Software Improvement Group

Analyse van de StUF-BG standaard

In opdracht van de gemeente Den Haag

25 september 2015

GETTING SOFTWARE RIGHT

Colofon

Niels van der Zwan & Liam Blythe
+31 6 5333 6051
n.vanderzwan@sig.eu

PUBLIC

Inhoudsopgave

Samenvatting 5

1 Context 7
1.1 De StUF standaard 7
1.2 De StUF-BG standaard 8
1.3 De Haagse Situatie 8
1.4 Eigenschappen standaard 9

2 Opdrachtbeschrijving 10
2.1 Onderzoeksvragen 10
2.2 Uitgangspunten SIG 10

3 Toetsingskader en aanpak 11
3.1 Beoordelingscriteria, begripsbepaling en operationalisatie 11
3.2 Gehanteerd proces 12
3.3 Planning 12

4 Procesbeschrijving 13
4.1 Uitvoering en verloop van het proces 13
4.2 Bijeenkomsten 13

5 Validatie van de claims 15
5.1 Claims door gemeente Den Haag 15
5.2 Aanpak validatie 16
5.3 Claim 1: De standaard tooling werkt niet met de geleverde contracten 17
5.4 Claim 2: De SOAP binding van StUF-BG is niet specifiek genoeg 25
5.5 Claim 3: De betekenis van algemene kennisgevingsberichten is afhankelijk van de context 26
5.6 Claim 4: StUF-BG kan alleen gedeeltelijk geïmplementeerd worden door de gemeente Den Haag. Dit vereist extra afstemming met afnemers 27
5.7 Claim 5: Implementatie van StUF-BG levert performance en beschikbaarheidsrisico's op 27

6 Bevindingen 29
6.1 Implementatie van StUF-BG vereist extra inspanning 29
6.2 Op StUF-BG gebaseerde koppelingen zijn niet direct herbruikbaar 31
6.3 Gebruik van de StUF-BG standaard vereist een aanzienlijke aanloop 31
6.4 Mogelijk gebruik van StUF-BG bij nieuwe architectuurstijlen en formaten 32

7 Conclusies 34
7.1 Interoperabiliteit 34
7.2 Kostenreductie 34
7.3 Bevorderen marktwerking 35
7.4 Bevorderen innovatie 35

8 Aanbevelingen 37
8.1 Aanbevelingen voor de Gemeente Den Haag 37
8.2 Aanbevelingen die de Gemeente Den Haag aan KING kan doen 37

A Bronnenoverzicht 42
A.1 Publieke bronnen 42
A.2 Ontvangen van klant: 42

B McCabe Cyclomatic Complexity 43
B.1 McCabe Cyclomatic Complexiteit in XML Schema's 43

C Benchmark XML standaarden 44
C.1 Standaarden 44

D Serializeren van *nil* elementen in Java en .NET 45
D.1 Serializeren met JAXB 45
D.2 Serializeren met .NET XMLSerializer 45

E Voorbeeld: StUF-BG met REST koppelvlak 47
E.1 Koppelvlak definities 47
E.2 Ophalen van resource 47

F Voorbeeld: StUF-BG met JSON formaat 48
F.1 Voorbeeldbericht NPS bevraging 48

G KING feedback op onderzoek SIG 49

Samenvatting

In opdracht van de gemeente Den Haag heeft SIG een analyse gedaan van de StUF-BG standaard. De gemeente Den Haag wil weten of het gebruik van de standaard StUF-BG haar doelstellingen op het gebied van interoperabiliteit, kostenreductie, marktwerking en innovatie voldoende ondersteunt. De gemeente Den Haag is bezig met het implementeren van een nieuwe applicatie (SBS) voor gemeentelijke basisgegevens, waarbij de interfaces o.b.v. de StUF-BG standaard moeten worden gebouwd door de afnemers. StUF-BG is een onderdeel van StUF, een standaard voor gegevensuitwisseling binnen de Nederlandse overheid, die bestaat sinds 1997. Ontwikkeling en beheer van de StUF standaard is belegd bij het Kwaliteitsinstituut Nederlandse Gemeenten (KING). SIG heeft in dit onderzoek met name de bruikbaarheid van de StUF-BG standaard bekeken voor ontwikkelaars die o.b.v. de StUF-BG standaard software schrijven voor koppelingen tussen/met gemeentelijke applicaties.

SIG concludeert dat het verplicht gebruik van de StUF-BG standaard in zijn huidige vorm slechts een beperkte garantie geeft voor interoperabiliteit van de interfaces tussen gemeentelijke systemen. Om een interface te bouwen op basis van de complete StUF-BG standaard is extra afstemming en werk noodzakelijk omdat de werkelijk ondersteunde operaties en berichten niet expliciet maar impliciet vast liggen in de contracten.
Bij een implementatie op de door KING bedoelde wijze van een StUF-BG interface ontstaat er iedere keer een nieuwe op StUF-BG gebaseerde sub-standaard. Hierdoor zal bij aansluiting van een nieuwe partij moeten worden vastgesteld of er voldoende overlap is tussen de door afnemer en provider gedefinieerde sub-standaarden. Dit zal niet kostenverlagend werken en de marktwerking en innovatie niet vergroten.

