjectType entiteit toegevoegd (zie Bijlage 2, paragraaf 1.9):

```
ENTITY ProjectType;
 namespace : STRING;
 description : STRING;
 startDate : OPTIONAL DATETIME;
 endDate : OPTIONAL DATETIME;
 state : OPTIONAL STRING;
 dateLamu : OPTIONAL DATETIME;
 userLamu : OPTIONAL STRING;
 language : OPTIONAL STRING;
 category : OPTIONAL STRING;
 helpInfo : OPTIONAL STRING;
 code : OPTIONAL STRING;
 complexElements : OPTIONAL SET [0:?] OF ComplexElementType;
END_ENTITY;
```

(N.B. naar aanleiding van een andere richtlijn zijn hierboven ook nog de velden startDate, endDate, state, dateLamu en userLamu ‘optional’ gemaakt; zie Bijlage 13)




De traceerbaarheid kan dan als volgt worden gewaarborgd:

- Bij een binnenkomend bericht wordt het xmlns (namespace) attribuut gelezen.
- Dit is nodig om het bijbehorende XML schema te vinden op basis van het targetNamespace attribuut.
- Deze namespace is gegenereerd uit het namespace attribuut van het bijbehorende raamwerk, waarmee het raamwerk zelf kan worden vastgesteld.
- Tot slot geeft het xmlns (namespace) attribuut van het raamwerk duidelijkheid bij welke Systematiek versie het betreffende raamwerk behoort.

Bovenstaande traceerstrategie veronderstelt wel dat de VISI-implementatie deze zoekbewerkingen kan uitvoeren bijvoorbeeld door hiervoor een register te onderhouden.

*Voorstel* (zoals in Systematiek Deel 1 1.3 verwerkt)
Wijzig de huidige namespace van het VISI berichtenverkeer voor een raamwerk-specifieke namespace die is ontleend aan het namespace attribuut van de ProjectType entiteit van het raamwerk. De promoter software zal dit gegeven gebruiken om de correcte target namespace te kunnen genereren.

*Implementatiegevolgen*
VISI software zal na invoering van deze verbetering er rekening mee moeten houden dat er meerdere versies van het raamwerk naast elkaar actief kunnen zijn. Bij het verwerken van een inkomend bericht bepaalt het xmlns attribuut volgens welke raamwerkversie het bericht gedecodeerd moet worden. Deze informatie bepaalt tevens welke acties kunnen voortvloeien uit de ontvangst van dit bericht. Een eventueel vervolgbericht zal uiteindelijk gevalideerd moeten worden volgens dezelfde versie van het raamwerk.

### 1.1.6 Appendixtypes koppelen aan transacties

TC006. Dit betreft een verbetering in Systematiek Deel 2 (zie bijlage 3), om in een raamwerk aan te kunnen geven wanneer welk AppendixType gekozen mag worden. Er bestaan namelijk raamwerken waarin meerdere AppendixTypes aangemaakt zijn voor verschillende type bijlagen. Omdat dit erg veel types kunnen zijn, wordt de gebruiker hiermee geholpen bij de keuze van het juiste AppendixType.

### 1.1.7 Standaard en richtlijn voor het archiveren van VISI-projecten

TC010. Dit betreft de eisen waaraan het uitwisseling- en archiveringsformaat voor afgeronde VISI projecten (het 'VISI archief') moet voldoen; zie bijlage 11. Als voorbeeld wordt tevens een (mogelijke) technische oplossing uitgewerkt.
De correcte heropbouw van berichten, raamwerk(en) en project specifieke bericht(en) kan uitsluitend met projecten waarbij alle berichten werken met een 'DateTime value' (dat is geïmplementeerd vanaf de VISI-systematiek versie 1.2). Indien de VISI software beschikt over het tijdstip waarop een VISI bericht is verstuurd, dan wordt deze informatie ook gearchiveerd.
Bij wijze van 'tip': in de systematiek versie 1.1 werd alleen met een 'Date value' gewerkt waardoor het niet mogelijk is om berichten per dag in de juiste volgorde terug te zetten. Omdat de DateTime informatie niet aanwezig is, kan voor het eerste bericht binnen een transactie op die dag tijdstip '00:01' worden ingevuld, en kan de software elk opvolgend bericht een minuut later als tijdstip meegeven (N.B. dat betekent dus dat het aantal berichten binnen een transactie binnen een dag is 'gelimiteerd' tot 1440).

### 1.1.8 Metaraamwerk

TC013. De wijze waarop VISI-communicatie voor een project wordt gestart en beheerd bleek niet duidelijk omschreven en wordt in de praktijk vaak via informele kanalen (telefoon, e-mail) aangestuurd. De TC heeft daarvoor de oplossing bedacht om de VISI systematiek zelf te gebruiken voor het managen van VISI-communicatie binnen een project. Het gaat dan om communicatie over de wijze van communiceren, oftewel metacommunicatie en in de VISI context is daar een raamwerk voor nodig of eigenlijk een 'metaraamwerk'. Dit is een procedure met transacties voor het verspreiden van een nieuwe versie van een raamwerk en/of een projectspecifiek bericht.
Bij (langlopende) projecten is het wenselijk om gedurende het project het raamwerk aan te kunnen passen. Voorheen kon het raamwerk van een project alleen gewijzigd worden indien er geen openstaande transacties zijn. Dit is een zware eis die in veel praktijksituaties niet haalbaar zal zijn. Een projectraamwerk moet daarom ook gewijzigd kunnen worden indien er nog wel openstaande transacties zijn. Het metaraakwerk is beschreven in bijlage 6.

### 1.1.9 Oneindige loops/successor

TC015. Dit betreft een werkwijze waarbij een gebruiker die een bepaalde rol in het VISI-raamwerk vervult, (door middel van een PersonInRole element (PIR)) wordt opgevolgd door een andere gebruiker. Hiervoor dient het link-attribuut 'successor' (in Systematiek Deel 1). Zie ook bijlage 7. Het is denkbaar dat in deze datastructuur op een bepaald moment een lus zou kunnen ontstaan doordat een gebruiker al eens eerder dezelfde rol heeft vervuld. Dat wordt ondervangen door het metaraamwerk bij

systematiek versie 1.3. Bijlage 7 is dus in feite meer gericht op de raamwerkbouwer, dan op de implementator van VISI.
N.B. In systematiek versie 1.2 kan dit probleem worden ondervangen door de PIR niet te “recyclen” maar een nieuwe PIR aan te maken. Dit geschiedt door het opstellen van het een Project Specifiek Bericht (PSB).

### 1.1.10 Aanvullende functionele eisen

TC016. Naar aanleiding van gebruikerswensen is een aantal functies en eisen beschreven die gebruikers van producten met het VISI-keurmerk minimaal van het desbetreffende product mogen verwachten. Dit komt voort uit:

- de verplichting van gebruikers om de authenticiteit van uitgewisselde VISI-berichten en bijbehorende bestanden te borgen;
- de wens om de communicatiestructuur die in een VISI-raamwerk is vastgelegd, op een herkenbare wijze te presenteren;
- de wens om gevoerde VISI-communicatie achteraf te kunnen reproduceren.

Deze aanvullende eisen gelden uitsluitend als het VISI-product een user-interface voor de eindgebruiker bevat. De aanvullende eisen hebben eveneens betrekking op het verkrijgen van het VISI-keurmerk. Deze eisen zijn uitgewerkt in bijlage 10.

### 1.1.11 Richtlijn Element Conditions

TC017. Element conditions kunnen worden toegepast op verschillende niveaus. Een conditie op een bepaald niveau kan invloed hebben op het gedrag van Simple Elements op andere niveaus. In de richtlijn wordt aangegeven hoe hier mee om moet worden gegaan. Zie bijlage 12.

### 1.1.12 Dubbele messagetypes

TC022. Dit betreft een aanpassing in Systematiek Deel 2 (bijlage 3), te weten de uitbreiding van de ‘Referenties’ van de entiteit ‘MessageInTransactionType’ met de eigenschap ‘firstMessage’. Daarmee kan expliciet worden aangegeven of een MessageInTransactionType een bericht betreft waarmee een nieuwe transactie kan worden gestart.
Deze nieuwe eigenschap is een optionele Boolean (‘True’, of ‘False’). Indien deze Boolean afwezig is, geldt de standaard waarde ‘False’, en kan met de MessageInTransactionType geen nieuwe transactie worden gestart. Wanneer deze Boolean aanwezig is en de waarde ‘True’ heeft, kan er met de MessageInTransactionType een nieuwe transactie worden gestart.
Indien voor een MessageInTransactionType geen “previous” is ingevuld wordt dit automatisch een startbericht , een firstMessage waarde is dan overbodig en dient te worden genegeerd.

### 1.1.13 Boolean waarden

TC024. Over logische (Boolean) waarden zegt de XML Schema Part 2, Datatypes Second Edition: *“An instance of a datatype that is defined as ‘Boolean’ can have the following legal literals {true, false, 1, 0}”*. Het maakt dus in het algemeen niet uit of de waarde ‘1’ of ‘True’ danwel ‘0’ of ‘False’ wordt gebruikt. Beide moet kunnen, met dien verstande dat ‘1’ = ‘True’, en ‘0’ = ‘False’. Daarnaast wordt hier bepaald dat ‘geen waarde’ gelijk is aan ‘0’, cq. ‘False’; dit omdat binnen de VISI-systematiek ‘geen waarde’ namelijk ook valide is.