Om de bruikbaarheid van StUF-BG te borgen adviseren wij de gemeente Den Haag er bij KING op aan te dringen regie te nemen bij het vaststellen en delen van sub-standaarden en koppelvlakken o.b.v. StUF-BG. Deze koppelvlakken zijn de eigenlijke standaard interfaces, waarmee de gemeenten en leveranciers van gemeentelijk software moeten werken, zodat er daadwerkelijk sprake is van een standaard stekker en stopcontact.

Om er voor te zorgen dat de inleertijd voor ontwikkelaars wordt verkleind is het belangrijk de complexiteit van het gebruik van de StUF-BG standaard te verkleinen d.m.v. een library, die ontwikkelaars eenvoudig kunnen gebruiken binnen hun ontwikkelomgevingen. Het gebruik van deze library moet laagdrempelig zijn, de library moet publiekelijk beschikbaar zijn en minimaal de meest gebruikte programmeertalen (C# en Java) en hun ontwikkelomgevingen ondersteunen. Regie en beheer over de library moeten bij één partij komen te liggen.
Daarnaast is het wenselijk om de minder gangbare constructies, die veelvuldig gebruikt worden in StUF-BG, te vervangen door constructies, die in ieder geval in Java en C# eenvoudig tot goed werkende software leiden.

Bij het verder implementeren van SBS door de gemeente Den Haag adviseren wij de gemeente de leverancier van SBS te vragen een koppelvlak te definiëren o.b.v. StUF-BG met de werkelijk ondersteunde operaties en berichten. De contracten (WSDL's en XSD's) van het koppelvlak moeten worden meegeleverd met het pakket. Dit koppelvlak kan worden gebruikt door de afnemers van de SBS gegevens, zodat ze eenvoudiger hun interfaces kunnen schrijven en gebruiken. In dit

nieuwe koppelvlak kunnen een aantal beschikbaarheids- en performancerisico's worden gemitigeerd door het aantal ondersteunde zoekpaden te beperken.

Claims, aanpak en bevindingen

SIG heeft interviews met ontwikkelaars en gebruikers van de StUF-BG standaard uitgevoerd en hieruit de claims van de gemeente Den Haag geformuleerd. In deze claims komt met name de omvang en complexiteit van de standaard naar voren en het feit dat er voor ontwikkelaars onduidelijkheden zijn in de standaard, die leiden tot mogelijke interpretatieverschillen en daarmee tot extra afstemming en extra werk bij het gebruik van StUF-BG. Het ondersteunen van de gehele standaard is daardoor naar de mening van de gemeente Den Haag niet mogelijk. Het gebruik van de gehele standaard leidt tot beschikbaarheids- en performance risico's.

Tijdens het onderzoek heeft SIG deze claims bestudeerd door middel van (1) vergelijking van StUF-BG met andere open XML standaarden, (2) bestudering van publiek beschikbare documentatie en (3) door StUF-BG te gebruiken in standaard C# en Java programmeeromgevingen. In het uitgevoerde onderzoek is gebleken dat de StUF-BG standaard groot en complex is in vergelijk met de andere standaarden in de vergelijking. Deze complexiteit ontstaat mede doordat StUF gebruik maakt van constructies die vrijwel niet worden gebruikt in de andere standaarden.
StUF-BG moet worden beschouwd als een halffabricaat, dat niet in z'n geheel kan worden gebruikt bij implementatie van interfaces tussen overheidssystemen (dit is bij SBS wel gebeurd). Om tot hanteerbare interfaces (kleiner, minder complex en bruikbaar binnen gangbare ontwikkelomgevingen) te komen, moeten eerst een aantal handmatige stappen worden uitgevoerd. Deze werkwijze en de te volgen stappen worden, voor zover wij hebben kunnen constateren, nergens expliciet beschreven en toegelicht in de beschikbare bronnen. Het gebruik van de StUF-BG standaard op de door KING bedoelde wijze vereist in de huidige situatie een aanzienlijke aanloop met een grote inspanning, mede door de complexiteit van de standaard en de grote hoeveelheid in documenten vastgelegde richtlijnen.

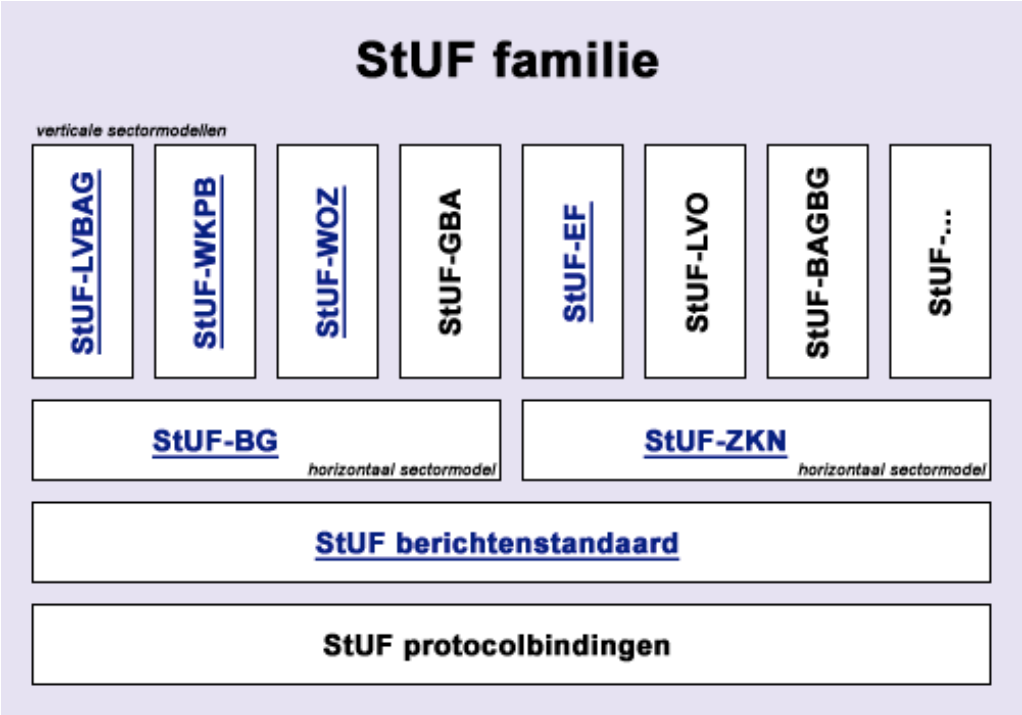
Elk van StUF-BG afgeleid koppelvlak is in feite een nieuwe sub-standaard. Op dit moment is er geen centraal beheer en sturing (door KING) op de afgeleide sub-standaarden. De huidige situatie bij SBS kan worden doorbroken door een op StUF-BG gebaseerd koppelvlak, inclusief de (nieuwe) technische contracten (WSDL's en XSD's), aan de ontwikkelaars van gemeentelijke software beschikbaar te stellen, zodat de gewenste interface software kan worden geschreven, gebruik makend van de beschikbare programmeeromgevingen en tooling.

De opzet van StUF-BG is naar onze inzichten zodanig dat er geen belemmeringen zijn om delen van StUF-BG te gebruiken voor het ondersteunen van nieuwe architectuurstijlen en formaten (bijv. REST en JSON). Tot op heden is dit echter niet gebeurd. Om dit in de toekomst mogelijk te maken is regie (van KING) noodzakelijk, zodat eenduidige en bruikbare nieuwe koppelvlakken ontstaan, die publiek beschikbaar zijn en door implementerende partijen kunnen worden gebruikt.

1 Context

1.1 De StUF standaard

StUF is een standaard voor gegevensuitwisseling binnen de Nederlandse overheid en bestaat sinds 1996. De actuele versie van StUF is 3.01. StUF wordt gebruikt door veel partijen en is een paraplu voor een aantal, meer specifieke producten (interface definities), zoals hieronder geïllustreerd.



Figuur 1.1: Overzicht van de StUF familie

1.1.1 Sectormodellen in StUF

De StUF berichtenstandaard dient als basis voor de horizontale sectormodellen StUF-BG (basisgegevens) en StUF-ZKN (zaken). Dit worden horizontale sectormodellen genoemd omdat zij in vrijwel alle domeinen een rol spelen. Andere sectormodellen worden verticale sectormodellen genoemd.

1.1.2 Technische structurering van StUF

De gehele StUF familie wordt vastgelegd in XML Schema's (XSD's). Voor de berichtenuitwisseling wordt een set protocolbindingen beschreven. De beschreven protocolbindingen van StUF zijn gebaseerd op SOAP (Simple Object Access Protocol). StUF en haar sectormodellen worden geleverd met SOAP interface contracten (WSDL's).

1.1.3 Ontwikkeling en beheer van StUF

Ontwikkeling en beheer van de StUF standaard zijn belegd bij het Kwaliteitsinstituut Nederlandse Gemeenten (KING). Voor het beheer van de StUF standaard is een beheerstructuur en proces opgezet door KING. Bij dit proces zijn de volgende partijen betrokken:

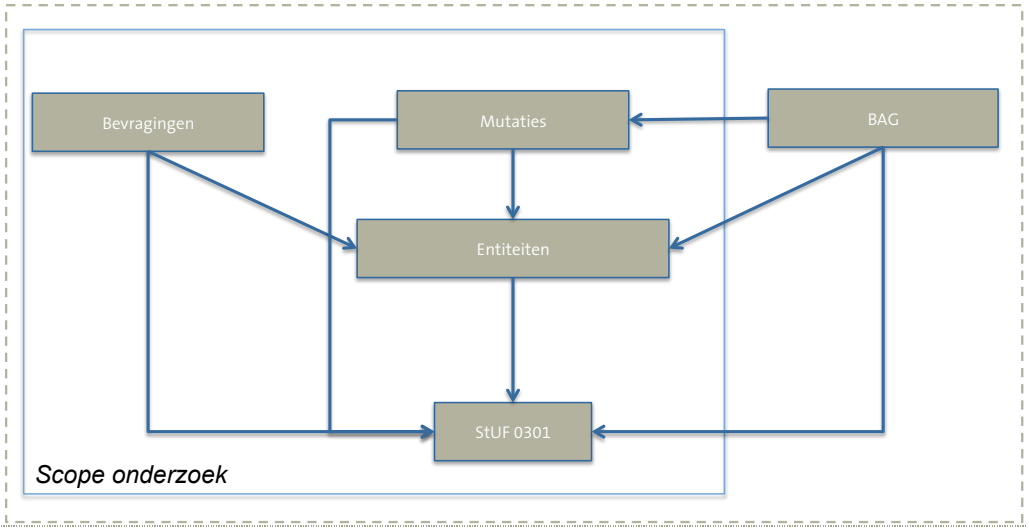
- De StUF Community maakt gebruik van een openbaar forum om vragen te stellen en te beantwoorden over de StUF standaard.