### 1.1.14 Optionele velden

Het doel hiervan is om raamwerken (xml en xsd) compacter te maken. Dit is gunstig voor VISI gebruikers, omdat een omvangrijk en inefficiënt raamwerk leidt tot vertraging in de afhandeling van VISI-berichten en het gebruik van VISI-software. Een compacter en efficiënter raamwerk resulteert in vergrootte schaalbaarheid en snelheid van VISI-software. Bijkomend voordeel is de leesbaarheid van raamwerken en VISI-berichten vergroot wordt. Zie bijlage 13.

## 1.2 Wijzigingen in versie 1.2 t.o.v. eerdere versies

### 1.2.1 Richtlijn voor transactiekoppeling; communicatie over meerdere transacties

De essentie van VISI is gebaseerd op transacties tussen twee rollen. Vanuit de praktijk is de wens gekomen om processen over meerdere transacties te ondersteunen. Dit heeft bijvoorbeeld te maken met het feit dat VISI in de praktijk niet alleen meerwaarde blijkt te hebben op het raakvlak van partijen, maar ook binnen partijen zelf. Daarnaast komt het ook voor dat een partij (executor) een transactie van een andere partij (initiator) pas kan afronden door of na transacties met (een) derde(n). In beide gevallen gaat het om processen die over meerdere transacties verlopen. Vanaf versie 1.2 wordt de mogelijkheid van communicatie over meerdere transacties ondersteund. Een belangrijk uitgangspunt hierbij is de traceerbaarheid van de communicatie achteraf.

#### 1.2.1.1 Inleiding

De VISI richtlijn voor transactiekoppeling bestaat uit twee delen. Het eerste deel behandelt de functionaliteit om met meer rollen over één onderwerp (transactie) te communiceren. Het tweede deel is gericht op de functionaliteit om een primaire transactie uit te zetten naar meerdere derde personen met verschillende instanties van een secundaire transactie.
De richtlijn is tot stand gekomen op basis van gevoerde discussies in het VISI Technisch Comité.

#### 1.2.1.2 Deel 1: Communiceren met meerdere rollen over één onderwerp

***VISI Transactions***

***Doel***
De huidige systematiek staat al toe transacties te definiëren welke afhankelijk zijn van andere transacties. Hiermee wordt een belangrijke vraag opgelost waarbij men met meerdere rollen (meer dan 2) over 1 onderwerp wil communiceren. Hoewel dit al mogelijk is tijdens het definiëren van een raamwerk, is de informatie die gegenereerd wordt tijdens het aanmaken van berichten onvoldoende om een ‘gekoppelde transactie juist op te slaan en een juiste historie bij te houden. Dit wordt aangepast met een uitbreiding van Systematiek Deel 2 (waarin wordt beschreven welke elementen een bericht bevat).

***Aanpassingen Systematiek***

***Systematiek Deel 1 behoeft geen aanpassingen***

***Systematiek Deel 2 behoeft een aanpassing in MessageTemplate***

*Oorspronkelijke versie*

```
ENTITY MessageTemplate;
 identification : STRING;
 dateSend : DATE;
 dateRead : DATE;
 state : STRING;
 dateLamu : DATE;
 userLamu : STRING;
 initiatorToExecutor : BOOLEAN;
 transaction : TransactionTemplate;
 template : ComplexElementTemplate;
END_ENTITY;
```

*Voorgestelde aanpassing*

```
ENTITY MessageTemplate;
 identification : STRING;
 dateSend : DATE;
 dateRead : DATE;
 state : STRING;
 dateLamu : DATE;
 userLamu : STRING;
 initiatingTransactionMessageID : OPTIONAL STRING;
 initiatorToExecutor : BOOLEAN;
 transaction : TransactionTemplate;
 template : ComplexElementTemplate;
END_ENTITY;
```

***Note: initiatingTransactionMessageID is de ID van het voorgaande bericht***

Het STRING veld bevat meer mogelijkheden dan de waarde van een ID, doch een constructie waar een exact ID ingevoerd kan worden is zo ingrijpend dat voor deze relatief simpele oplossing is gekozen.

***Specifieke keuzes en achterliggende argumentaties***

***Voorgaande berichten worden niet als compleet bericht opgenomen:***

*Voordelen*

- Een voorgaand bericht zou ook een attachment kunnen bevatten, als deze attachment groot is ontstaat er een hoop vervuiling.
- Een voorgaand bericht zou ook weer een voorgaand bericht kunnen bevatten dit kan tot een opstapeling van voorgaande berichten leiden.
- Huidige systemen zijn ingericht op 1 bericht per 12.xml file (m.u.v. project specifiek bericht, waar 0 berichten per 12.xml file aanwezig zijn). Deze keuze houdt dit valide.
- Een systeem weet zelf zijn historie en heeft derhalve geen problemen met het terugvinden van de voorgaande berichten, de ID wordt gebruikt en deze is uniek .

*Nadelen*

- Een bericht bevat als losstaand bericht niet alle informatie zonder externe referenties zoals tot nu toe wel het geval was. Het is dus niet mogelijk om op basis van 1 bericht de gehele context van dit bericht weer te geven (verticaal),

*Conclusie*

- Het nadeel weegt niet op tegen de voordelen. Derhalve is gekozen om alleen referentie naar bovenliggend bericht aan te geven. Dit nadeel is (deels) goed te praten daar op basis van alleen 1 bericht al nooit gehele horizontaal context (zeg maar voorgaande en nakomende berichten) te achterhalen zijn, waardoor het ook plausibel is dat dit ook geldt voor verticale berichten (uit aanroepende transactie, dan wel berichten aangeroepen vanuit deze transactie).

***Berichten die niet als direct voorgaand bericht een bericht uit een andere transactie hebben, hebben nog steeds de referentie naar het bericht uit deze transactie***

*Voordelen*

- Conceptueel is het mooi bij overgang naar ander bericht zoveel mogelijk informatie gelijk te houden.
- Uit elk bericht is na te gaan of er een aanroepende transactie is waartoe als reactie op een bericht teruggestapt kan worden.

*Nadelen*

- Uit een bericht alleen is niet (altijd) af te lijden of het voorgaande bericht het bericht uit de andere transactie was.

*Conclusie*

- Het nadeel is alleen van toepassing als een gekoppelde transacties aangegeven moeten worden. In deze gevallen is de applicatie zich bewust van alle berichten en heeft het voldoende informatie uit alle berichten samen om de gekoppelde transactie aan te geven. De voordelen wegen dus op tegen het nadeel omdat het nadeel slechts slimmere applicaties eist en geen overall informatieverlies oplevert.

***Voor bijlagen bij berichten is geen specifieke oplossing gezocht***

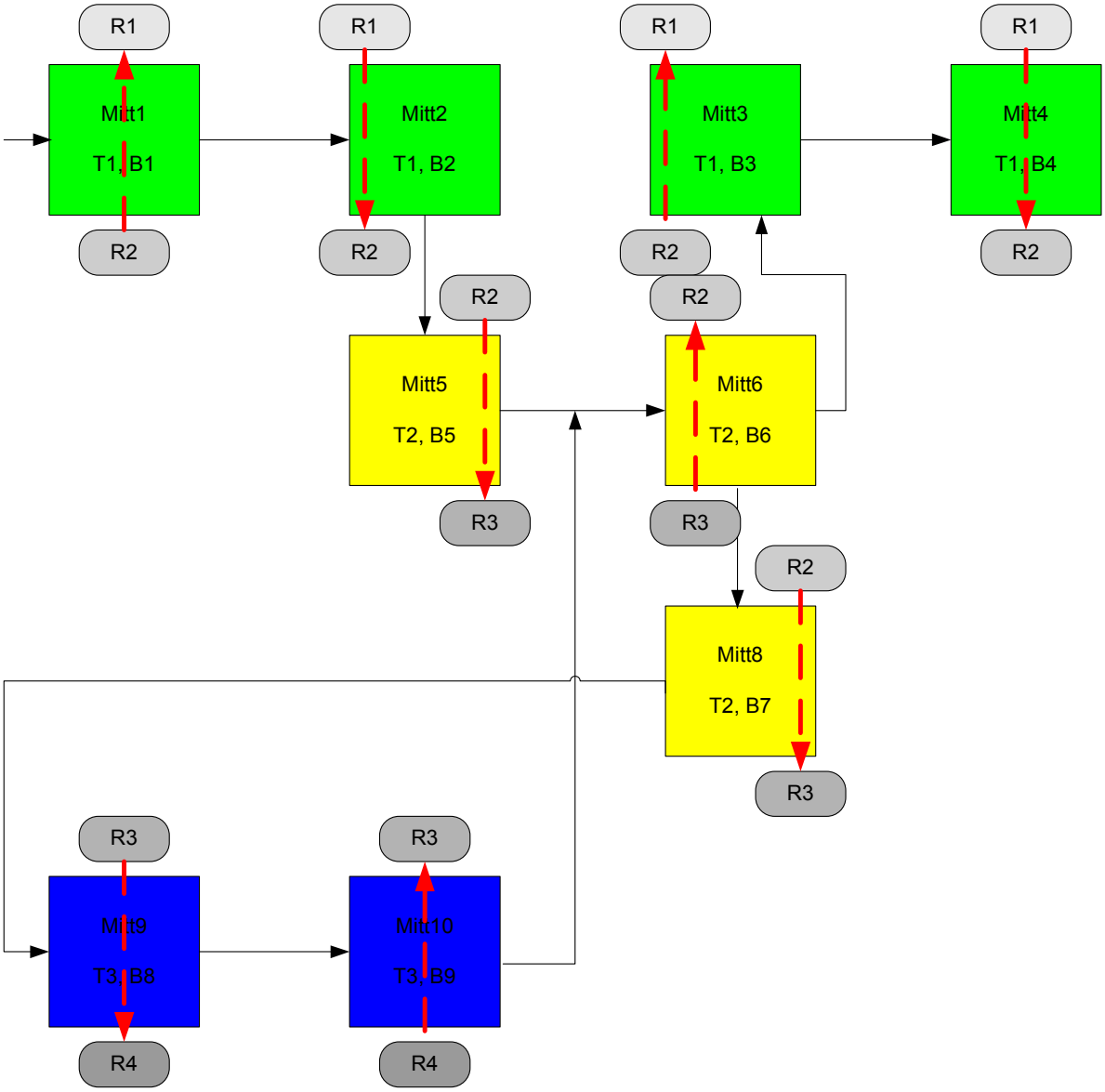
*Voordelen*

- Conceptueel is het in lijn met andere VISI aanpak om de keuze van bijlagen zelf aan applicaties over te laten.