- De StUF Expertgroep richt zich op het inhoudelijk ontwikkelen van StUF en de bijbehorende documentatie, het voorbereiden van de releaseplanning en de inhoudelijke goedkeuring van nieuwe onderdelen of versies van bestaande onderdelen van de StUF familie.
- De StUF Regiegroep (Regiegroep Gegevens- en berichtenstandaarden) geeft richting en vorm aan de ontwikkelingen van de familie van gegevens- en berichtenstandaarden, waaronder StUF.

1.2 De StUF-BG standaard

StUF-BG 0310 is een horizontaal sectormodel gebouwd op de StUF 0301 berichtenstandaard en is ontwikkeld voor de uitwisseling van basisgegevens. StUF-BG 0310 is gebaseerd op het RSGB 2.0 informatiemodel en bestaat uit vier hoofdcomponenten:

- **Bevragingen:** Synchrone en asynchrone vraag/antwoord berichten.
- **Mutaties en Kennisgevingen:** Synchrone en asynchrone kennisgevings- en synchronisatie berichten van StUF-BG.
- **BAG**: Bevragingen, kennisgevings- en synchronisatieberichten voor de Basisadministratie Adressen en Gebouwen (BAG).
- **Entiteiten:** Gedeelde entiteittypen die gebruikt worden door bevragingen, mutaties en kennisgevingen en BAG.



Figuur 1.2: Scope van het onderzoek door SIG

Scope van het onderzoek zijn alle onderdelen van StUF-BG met uitzondering van BAG.

1.3 De Haagse Situatie

De gemeente Den Haag gaat in 2015 het nieuwe Stelsel Beheer Systeem (SBS) in gebruik nemen, welk recentelijk is overgenomen van de gemeente Rotterdam en coöperatief wordt doorontwikkeld. SBS is het gegevensmagazijn voor gemeentelijke basisgegevens, welk ontsloten worden middels een StUF-BG interface. Het gegevensmagazijn zal basisgegevens leveren voor meer dan 200 gemeentelijke processen, alsmede processen van ketenpartners.

Compliancy aan de gemeentelijke standaarden RSGB en StUF-BG is een belangrijke drijfveer voor de gemeenten om SBS te implementeren en door te ontwikkelen. De achterliggende motivatie hierbij is dat met het gebruik van de StUF-BG standaard een aantal doelstellingen behaald kunnen worden. Echter, ervaringen in het gebruik van de StUF-BG standaard hebben bij de gemeente Den Haag geleid tot twijfels over die haalbaarheid. Hiervoor heeft zij SIG gevraagd om de StUF-BG standaard te evalueren in relatie tot haar gestelde doelstellingen.

1.4 Eigenschappen standaard

SIG gaat er van uit dat een goede standaard voldoet aan de volgende eigenschappen:

- **Geaccepteerd**: Door een autoriteit en/of een groep belanghebbenden
- **Gangbaar**: Het gebruik wordt algemeen als normaal beschouwd
- **Toepasselijk**: Toepasbaar binnen een afgebakend domein of probleem set
- **Bruikbaar**: Wordt ervaren als eenvoudig in gebruik. Vermindert de inspanning/drempel om de standaard te gebruiken.
- **Volwassen**: Voldoende ondersteunende tooling, ervaring, documentatie. Referentie implementatie(s)
- **Stabiel**: Gestructureerd verbeteringsproces, wijzigingen zijn backward compatibel

2 Opdrachtbeschrijving

2.1 Onderzoeksvragen

De gemeente Den Haag heeft SIG gevraagd om als onafhankelijke autoriteit op het gebied van kwaliteitsanalyses van software systemen te onderzoeken welke gevolgen het gebruik van StUF-BG heeft op de volgende doelstellingen:

1. **Interoperabiliteit**: (door een eenduidig service contract); Interoperabiliteit maakt het mogelijk om op basis van een interfacedefinitie “standaard stekkers en stopcontacten” te maken die altijd passen. Hierdoor kunnen koppelingen worden hergebruikt, loont voorinvestering in koppelingen en is minder maatwerk nodig voor het bouwen van koppelingen. Compliancy met andere (internationale) standaarden speelt hierbij een belangrijke rol
2. **Kostenreductie**: Naast minder maatwerk voor het bouwen van koppelingen en hergebruik van bestaande code door interoperabiliteit moet de standaard eenvoudig met behulp van (recente versies van) de gangbare ontwikkelomgevingen te implementeren zijn. Daarnaast moet het StUF instrumentarium voor authenticatie en autorisatie aansluiten op hetgeen door de gangbare IAM voorzieningen en voorzieningen voor API management (WS gateway’s, servicebussen e.d.) wordt ondersteund. De bovengenoemde voordelen leiden tot kostenreductie tijdens bouw en operatie
3. **Bevorderen marktwerking**: Marktwerking zal worden bevorderd wanneer de standaard dusdanig is opgezet (laagdrempelig & niet complex) dat voorinvestering (in stekkers/stopcontacten) loont, en de investering voor nieuwkomers relatief laag is. Een hoge adoptiegraad (veel stekkers om te bedienen, veel stopcontacten om op aan te sluiten) heeft een aanzuigende werking. Ook de aansluiting bij moderne architectuurstijlen speelt een rol bij adoptie van de standaard door nieuwe jonge spelers in de markt
4. **Bevorderen innovatie**: Innovatie wordt bevorderd door adoptie binnen de standaard van gangbare architectuurstijlen en formaten voor gegevensuitwisseling met moderne consumers. Denk aan mobile devices, the internet of things (IoTs), en Linked Data. Hierbij is het belangrijk om te constateren of het wijzigingsproces met name gericht is op “aanjagen en bijblijven” of op “backward compatibility” t.b.v. bestaande systemen?

Het gaat daarbij om de vraag of gemeenten de beoogde doelen kunnen verwezenlijken door compliancy aan de StUF-BG standaard, of dat er alternatieve benaderingen zijn waarmee de beoogde doelen ook kunnen worden gehaald.

2.2 Uitgangspunten SIG

Ter beantwoording van de gestelde vraagstukken evalueert SIG met name de bruikbaarheid van StUF-BG 0310. Specifiek bekijkt SIG de bruikbaarheid van StUF-BG vanuit het perspectief van een gemeentelijke ontwikkelaar die de standaard moet implementeren.

SIG hanteert de volgende aannames m.b.t. de gemeentelijke ontwikkelaar:

- Werkt met de meest gebruikte implementatietechnologieën en tooling.
- Loopt niet voorop op het gebied van technologische ontwikkelingen.
- Heeft te maken met een combinatie van legacy en nieuwe systemen.
- Heeft ervaring met bekende SOAP/XML-standaarden.

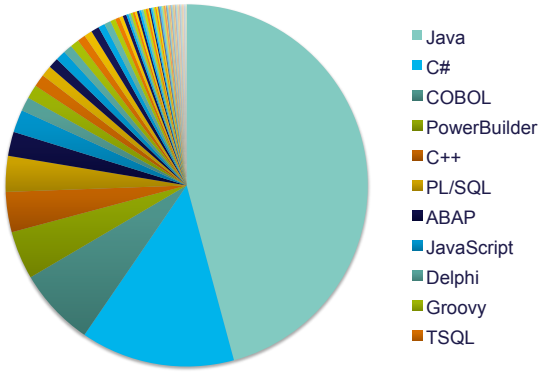
3 Toetsingskader en aanpak

3.1 Beoordelingscriteria, begripsbepaling en operationalisatie

In de volgende paragrafen wordt beschreven welke implementatie technologieën, ontwikkel-tooling en bronnen gebruikt zijn tijdens het onderzoek van SIG.

3.1.1 Java en .NET zijn de meest gangbare implementatie technologieën in de markt

Ten tijde van het onderzoek heeft SIG 1.640 systemen in haar benchmark. Uit de benchmark blijkt dat Java en C# (.NET) de meest gebruikte technologieën zijn in de markt. Het is daarom aannemelijk dat systemen die een StUF-BG interface willen implementeren van een van deze technologieën gebruik maken. In het onderzoek heeft SIG gericht op het onderzoeken van StUF-BG i.c.m. deze twee technologieën.



Figuur 3.1: Verdeling van de technologieën in de SIG benchmark

3.1.2 Gebruikte standaard ontwikkel-tooling

SIG heeft de volgende standaard Java en .NET ontwikkelomgevingen en tooling gebruikt in haar onderzoek.

Ontwikkelomgevingen

SIG heeft gebruik gemaakt van de volgende ontwikkelomgevingen:

- **Java:** Eclipse Luna
- **.NET (C#):** Microsoft Visual Studio Express 2013

Code generatoren

Code generatoren zijn tools die classes in .NET en Java genereren o.b.v. WSDL's (en bijbehorende XSD's). SIG heeft de volgende code generatoren gebruikt tijdens haar onderzoek:

- **In Java:** JAXB (Apache CXF tools)
- **In .NET (C# of Visual Basic):** Wsdl.exe en SvcUtil.exe. NB: Xsd.exe valt buiten de scope van het onderzoek, omdat deze tool beperkt is tot het genereren van code o.b.v. XML Schema's.

Serializers en deserializers

Serializers zijn tools die (code) objecten omzetten naar XML berichten. Deserializers zetten XML berichten om in code objecten. SIG heeft de volgende (de)serializers tijdens haar onderzoek gebruikt:

- **In Java:** JAXB (serializer en deserializer)
- **In .NET (C# of Visual Basic):** XMLSerializer (serializer en deserializer). De XMLSerializer is gebruikt i.c.m. met alle genoemde .NET code generatoren.