*Nadelen*

- Niet alle bijlagen staan automatisch op alle servers behorende bij rollen R1 t/m R4

***Structuur van het raamwerk***




**Figuur 1**

De bovengenoemde zwarte pijlen geven de berichtenvolgorde in tijd weer. De rode pijlen geven weer welke rol het bericht verzendt en welke rol het bericht ontvangt.

De zwarte pijlen zijn niet aangegeven in de 12.xml files. Dit is informatie welke de applicatie moet onthouden indien vereist. Een uitzondering hierop vormen de pijlen waarbij een andere Transactie wordt geïnitieerd, dus van *B2 naar B5*, deze wordt opgeslagen in de instances van Mitt5, Mitt6 en Mitt8) en van *B7 naar B8*, deze wordt opgeslagen in de instances van Mitt9 en Mitt10).
De richting van de rode pijlen is aangegeven in elk 12.xml file. Elk bericht heeft altijd exact 1 rode pijl (en twee rollen).

***Raamwerk in XML (_7.xml part)***

***Bericht B1 (12.xml part) T1***

```
<?xml version="1.0" encoding="UTF-8"?>
<B1 id="">
        ...
        <initiatorToExecutor>false</initiatorToExecutor>
        <T1 id="">
                ...
                <initiator>
                        <R1 id="">
                                ...
                        </R1>
                </initiator>
                <executor>
                        <R2 id="">
                                ...
                        </R2>
                </executor>
                ...
        </T1>
        ...
</B1>
```

***Bericht B2 (12.xml part) T1***

```
<?xml version="1.0" encoding="UTF-8"?>
<B2 id=" messageB2">
        ...
        <initiatorToExecutor>true</initiatorToExecutor>
        <T1 id="">
                ...
                <initiator>
                        <R1 id="">
                                ...
                        </R1>
                </initiator>
                <executor>
                        <R2 id="">
                                ...
                        </R2>
                </executor>
```

```
            ...
        </T1>
        ...
</B2>
```

***Bericht B5 (12.xml part)  T2***

```
<?xml version="1.0" encoding="UTF-8"?>
<B5 id="">
        ...
        <initiatingTransactionMessageID>messageB2</initiatingTransactionMessageID>
        ...
        <initiatorToExecutor>true</initiatorToExecutor>
        <T2 id="">
                ...
                <initiator>
                        <R2 id="">
                                ...
                        </R2>
                </initiator>
                <executor>
                        <R3 id="">
                                ...
                        </R3>
                </executor>
                ...
        </T2>
        ...
</B5>
```

***Bericht B6 (12.xml part)  T2***

```
<?xml version="1.0" encoding="UTF-8"?>
<B6 id="">
        ...
        <initiatingTransactionMessageID>messageB2</initiatingTransactionMessageID>
        ...
        <initiatorToExecutor>false</initiatorToExecutor>
        <T2 id="">
                ...
                <initiator>
                        <R2 id="">
                                ...
                        </R2>
                </initiator>
                <executor>
                        <R3 id="">
                                ...
                        </R3>
                </executor>
                ...
        </T2>
```

```
        ...
</B6>
```

***Bericht B7 (12.xml part)  T2***

```
<?xml version="1.0" encoding="UTF-8"?>
<B7 id="messageB7">
        ...
        <initiatingTransactionMessageID>messageB2</initiatingTransactionMessageID>
        ...
        <initiatorToExecutor>true</initiatorToExecutor>
        <T2 id="">
                ...
                <initiator>
                        <R2 id="">
                                ...
                        </R2>
                </initiator>
                <executor>
                        <R3 id="">
                                ...
                        </R3>
                </executor>
                ...
        </T2>
        ...
</B7>
```

***Bericht B8 (12.xml part)  T3***

```
<?xml version="1.0" encoding="UTF-8"?>
<B8 id="">
        ...
        <initiatingTransactionMessageID>messageB7</initiatingTransactionMessageID>
        ...
        <initiatorToExecutor>true</initiatorToExecutor>
        <T3 id="">
                ...
                <initiator>
                        <R3 id="">
                                ...
                        </R3>
                </initiator>
                <executor>
                        <R4 id="">
                                ...
                        </R4>
                </executor>
                ...
        </T3>
        ...
</B8>
```

***Bericht B9 (12.xml part)  T3***

```
<?xml version="1.0" encoding="UTF-8"?>
<B9 id="">
        ...
        <initiatingTransactionMessageID>messageB7</initiatingTransactionMessageID>
        ...
        <initiatorToExecutor>false</initiatorToExecutor>
        <T3 id="">
                ...
                <initiator>
                        <R3 id="">
                                ...
                        </R3>
                </initiator>
                <executor>
                        <R4 id="">
                                ...
                        </R4>
                </executor>
                ...
        </T3>
        ...
</B9>
```

***Bericht B6 (12.xml part)`  T2***

```
<?xml version="1.0" encoding="UTF-8"?>
<B6 id="">
        ...
        <initiatingTransactionMessageID>messageB2</initiatingTransactionMessageID>
        ...
        <initiatorToExecutor>false</initiatorToExecutor>
        <T2 id="">
                ...
                <initiator>
                        <R2 id="">
                                ...
                        </R2>
                </initiator>
                <executor>
                        <R3 id="">
                                ...
                        </R3>
                </executor>
                ...
        </T2>
        ...
</B6>
```

***Bericht B3 (12.xml part) T1***

```
<?xml version="1.0" encoding="UTF-8"?>
<B3 id="">
        ...
        <initiatorToExecutor>false</initiatorToExecutor>
        <T1 id="">
                ...
                <initiator>
                        <R1 id="">
                                ...
                        </R1>
                </initiator>
                <executor>
                        <R2 id="">
                                ...
                        </R2>
                </executor>
                ...
        </T1>
        ...
</B3>
```

***Bericht B4 (12.xml part) T1***

```
<?xml version="1.0" encoding="UTF-8"?>
<B4 id="">
        ...
        <initiatorToExecutor>true</initiatorToExecutor>
        <T1 id="">
                ...
                <initiator>
                        <R1 id="">
                                ...
                        </R1>
                </initiator>
                <executor>
                        <R2 id="">
                                ...
                        </R2>
                </executor>
                ...
        </T1>
        ...
</B4>
```

#### 1.2.1.3 Deel 2: Meerdere instanties van een secundaire transactie

***Problematiek***

Huidige situatie:
- Alle transacties vinden plaats tussen twee partijen.
- Het is mogelijk transacties te koppelen, hierbij gaat een transactie over naar een andere transactie en aan het einde van deze transactie terug naar de eerste transactie (dit kan genest).

Wens:
- Een transactie doorzetten naar meerdere derde partijen.