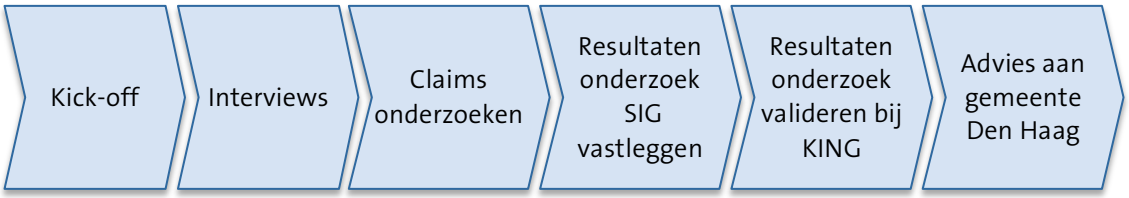
3.1.3 Gebruikte StUF-specifieke bronnen

SIG heeft gedurende het onderzoek een aantal bronnen geraadpleegd. Deze bronnen bestaan uit zowel openbare StUF-specifieke documentatie, procesbeschrijvingen en tooling, alsmede StUF-BG implementaties en documentatie die is opgesteld door de gemeente Den Haag en haar leveranciers t.b.v. het gebruik van StUF(-BG). De volgende bronnen zijn geraadpleegd:

- De StUF documentatie m.b.t. het gebruik en beheer van StUF(-BG)
- Wijzigingsverzoeken en discussies op het openbaar forum
- Handleidingen en documenten over het gebruik van StUF-BG, geschreven en gebruikt door de gemeente Den Haag en gemeente Rotterdam, SLTN InterAccess en Wigo4it
- Broncode van de door SLTN InterAccess gerealiseerde StUF-BG interface
- De XSD Resolver documentatie van KING

Het complete overzicht van de bronnen is beschreven in Appendix A.

3.2 Gehanteerd proces



Figuur 3.2: Het door SIG gehanteerde proces

3.3 Planning

De werkzaamheden om tot een eindrapport te komen waren aanvankelijk gepland binnen een periode van 4 weken. De werkzaamheden zijn uiteindelijk over een periode van 17 weken uitgevoerd.

- April 2015: Kick-off
- April t/m juni 2015: Interviews met gemeente Den Haag, SLTN InterAccess en Wigo4it
- Juni 2015: Claims onderzoeken en de resultaten van het onderzoek door SIG vastleggen
- Juli 2015: Resultaten van het onderzoek valideren met KING
- Augustus 2015: Oplevering van het rapport aan de gemeente Den Haag

4 Procesbeschrijving

4.1 Uitvoering en verloop van het proces

4.1.1 Interviews en verslaglegging

SIG heeft interview sessies gehouden met medewerkers van de gemeente Den Haag, SLTN InterAccess en Wigo4it. Na elke sessie heeft SIG een verslaglegging gedaan van de bevindingen en deze gevalideerd bij de ondervraagden. Op basis van de bevindingen uit de interviews heeft SIG een aantal claims van de gemeente Den Haag geformuleerd. Het onderzoek van SIG heeft zich gericht op het individueel onderzoeken van deze claims.

4.1.2 Claims onderzoeken

SIG heeft de bruikbaarheid van StUF-BG onderzocht o.b.v. de geformuleerde claims. Tijdens het onderzoek heeft SIG de volgende handelingen verricht:

- Bestuderen van StUF en StUF-BG documentatie en beheerprocessen
- Onderzoeken academische invalshoeken m.b.t. het meten van omvang en complexiteit van XML schema's
- StUF-BG vergelijken met een set open standaarden
- De bruikbaarheid van StUF-BG onderzoeken met gebruik van standaard .NET en Java ontwikkelomgevingen en tooling

4.1.3 Resultaten onderzoek SIG vastleggen

Resultaten die gedurende het onderzoek van SIG zijn verzameld zijn vastgelegd in een PowerPoint presentatie, welke zijn gepresenteerd aan KING ter validatie van de onderzoeksresultaten.

4.1.4 Validatie van onderzoeksresultaten bij KING

De door SIG verzamelde onderzoeksresultaten zijn met KING gevalideerd. De feedback van KING is, indien relevant, meegenomen in het onderzoek. Na het verwerken van de feedback van KING, zijn de definitieve onderzoeksresultaten vastgelegd in dit document.

4.1.5 Advies aan de gemeente Den Haag

De evaluatie van de in hoofdstuk 2 gestelde doelstellingen en het uiteindelijke advies van SIG dat volgt uit het gevoerde onderzoek, is vastgelegd in dit document.

4.1.6 Ervaringen met StUF-BG tijdens het onderzoek

Het gebruik van StUF-BG vereist een aanzienlijke aanloop. Uitdagingen bij het doorgronden en te gebruiken van de standaard:

- De omvang van de StUF-BG standaard
- Het grote aantal gedocumenteerde richtlijnen
- De onduidelijkheden in het beoogd gebruik van de standaard
- De problemen in het gebruik van standaard .NET en Java ontwikkelomgevingen en tooling i.c.m. StUF-BG

4.2 Bijeenkomsten

30 april 2015 – gemeente Den Haag

- Cathy Dingemanse – Projectmanager migratie Haagse Bron
- Borjan Cace – Adviseur Applicatie Integratie bij gemeente Den Haag
- André van den Nouweland – Integratie-architect en lid van de StUF expertgroep

15 mei 2015 – SLTN InterAccess

- Ewout van Battum – Architect bij SLTN.

2 juni 2015 – Wigo4IT

- Robert Ruitenbeek
- Derk-Jan Krasenberg

28 juli 2015 – KING

- Maarten van den Broek
- Henri Korver

5 Validatie van de claims

5.1 Claims door gemeente Den Haag

Vanuit de interview sessies heeft SIG een aantal claims van de gemeente Den Haag gedefinieerd. Op basis van de claims heeft SIG de bruikbaarheid van StUF-BG nader onderzocht. In de volgende paragrafen worden de claims van de gemeente Den Haag beschreven.

5.1.1 StUF-BG is niet goed bruikbaar voor de programmeurs/beheerders

De gemeente Den Haag heeft aangegeven dat zij problemen ervaart in het gebruik van StUF-BG, doordat de gebruikte standaard ontwikkelomgevingen en tooling niet werkt met de geleverde StUF-BG contracten (WSDL's en XSD's). De gemeente Den Haag geeft hiervoor de volgende redenen:

- **StUF-BG is groot en complex (C1a):** De StUF-BG standaard is dusdanig groot en complex dat standaard ontwikkel-tooling niet werkt met de XML schema's van StUF-BG.
- **StUF-BG bevat constructies die niet voorkomen in andere bekende standaarden (C1b):** De XML schema's van StUF-BG maken gebruik van XSD constructies die zelden worden gebruikt door andere standaarden. De gebruikte standaard ontwikkel-tooling is niet in staat deze constructies goed te verwerken.

5.1.2 De SOAP binding van StUF-BG is niet specifiek genoeg (C2)

De gemeente Den Haag geeft aan dat de StUF standaard onduidelijk is over het gebruik van stuurgegevens bij SOAP implementaties. Stuurgegevens worden in de body van een SOAP bericht opgenomen en bevatten o.a. de adressering van de zender en de ontvanger van een bericht. In de StUF standaard staat niet beschreven hoe de stuurgegevens kunnen worden beveiligd. Beveiliging (bijv. door middel van encryptie) is noodzakelijk om de stuurgegevens te kunnen gebruiken voor authenticatie en autorisatie. Omdat het beoogd gebruik van stuurgegevens bij een SOAP implementatie niet wordt beschreven, geven partijen verschillende invullingen aan het gebruik ervan, waardoor afstemming tussen partijen is vereist voordat een koppeling o.b.v. de StUF-BG standaard kan worden gebruikt.

5.1.3 De betekenis van generieke kennisgevingsberichten is afhankelijk van de context (C3)

Een kennisgeving is een "mutatie" van de gegevens in het bronsysteem waarbij de gegevens in de "oude" en de "huidige" toestand worden beschreven. Volgens de gemeente Den Haag is uit een kennisgevingsbericht niet direct op te maken wat de "oude" en "huidige" situatie is, noch welke wijzigingen plaats hebben gevonden. Wijzigingen moeten worden afgeleid door de objecten in het bericht te vergelijken en eventueel nadere details op te vragen via een extra bericht.

5.1.4 StUF-BG kan alleen gedeeltelijk geïmplementeerd worden door de gemeente Den Haag. Dit vereist extra afstemming met afnemers (C4)

De gemeente Den Haag geeft aan dat de complete StUF-BG standaard dusdanig groot en complex is, dat het implementeren van de gehele standaard niet mogelijk is. Als provider kan zij de standaard slechts gedeeltelijk implementeren. Hierdoor biedt de technische interface van de StUF-BG standaard geen inzicht in de werkelijk ondersteunde functionaliteit, waardoor extra afstemming is vereist met afnemers alvorens de standaard kan worden gebruikt. Er is hierdoor dus geen sprake van een standaard StUF stekker en stopcontact.

5.1.5 Implementatie van StUF-BG levert performance en beschikbaarheidsrisico's op (C5)

Volgens de gemeente Den Haag, kunnen in de 'SELECT à la SQL' structuur van een vraagbericht veel (combinaties van) zoekcriteria worden gespecificeerd. Deze combinaties van zoekcriteria in

bevragingen van StUF-BG kunnen dusdanig complex zijn, dat beantwoording door de provider beschikbaarheidsrisico's kunnen opleveren met eventueel uitval als gevolg. Problemen kunnen ontstaan doordat de provider ingewikkelde combinaties van data moet uitvragen/samenvoegen en/of doordat de hoeveelheid opgevraagde informatie groot is. De volgende bevragingen hebben beschikbaarheidsrisico's opgeleverd voor het SBS systeem van de gemeente Den Haag:

- Het opvragen van alle personen die wonen op een adres met nummer tussen 0 en 200.
- Opvragen van alle personen die wonen op een adres beginnend met een 'A' en een postadres met een 'E' (Gebruik van wildcards met een opgave 'niet exact'). Verwerking leidt tot een *join* tussen verschillende tabellen, welke uitval tot gevolg kan hebben, nog voordat SBS met (een groot aantal) resultaten kan komen.

5.2 Aanpak validatie

Om de claims van de gemeente Den Haag te valideren heeft SIG een aantal werkzaamheden uitgevoerd. Deze werkzaamheden staan in deze paragraaf beschreven.