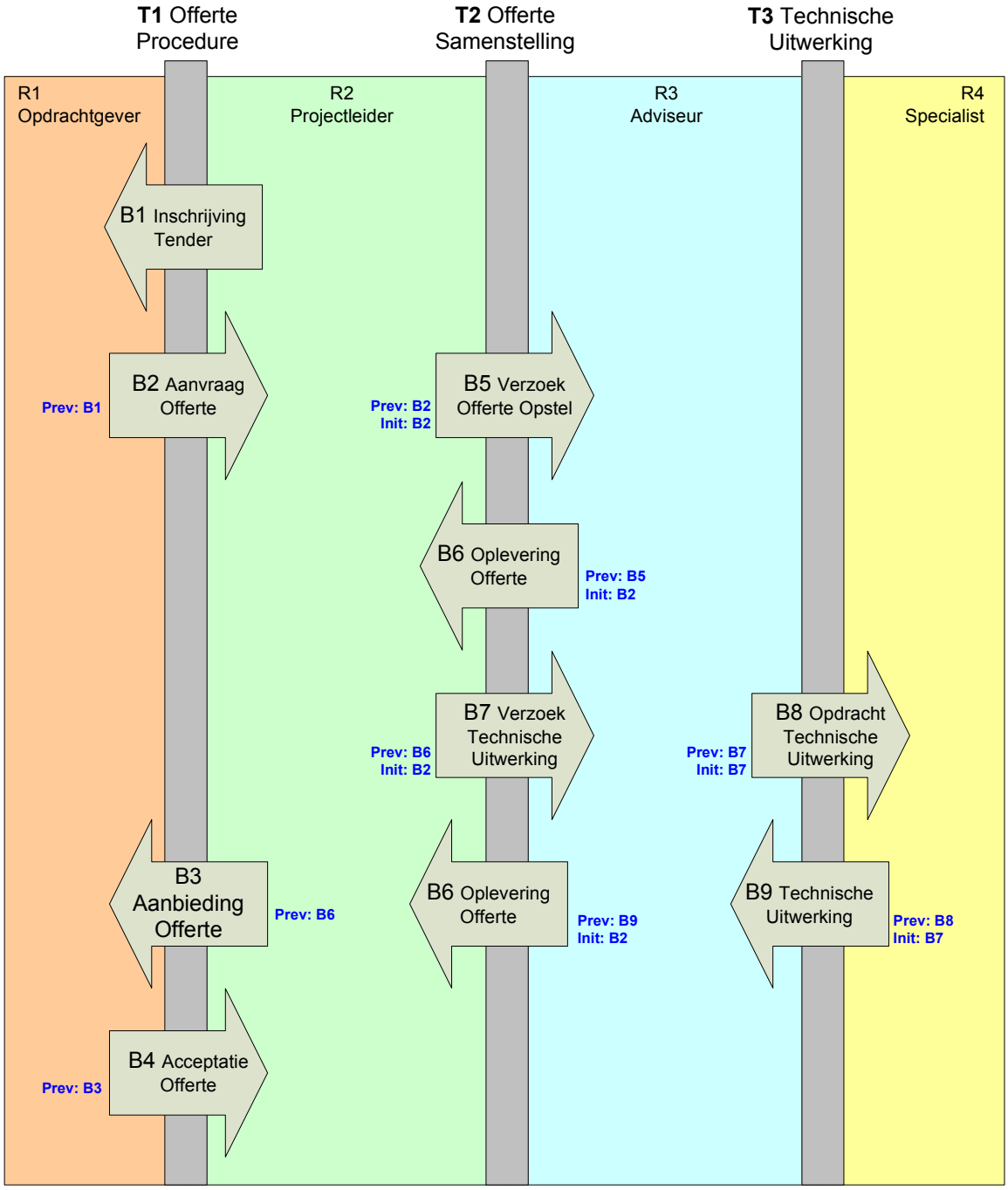
***Voorbeeld***

Stel we hebben de situatie volgens Figuur 2.

Hierin vinden we 3 transacties. De transactieovergangen van T1 naar T2 en van T2 naar T3 is reeds beschreven in notitie ‘Richtlijn voor transactiekoppeling’.

Huidige situatie:
Op dit moment is iemand als projectleider (R2) vanuit B2 (T1) gemachtigd exact één adviseur te verzoeken een offerte op te stellen (B5). Op zijn beurt kan de persoon zijnde adviseur (R3) bij bericht B7 (T2) exact één specialist opdracht geven e.e.a. technisch uit te werken (B8).

Wens:
De wens is dat de projectleider (R2) vanuit B2 (T1) meerdere adviseurs kan raadplegen, daarnaast dat ook de adviseur (R3)vanuit B7 (T2) ook meerdere specialisten kan raadplegen.




**Figuur 2** Source Custom

***Technische uitwerking***
We onderscheiden 3 verschillende situaties die we willen ondersteunen:

1. vanuit een bericht in transactie X willen we meerdere transacties starten van type transactie Y waarbij we weer willen doorgaan met transactie X als ALLE transacties Y zijn afgerond.
2. vanuit een bericht in transactie X willen we meerdere transacties starten van type transactie Y waarbij we weer willen doorgaan met transactie X als tenminste 1 van de type Y transacties is afgerond (1 OF MEER).
3. vanuit een bericht in transactie X willen we meerdere transacties starten van type transactie Y waarbij we weer willen doorgaan met transactie X ongeacht het aantal afgeronde transacties van het type Y zijn afgerond (0 OF MEER).

NOTE: Voor alle situaties gaan we ervanuit dat er een ongelimiteerd aantal secundary transacties (van het transactietype Y) mogen worden gestart.

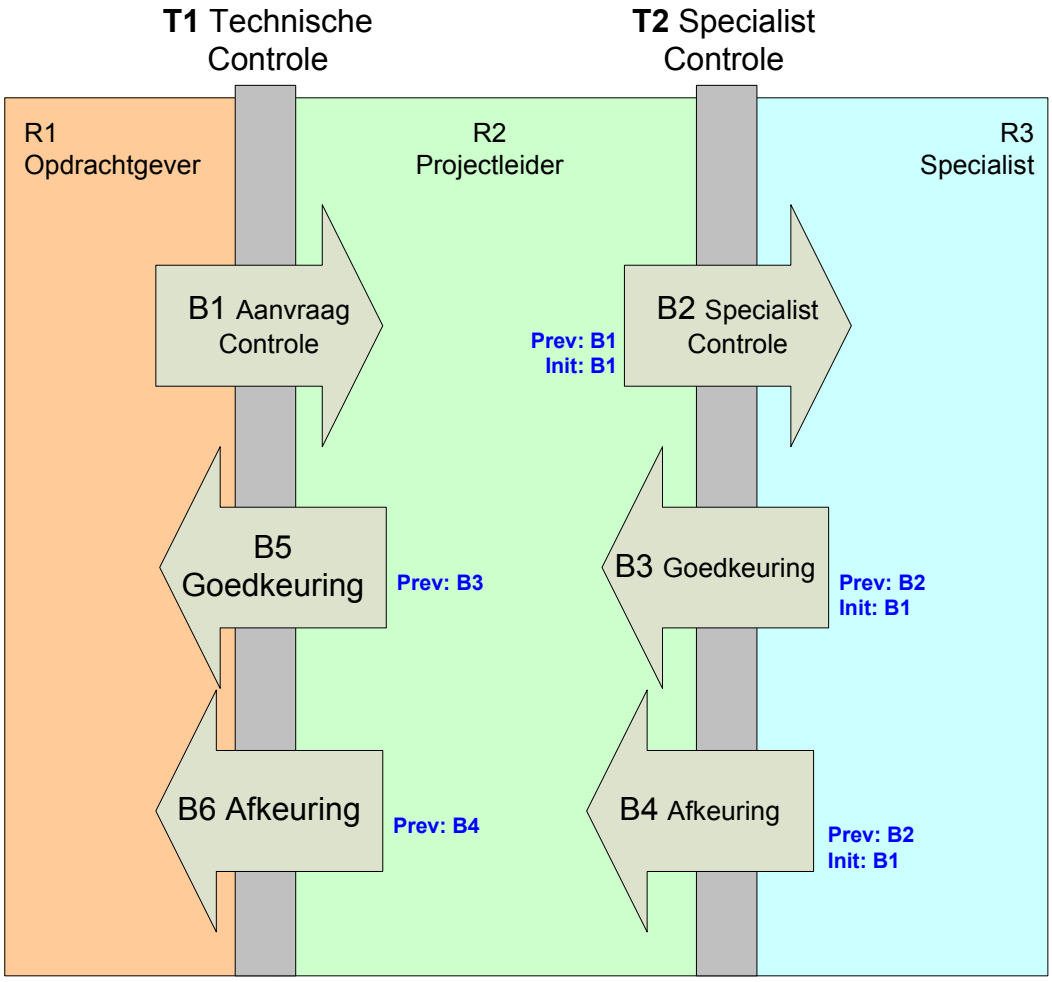
Een nieuw element **openSecondaryTransactionsAllowed** in **MessageInTransactionType** geeft aan of bij het versturen van een bericht (uit transactie X) gewacht moet worden tot alle voorgaande berichten (indien komende uit een andere secondaire transactie Y) moeten zijn voltooid. Default waarde is TRUE, dus indien TRUE of niet gedefinieerd is er slechts 1 voorgaand bericht nodig en mogen alle andere gestarte transacties van het type Y nog ergens bezig zijn. Indien FALSE dan zal gewacht moeten worden tot alle transacties van het type Y zover zijn dat ze aan dit bericht uit transactie X toe zijn.

***Situatie 1***
Als we ons voorbeeld van Figuur 2 erbij pakken en in alle gevallen willen dat alle transacties T3 zijn afgerond voor B6 en alle transacties T2 zijn afgerond voor B3 dan zullen MessageInTransactionType voor B6 en voor B3 **openSecondaryTransactionsAllowed** op FALSE gezet moeten zijn**.**

***Situatie 2***
Als we ons voorbeeld van Figuur 2 erbij pakken en in alle gevallen willen dat ten minste 1 transactie T3 is afgerond voor B6 en tenminste 1 transacties T2 is afgerond voor B3 dan zullen MessageInTransactionType voor B6 en voor B3 **openSecondaryTransactionsAllowed** op TRUE gezet moeten zijn. Daar TRUE de default value is, mag dit gewoon weggelaten worden.

***Situatie 3***
Als we ons voorbeeld van Figuur 2 erbij pakken en in alle gevallen willen dat geen enkele T3 afgerond hoeft te zijn voor B6 en geen enkele transacties T2 afgerond hoeft te zijn voor B3 dan kunnen we eenvoudig B2 als voorgaande MessageInTransactionType van B3 definiëren en B7 als voorgaand MessageInTransactionType van B6.

***Speciale Situaties***
In veel gevallen is een combinatie gewenst van beide, Zie Figuur 3.




**Figuur 3**

We zien hier een controle waarbij de opdrachtgever (R1) de projectleider (R2) vraagt een controle uit te voeren. De projectleider (R2) vraagt vervolgens verschillende specialisten (R3) een controle uit te voeren.

Wat we nu willen is de mogelijkheid iets af te keuren na 1 feedback van een specialist en pas goedkeuring te kunnen geven nadat alle specialisten goedkeuring hebben gegeven.

Dit betekent dat B5 zich in Situatie 1 bevindt en dat B6 zich in Situatie 2 bevindt. Technisch betekent dit dat MessageInTransactionType voor B5 **openSecondaryTransactionsAllowed** op FALSE heeft staan. (B6 is gewoon default en maakt net als alle andere MessageInTransactionTypes dus geen gebruik van dit element).

***Aanpassingen Interpretatie***

De volgende nieuwe aanname is gemaakt:

- bij een overgang naar een andere transactie (zie ook notitie 'Richtlijn voor transactie-koppeling') kan een ongelimiteerd aantal nieuwe instanties van de nieuwe Transactie worden aangemaakt.
- er mogen geen andere transacties meer geïnitieerd worden op het moment dat de initiërende transactie naar een volgend bericht is gegaan.
- indien het nieuwe element **openSecondaryTransactionsAllowed** van **MessageInTransactionType** niet is gedefinieerd wordt hij geïnterpreteerd als zijnde TRUE.

Gevolgen:

- openstaande geïnitieerde transacties kunnen dus blijven doorlopen ook al is de initiërende transactie al verder of zelfs afgerond.

***Aanpassingen Systematiek***

**Systematiek Deel 1 behoeft een aanpassing in MessageInTransactionType**

*Oorspronkelijke versie*

```
ENTITY MessageInTransactionType;
 requiredNotify : INTEGER;
 dateLamu : DATETIME;
 userLamu : STRING;
 received : BOOLEAN;
 send : BOOLEAN;
 state : STRING;
 initiatorToExecutor : OPTIONAL BOOLEAN;
 message : MessageType;
 previous : OPTIONAL SET [0:?] OF MessageInTransactionType;
 transaction : TransactionType;
 transactionPhase : OPTIONAL TransactionPhaseType;
 previous : OPTIONAL SET [0:?] OF MessageInTransactionType;
END_ENTITY;
```

*Voorgestelde aanpassing*

```
ENTITY MessageInTransactionType;
 requiredNotify : INTEGER;
 dateLamu : DATETIME;
 userLamu : STRING;
 received : BOOLEAN;
 send : BOOLEAN;
 state : STRING;
 initiatorToExecutor : OPTIONAL BOOLEAN;
 openSecondaryTransactionsAllowed : OPTIONAL BOOLEAN;
 message : MessageType;
 previous : OPTIONAL SET [0:?] OF MessageInTransactionType;
 transaction : TransactionType;
 transactionPhase : OPTIONAL TransactionPhaseType;
 previous : OPTIONAL SET [0:?] OF MessageInTransactionType;
END_ENTITY;
```

Systematiek Deel 2 behoeft geen aanpassingen.

### 1.2.2 Hergebruik en blokkeren gegevenselementen

In systematiek versie 1.1 werd nog geen ondersteuning geboden voor het hergebruik van waarden van 'Simple Elements' uit opvolgende berichten. Berichten moesten hierdoor, als in de VISI-compatible software hier geen functionaliteit voor is ontwikkeld, opnieuw worden ingevuld terwijl de waarden al in het voorgaande bericht stonden. Het kon ook voorkomen dat waarden dienen te worden overgenomen uit het voorgaande bericht, maar niet aangepast mogen worden. Vanaf versie 1.2 is het mogelijk om waarden van 'simple elements' uit opvolgende berichten te hergebruiken.

#### 1.2.2.1 Doel

Op dit moment moeten bij elk bericht velden opnieuw worden ingevuld. In veel gevallen blijkt echter dat dit niet nodig is daar de informatie al aanwezig was in voorgaande berichten, of zelfs niet gewenst omdat waarden uit vorige berichten gebruikt moeten worden en niet aangepast mogen worden. Dit document beschrijft hoe de Systematiek kan worden uitgebreid om dit mogelijk te maken.

#### 1.2.2.2 Waarden en aanpasbaarheid van SimpleElements en ComplexElements

De regels die gelden bij het huidige raamwerk:

1. Indien het MITT van een bericht geen previous heeft zijn alle SimpleElements optioneel.
2. Indien een bericht een reactie op een ander bericht is en zijn alle ComplexElements gelijk aan de corresponderende ComplexElements uit het voorgaande bericht, dan wordt de content volledig overgenomen en is niet wijzigbaar.
3. Indien een bericht een reactie op een ander bericht is zijn alle SimpleElements die in het voorgaande bericht voorkomen automatisch gevuld met de waarde van het SimpleElement in het voorgaande bericht en niet aanpasbaar (SimpleElements uit het voorgaande bericht welke vallen onder ComplexElements die ook voorkomen in het nieuwe bericht worden genegeerd).
4. In geval van meerdere dezelfde SimpleElements wordt de volgorde van vulling uit het voorgaande bericht aangehouden, indien er in het nieuwe bericht meer instanties van hetzelfde SimpleElement aanwezig zijn dan in het vorige bericht wordt weer bij het eerste instantie uit het vorige bericht begonnen (zie voorbeelden voor verdere uitleg).

Bovenstaande regels hebben meer invloed op hoe te reageren op een bericht dan het in eerste instantie lijkt. Een paar belangrijke randvoorwaarden die gelden:

- Het voorgaande bericht hoeft niet noodzakelijk tot dezelfde transactie te behoren.
- De waarde van SimpleElements volgens deze regel zijn onafhankelijk (let wel dit geldt alleen voor de regels, niet voor uitzonderingen die verderop worden beschreven)
- Berichten die voor het voorgaande bericht zijn geweest hebben geen (directe) invloed.

Om de mogelijkheden duidelijk te krijgen zullen een aantal voorbeelden worden uitgewerkt.

***Uitzondering***

In sommige gevallen zullen bovenstaande regels een effect hebben dat waarden niet aangepast kunnen worden terwijl we dit wel willen, of waarden kunnen aangepast worden terwijl we dit niet willen. **ElementCondition** kan bovenstaande regels overrulen.

In onze voorbeelden zullen we uitgaan van het volgende ‘voorgaande’ bericht:

```
<VoorgaandBericht id="ID_0000">
        ...
        <ceVoorbeeldX>
                <CeVoorbeeldX id="ID_0100">
                        <a>0</a>
                        <b>1</b>
                        <c>2</c>
                        <c>3</c>
                </CeVoorbeeldX>
        </ceVoorbeeldX>
        <ceVoorbeeldY>
                <CeVoorbeeldY id="ID_0110">
                        <d>4</d>
                        <b>5</b>
                        <b>6</b>
                        <e>7</e>
                </CeVoorbeeldY>
                <CeVoorbeeldY id="ID_0111">
                        <d>8</d>
                        <b>9</b>
                        <b>10</b>
                        <e>11</e>
                </CeVoorbeeldY>
        </ceVoorbeeldY>
</VoorgaandBericht>
```

Stel dat we een bericht aanmaken met dezelfde ComplexElements, maar andere aantallen.

```
<BerichtEen id="ID_0001">
        ...
        <ceVoorbeeldX>
                <CeVoorbeeldX id="ID_1100">
                        <a>0</a>
                        <b>1</b>
                        <c>2</c>
                        <c>3</c>
                </CeVoorbeeldX>
                <CeVoorbeeldX id="ID_1101">
                        <a></a>
                        <b></b>
                        <c></c>
                        <c></c>
                </CeVoorbeeldX>
        </ceVoorbeeldX>
        <ceVoorbeeldY>
                <CeVoorbeeldY id="ID_1112">
                        <d>4</d>
                        <b>5</b>
                        <b>6</b>
                        <e>7</e>
                </CeVoorbeeldY>
        </ceVoorbeeldY>
</BerichtEen>
```

Daar alle ComplexElements ook in het voorgaande bericht zaten zijn de ComplexElements niet aan te passen met uitzondering van de elementen in het ComplexElement “ID_1101”.

Stel dat we een bericht aanmaken met andere ComplexElements, maar dezelfde SimpleElements binnen de ComplexElements.

```
<BerichtEen id="ID_0002">
        ...
        <ceVoorbeeldXadj>
                <CeVoorbeeldYadj id="ID_2120">
                        <a>0</a>
                        <b>1</b>
                        <c>2</c>
                        <c>3</c>
                </CeVoorbeeldXadj>
        </ceVoorbeeldXadj>
        <ceVoorbeeldYadj>
                <CeVoorbeeldYadj id="ID_2130">
                        <a></a>
                        <b>5</b>
                        <c>2</c>
                        <c>3</c>
                </CeVoorbeeldYadj>
                <CeVoorbeeldYadj id="ID_2131">
                        <d>4</d>
                        <b>6</b>
                        <b>9</b>
                        <e>7</e>
                </CeVoorbeeldYadj>
        </ceVoorbeeldYadj>
</BerichtEen>
```

Daar alle SimpleElements ook in het voorgaande bericht zaten zijn de SimpleElements niet aan te passen.

Typische eigenschappen:

- a en c komen meer keren voor dan in het voorgaande bericht, we starten bij het vullen van c steeds weer opnieuw, dit gebeurt tevens bij a alhoewel deze slechts één waarde heeft.
- b komt minder vaak voor dan in het voorgaande bericht, we komen hier dus niet volledig uit met de telling. Wat hier belangrijk is dat we de volgorde van het voorgaande bericht aanhouden zonder dat we kijken naar ComplexElements.

Stel dat we een bericht aanmaken met andere ComplexElements.

```
<BerichtTwee id="ID_0003">
        ...
        <ceVoorbeeldZ>
                <CeVoorbeeldZ id="ID_3140">
                        <h>1000</h>
                        <b>1</b>
                        <a>0</a>
                        <a>0</a>
                        <b>5</b>
                        <e>7</e>
                        <i>1001</i>
                        <e>11</e>
                        <e>7</e>
                </CeVoorbeeldZ>
        </ceVoorbeeldZ>
</BerichtTwee>
```

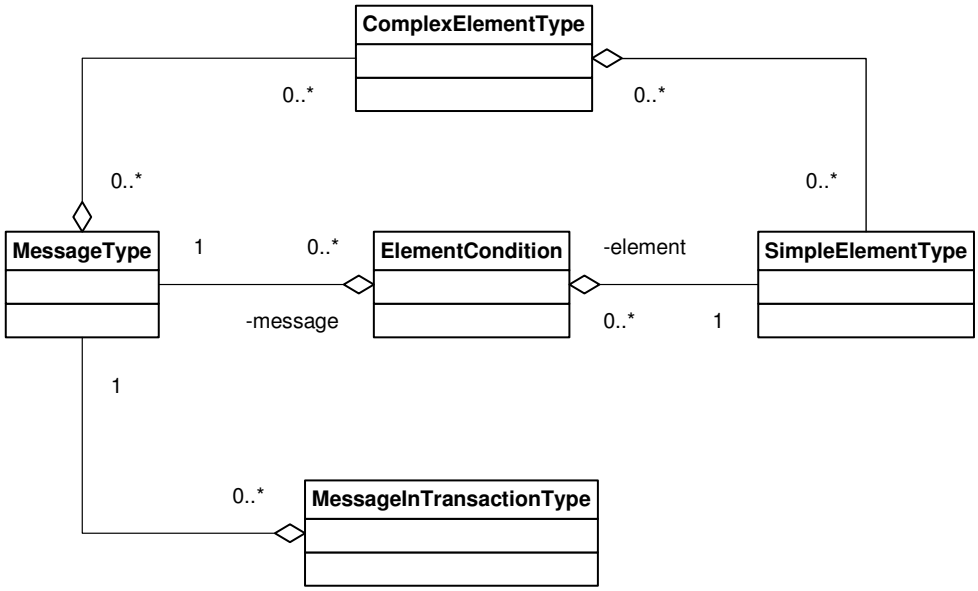
Daar niet alle SimpleElements in het voorgaande bericht zaten zijn de SimpleElements h en i aan te passen.

Typische eigenschappen:

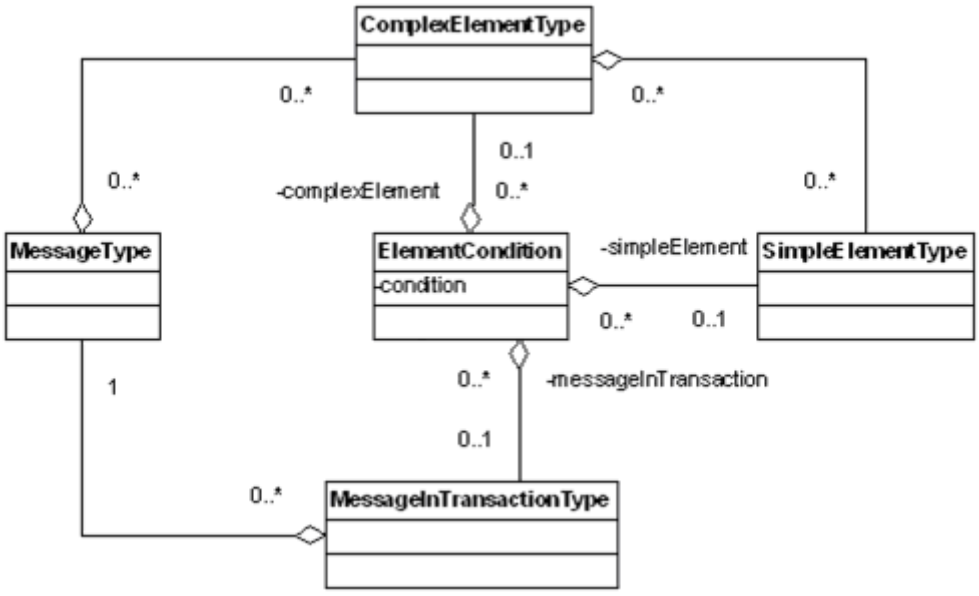
- a en e komen meer keren voor dan in het voorgaande bericht, we starten bij het vullen van e steeds weer opnieuw, dit gebeurd tevens bij a alhoewel deze slechts één waarde heeft.

**Aanpassingen aan systematiek Deel 1**

Het deel van ElementCondition uit de oorspronkelijke versie:




De nieuwe voorgestelde aanpassing van ElementCondition:




Beschrijving van de nieuwe relaties:
*ElementCondition* : Hiermee kan de conditie (hierboven genoemde regels) van SimpleElements binnen een MITT (een BerichtType in de context van een TransactieType) worden overruled .

*condition* : enumeratie van

| | |
|---|---|
| FREE | : in alle gevallen zal het SimpleElement vrij invulbaar zijn |
| FIXED | : in alle gevallen zal het SimpleElement gefixeerd zijn |
| EMPTY | : als FREE, maar nu zonder een waarde in te vullen |

*simpleElement* : het SimpleElement welke in deze context bedoeld wordt (optionele link).
*complexElement* : het ComplexElement welke in deze context bedoeld wordt (optionele link).

*messageInTransaction* : de context (het bericht) waarvoor deze overrule conditie geldt (optionele link). Indien deze link niet is gebruikt geldt deze element conditie voor elke MessageInTransaction waar dit simple/complex element wordt gebruikt.

**Systematiek Deel 1 behoeft een aanpassing in ElementCondition**

*Oorspronkelijke versie*

```
ENTITY ElementCondition;
 description : STRING;
 requiredNotify : INTEGER;
 minValue : OPTIONAL STRING;
 maxValue : OPTIONAL STRING;
 format : OPTIONAL STRING;
 helpInfo : OPTIONAL STRING;
 element : SimpleElementType;
 message : MessageType;
END_ENTITY;
```

*Voorgestelde aanpassing*

```
ENTITY ElementCondition;
 description : STRING;
 condition : STRING;
 helpInfo : OPTIONAL STRING;
 complexElement : OPTIONAL ComplexElementType;
 simpleElement : OPTIONAL SimpleElementType;
 messageInTransaction : OPTIONAL MessageInTransactionType;
END_ENTITY;
```

Systematiek Deel 2 behoeft geen aanpassingen.

### 1.2.3 DateTime format

Vanaf systematiek 1.2 wordt (in Systematiek Deel 2) het DateTime format gebruikt in plaats van het Date format zodat naast de datum ook de tijd kan worden geregistreerd. Het DATETIME datatype refereert aan de "xsd:dateTime" definitie (Instant of time (Gregorian calendar)).

### 1.2.4 Successor (zie ook 1.1.9 oneindige loops)

Systematiek 1.2 (Deel 2) is aangepast om de functionaliteit van de mogelijkheid dat tijdens een project een persoon kan worden opgevolgd door een ander persoon te ondersteunen. Hierbij wordt rekening gehouden met openstaande instanties van transacties (PSB).

### 1.2.5 Toevoeging MITT aan bericht

Binnen transacties was het mogelijk om hetzelfde bericht meerdere malen op verschillende plekken in de flow van de transactie op te nemen. Hierdoor waren gevallen denkbaar waarbij hetzelfde bericht binnen een transactie andere vervolgberichten heeft, afhankelijk van de positie in de flow. Om dit te voorkomen is in Systematiek 1.2 het MessageInTransactionType in het bericht vastgelegd.

***Achterliggende reden***
Binnen transacties is het mogelijke meerdere dezelfde berichten op verschillende plaatsen in een workflow op te nemen. Hierdoor zijn gevallen denkbaar waarbij hetzelfde bericht binnen een transactie andere vervolgberichten heeft afhankelijk van zijn positie in de workflow.
Afgesproken is dit probleem op te lossen door de bijbehorende MessageInTransactionType mee te nemen in het bericht. Hierdoor is de positie in de workflow van een transactie altijd eenduidig te achterhalen.

***Voorbeeld***
Er verandert niets aan een raamwerk. Slechts de berichten worden uitgebreid.
Stel we hebben de volgende constructie in een raamwerk:

```
<MessageType id="Bericht_001">
        ...
</MessageType>
...
<MessageInTransactionType id="BerichtInTransactie_001">
        ...
        <message>
                <MessageTypeRef idref="Bericht_001"/>
        </message>
        <transaction>
                <TransactionTypeRef idref="Transactie_001"/>
        </transaction>
        ...
</MessageInTransactionType>
...
<TransactionType id="Transactie_001">
        ...
</TransactionType>
```

In een bericht vinden we dan oorspronkelijk de volgende constructie terug:

```
<Bericht_001 id="a001">
        <identification>...</identification>
        <dateSend>...</dateSend>
        <dateRead>...</dateRead>
        <state>...</state>
        <dateLamu>...</dateLamu>
        <userLamu>...</userLamu>
        <initiatorToExecutor>...</initiatorToExecutor>
        <transaction>
                <Transactie_001Ref idref="a002"/>
        </transaction>
        ...
</Bericht_001>
<Transactie_001 id="a002">
        <name>...</name>
        <description>...</description>
        <startDate>...</startDate>
        <endDate>...</endDate>
        <state>...</state>
        <dateLamu>...</dateLamu>
        <userLamu>...</userLamu>
        <initiator>
                <PersonInRoleRef idref="..."/>
        </initiator>
        <executor>
                <PersonInRoleRef idref="..."/>
        </executor>
</Transactie_001>
```

Na de aanpassing vinden we terug:

```
<Bericht_001 id="a001">
        <identification>...</identification>
        <dateSend>...</dateSend>
        <dateRead>...</dateRead>
        <state>...</state>
        <dateLamu>...</dateLamu>
        <userLamu>...</userLamu>
        <initiatorToExecutor>...</initiatorToExecutor>
        <messageInTransaction>
                <BerichtInTransactie_001Ref idref="a003"/>
        </messageInTransaction>
        <transaction>
                <Transactie_001Ref idref="a002"/>
        </transaction>
        ...
</Bericht_001>
<BerichtInTransactie_001 id="a003">
        <name>...</name>
        <description>...</description>
        <startDate>...</startDate>
        <endDate>...</endDate>
```

```
            <state>...</state>
            <dateLamu>...</dateLamu>
            <userLamu>...</userLamu>
        </BerichtInTransactie_001>
        <Transactie_001 id="a002">
            <name>...</name>
            <description>...</description>
            <startDate>...</startDate>
            <endDate>...</endDate>
            <state>...</state>
            <dateLamu>...</dateLamu>
            <userLamu>...</userLamu>
            <initiator>
                <PersonInRoleRef idref="..."/>
            </initiator>
            <executor>
                <PersonInRoleRef idref="..."/>
            </executor>
        </Transactie_001>
```

***Aanpassingen Systematiek***
Alleen Systematiek Deel 2 hoeft te worden aangepast.
Het betreft hier een uitbreiding van de Systematiek met een nieuw concept te weten MessageInTransactionTemplate. In principe is hiermee de TransactionTemplate overbodig geworden, deze wordt echter om eenvoudigheidsredenen en potentiële problemen te voorkomen niet verwijderd.

Oorspronkelijke invulling:

```
ENTITY MessageTemplate;
 identification : STRING;
 dateSend : DATETIME;
 dateRead : DATETIME;
 state : STRING;
 dateLamu : DATETIME;
 userLamu : STRING;
 initiatingTransactionMessageID : OPTIONAL STRING;
 initiatorToExecutor : BOOLEAN;
 transaction : TransactionTemplate;
 template : ComplexElementTemplate;
END_ENTITY;
```

Nieuwe invulling:

```
ENTITY MessageTemplate;
 identification : STRING;
 dateSend : DATETIME;
 dateRead : DATETIME;
 state : STRING;
 dateLamu : DATETIME;
 userLamu : STRING;
 initiatingTransactionMessageID : OPTIONAL STRING;
 initiatorToExecutor : BOOLEAN;
 messageInTransaction : MessageInTransactionTemplate;
 transaction : TransactionTemplate;
 template : ComplexElementTemplate;
```

```
END_ENTITY;

ENTITY MessageInTransactionTemplate;
 identification : STRING;
 dateSend : DATETIME;
 dateRead : DATETIME;
 state : STRING;
 dateLamu : DATETIME;
 userLamu : STRING;
END_ENTITY;
```

# 2 Frequently Asked Questions

Hieronder volgt een aantal van de – voor implementatie van VISI – belangrijkste, veel gestelde vragen (‘frequently asked questions’). De volledige (en meest recente) lijst van FAQ’s is te raadplegen op de VISI website.

## *1. Wat is het eerste MessageType in een TransactionType?*

Het eerste MessageType is het MessageType dat binnen een TransactionType geen voorgaand bericht heeft.

## *2. Wat is het laatste MessageType in een TransactionType?*

Het laatste MessageType binnen een transactionType is het MessageType waaraan niet door andere MessageTypes wordt gerefereerd.

## *3. Indien een bericht verstuurd is in een transacties, is het dan toegestaan om nog een bericht te versturen?*

Binnen een transactie is altijd één rol aan zet. Het is wel mogelijk berichten uit een andere transactie te versturen (ook als transacties aan elkaar gerelateerd zijn). Dus als een bericht wordt verzonden in een secundaire transactie maar er is ook nog een MITT in de hoofdtransactie die op basis van een voorgaande MITT verzonden mag worden, dan is dit gewoon mogelijk.

## *4. Hoe wordt met het DATETIME datatype omgegaan op het moment dat geen tijdzone is gespecificeerd?*

Het DATETIME datatype refereert aan de "xsd:dateTime" definitie (Instant of time (Gregorian calendar)). We passen echter één afwijking hierop toe: Als geen tijdzone is gespecificeerd wordt UTC[2] (=Z of +00:00) verondersteld.

- The DATETIME datatype describes instances identified by the combination of a date and a time. Its value space is described as a combination of date and time of day in Chapter 5.4 of ISO 8601. Its lexical space is the extended format:
  [-]CCYY-MM-DDThh:mm:ss[Z|(+|-)hh:mm]

- The time zone may be specified as Z (UTC) or (+|-)hh:mm. Time zones that aren't specified are considered undetermined.

- Example
  Valid values for xsd:dateTime include: 2001-10-26T21:32:52, 2001-10-26T21:32:52+02:00, 2001-10-26T19:32:52Z, 2001-10-26T19:32:52+00:00, -2001-10-26T21:32:52, or 2001-10-26T21:32:52.12679.
  The following values are invalid: 2001-10-26 (all the parts must be specified), 2001-10-

[2] UTC is bijna gelijk aan Greenwich Mean Time (GMT). GMT is echter een zuiver astronomische tijd. Om het door de vertraagde aardrotatie veroorzaakte verschil te compenseren, moeten er schrikkelsecondes worden gebruikt. Het verschil is nooit meer dan een seconde en voor de meeste toepassingen dan ook niet van belang.

26T21:32 (all the parts must be specified), 2001-10-26T25:32:52+02:00 (the hours part—25—is out of range), or 01-10-26T21:32 (all the parts must be specified).

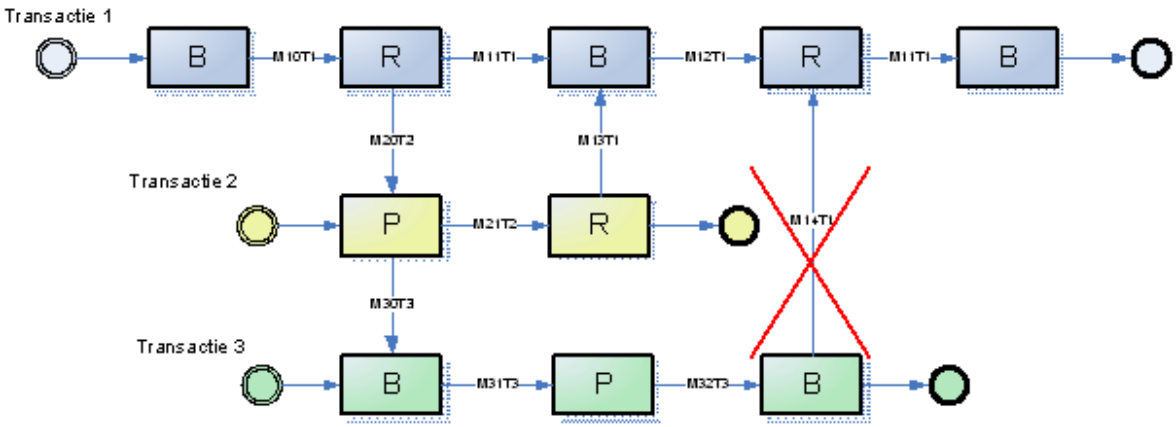
## 5. Als ik een raamwerk heb gepromoot en een berichtenschema gegenereerd, heb ik dan de garantie dat dit raamwerk in de praktijk werkt?

Het promoten van een raamwerk garandeert alleen dat het raamwerk aan de systematiek voldoet en ingelezen kan worden in een VISI compatible software. Dit garandeert niet dat een raamwerk in de praktijk bruikbaar is. Als de raamwerkbouwer bijvoorbeeld geen complex element heeft toegevoegd aan een bericht, dan kan dit in de praktijk in het algemeen geen gewenste situatie opleveren. Dit neemt niet weg dat het raamwerk volgens de systematiek valide is.

## 6. Als ik transacties aan elkaar relateer is de traceerbaarheid van communicatie altijd gegarandeerd?

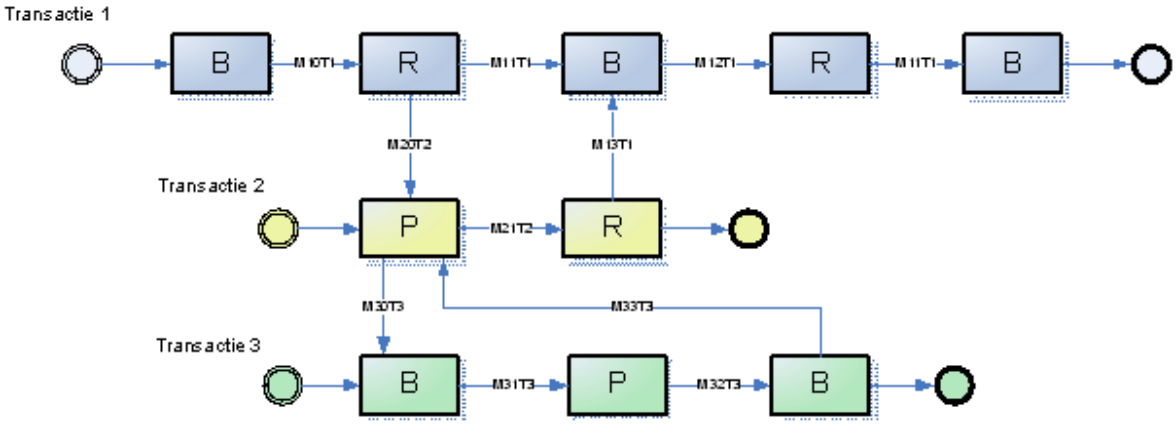
Zolang de VISI filosofie wordt gevolgd, waarin transacties altijd tussen twee rollen (initiator en executor) verlopen en de initiator altijd een bericht terug krijgt van de executor, blijft de communicatie traceerbaar. In uitzonderlijke situaties is het echter mogelijk om raamwerken zo op te zetten dat de traceerbaarheid in gevaar kan komen. Om dergelijke situaties te voorkomen wordt hieronder een voorbeeld gegeven van een dergelijke uitzondering en vervolgens wordt geïllustreerd hoe het volgens de VISI filosofie hoort te zijn. Indien meer van dit soort situaties worden gesignaleerd, dan zullen deze worden toegevoegd aan deze FAQ.

De VISI systematiek gaat ervan uit dat een transactie altijd tussen twee rollen plaatsvindt. Bij het opstellen van een raamwerk waar de communicatie over meerdere transacties verloopt dient hier rekening mee gehouden te worden. Daarnaast hoort een proces dat over bijvoorbeeld drie transacties verloopt ook weer over dezelfde transacties terug te verlopen (belangrijk om communicatie ook in een later stadium traceerbaar te houden). Het onderstaande voorbeeld is fout, het is niet verstandig een raamwerk op een dergelijke wijze op te zetten.




**Figuur 4**

De opzet hieronder is juist, de communicatie blijft traceerbaar.



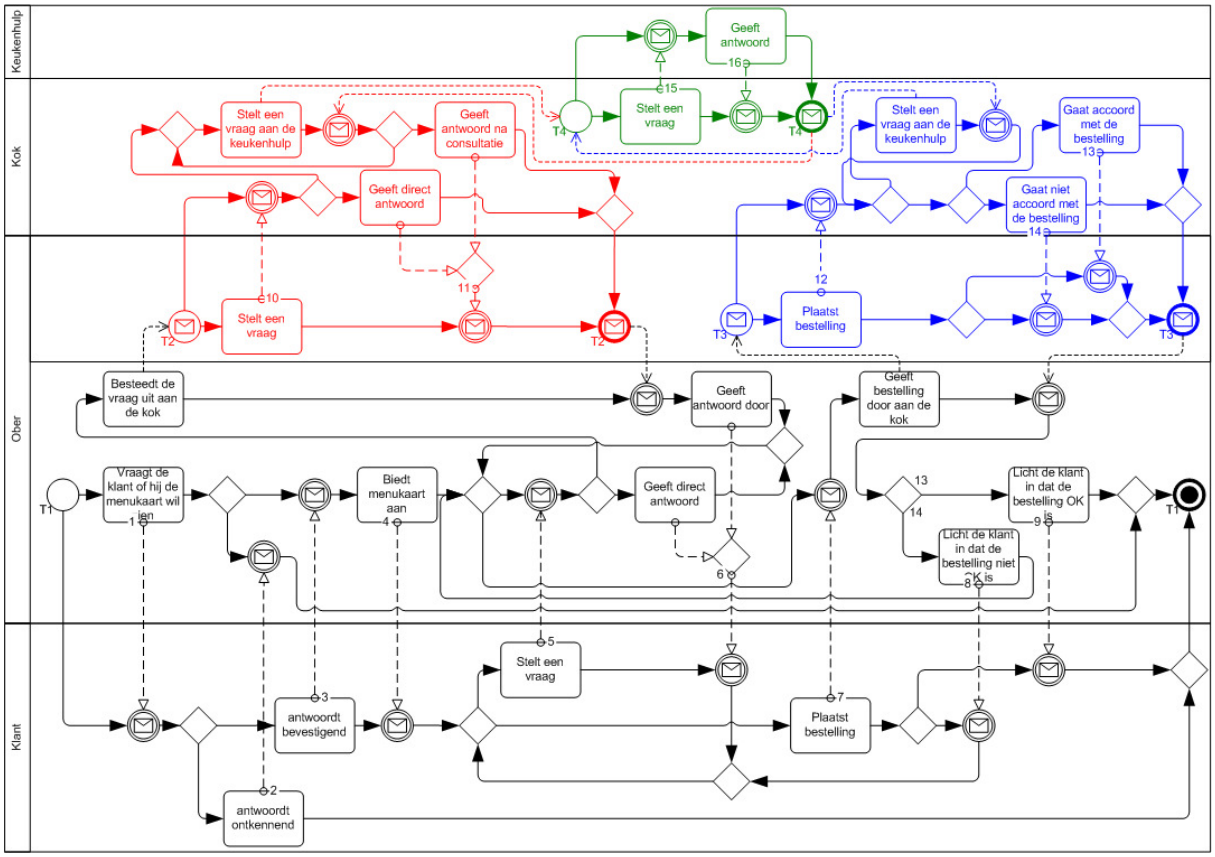

**Figuur 5**

Let bij het interpreteren van de schema's op het volgende. De cirkeltjes aan het begin en het eind geven respectievelijk het begin en het einde van een transactie aan. De blokjes zijn rollen en de letters in de blokjes staan voor een specifieke rol. De codes in de pijlen staan voor MessageTypes. Voor de duidelijkheid is ook de transactie waar de MessageType toe behoort in de code verwerkt. M31T3 staat dus voor MessageType 31 uit transactie 3.

## *7. Hoe geeft ik in een raamwerk aan of een bericht een startbericht is, of niet?*

Sinds VISI Standaard versie 1.2 kan in een raamwerk een transactie in cascadevorm aan een andere transactie worden gekoppeld. Deze koppeling vindt plaats op berichtniveau. Het startbericht van een subtransactie heeft het bericht uit de hoofdtransactie dan als voorgaande (='previous'). Bovenstaande definitie conflicteert echter met de ongeschreven regel dat een bericht zonder voorgaand bericht (='previous') een startbericht van een transactie is. Welk bericht het startbericht van een subtransactie is, is niet eenduidig vastgelegd en is voor bepaalde raamwerken niet te achterhalen.

In de praktijk is gebleken dat dit bijvoorbeeld onduidelijke situaties oplevert indien een transactie een subtransactie is van meer dan 1 transactie en op één enkel punt naar 2 of meer hoofdtransacties kan terugkeren. De groengekleurde transactie in figuur 6 hieronder voldoet aan deze voorwaarden.




**Figuur 6**

Indien in de bovenstaande figuur de communicatie vanaf T1 (zwart) via T2 (rood) naar T4 (groen) is gelopen, en het laatste bericht van T4 (groen) verstuurd is, is er de keuze om terug te gaan naar T2 (rood) of vooruit te gaan en een nieuwe subtransactie T3 (blauw) te starten.
In dit voorbeeld is het voor een mens visueel duidelijk te zien dat dit in dit geval niet de bedoeling is, maar voor software is dit niet mogelijk.

Om dit aan software wel duidelijk te maken, moet er een toevoeging in het raamwerk komen om aan te geven dat bepaalde berichten wel of niet een startbericht zijn. In het raamwerk moet dan komen te staan dat T3 (blauw) alleen kan worden gestart vanaf T1 (zwart) en NIET vanaf T4 (groen).

Daartoe zijn de ‘Referenties’ van de entiteit ‘MessageInTransactionType’ in de VISI Systematiek Deel 1 (voorheen Systematiek I) uitgebreid met de eigenschap ‘firstMessage’. Daarmee kan worden aangegeven of een MessageInTransaction een startbericht betreft.

```
ENTITY MessageInTransactionType;
  requiredNotify : INTEGER;
  dateLamu   : DATETIME;
  userLamu   : STRING;
  received   : BOOLEAN;
  send       : BOOLEAN;
  state      : STRING;
  initiatorToExecutor : OPTIONAL BOOLEAN;
  openSecondaryTransactionsAllowed : OPTIONAL BOOLEAN;
  firstMessage      : OPTIONAL BOOLEAN;
  message           : MessageType;
  previous          : OPTIONAL SET [0:?] OF MessageInTransactionType;
  transaction       : TransactionType;
  transactionPhase  : OPTIONAL TransactionPhaseType;
```

```
  group             : GroupType;
END_ENTITY;
```

Deze nieuwe eigenschap is een optionele Boolean (‘True’, of ‘False’). Indien deze Boolean afwezig is, geldt de standaard waarde ‘False’ en kan er met de betreffende MessageInTransactionType GEEN nieuwe transactie worden gestart.
Wanneer deze Boolean aanwezig is en de waarde ‘True’ heeft, kan er met de MessageInTransactionType wel een nieuwe transactie worden gestart.
In het voorbeeld hierboven dient voor de startberichten van T1, T2, T3 en T4 deze Boolean dus de waarde ‘True’ te hebben.
Mocht dit per ongeluk niet zijn ingesteld voor T1 dan mag T1 toch worden gestart, omdat het startbericht van T1 geen ‘previous’ heeft. Deze voorwaarde prevaleert altijd t.o.v. de Boolean.

< einde Bijlage 5 >