5.2.1 Bestuderen van de StUF documentatie

SIG heeft de StUF en StUF-BG documentatie bestudeerd om kennis te verwerven over het gebruik van StUF-BG. Daarnaast heeft SIG ook gekeken naar protocolbindingen en beheerprocessen van StUF-BG en instructie documenten die zijn opgesteld door de gemeente Den Haag t.b.v. het gebruik van StUF-BG.

5.2.2 Academische invalshoeken

N.a.v. de claims van de gemeente Den Haag heeft SIG in de wetenschappelijke literatuur gekeken naar methoden om kenmerken van XML schema's te meten en te vergelijken:

- **Omvang:** SIG meet de omvang van XML schema's in een standaard in Lines of Code (LOC)[1].
- **Complexiteit:** SIG meet de complexiteit van XML schema's in een standaard m.b.v. de relatieve McCabe Cyclomatic Complexity (MCC_{rel}). Informatie over de gehanteerde metrieken voor het berekenen van de MCC_{rel} is te vinden in appendix B.

5.2.3 Benchmark van XML standaarden

SIG heeft StUF-BG vergeleken met een geselecteerde set open standaarden waarin XML schema's worden gebruikt. De 'benchmark set' bevat 17 standaarden, bestaande uit:

- Nationale en internationale standaarden
- Gangbare en minder gangbare standaarden
- Overheids-specifieke en algemene standaarden

N.a.v. de claims van de gemeente Den Haag heeft SIG de omvang en complexiteit van StUF-BG vergeleken met de andere standaarden in de SIG benchmark. Daarnaast heeft de gemeente Den Haag ook aangegeven dat StUF-BG gebruik maakt van constructies die niet gebruikt worden in andere standaarden. SIG heeft het gebruik van een aantal specifieke constructies in StUF-BG vergeleken met de andere standaarden. De lijst van de gebruikte standaarden in de SIG benchmark staat in appendix C.

5.2.4 Onderzoek naar de bruikbaarheid van StUF-BG met standaard tooling en ontwikkelomgevingen

[1] Heitlager, Ilja, Tobias Kuipers, and Joost Visser. "A practical model for measuring maintainability." *Quality of Information and Communications Technology, 2007. QUATIC 2007. 6th International Conference on the*. IEEE, 2007.

SIG heeft onderzoek verricht naar de bruikbaarheid van StUF-BG in combinatie met standaard ontwikkel-tooling en ontwikkelomgevingen (voor .NET en Java). Hierbij zijn de volgende handelingen verricht:

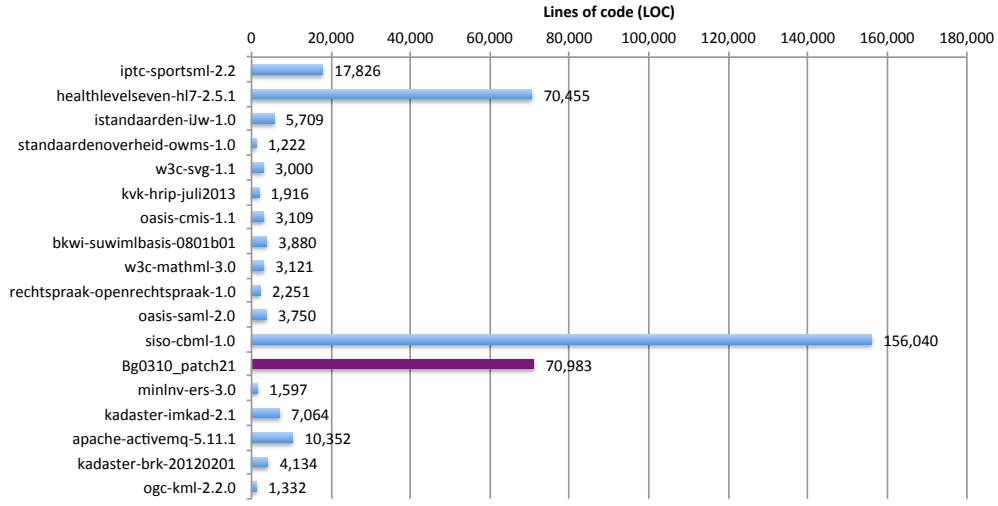
- Het genereren van resolved StUF-BG schema's m.b.v. de XSD Resolver van KING
- Het genereren van StUF-BG code in Java en .NET o.b.v. de XML schema's die geproduceerd worden door de XSD Resolver. Het genereren van Java en .NET code o.b.v. de oorspronkelijke StUF-BG schema's bleek niet mogelijk.
- Het onderzoeken van wat de impact bij ontwikkelen is van het gebruik van constructies die in StUF-BG veel worden gebruikt, maar weinig voorkomen in andere standaarden of door de gemeente Den Haag als problematisch zijn aangemerkt.

5.3 Claim 1: De standaard tooling werkt niet met de geleverde contracten

In deze paragraaf staat beschreven hoe de door de gemeente Den Haag beschreven claims zijn onderzocht en wat de resultaten zijn van het onderzoek door SIG.

5.3.1 StUF-BG is een grote standaard

De omvang van de standaard is bepaald o.b.v. het aantal regels (Lines of Code – LOC), waaruit de standaard bestaat. In vergelijking met andere standaarden in de SIG benchmark is StUF-BG een grote standaard.

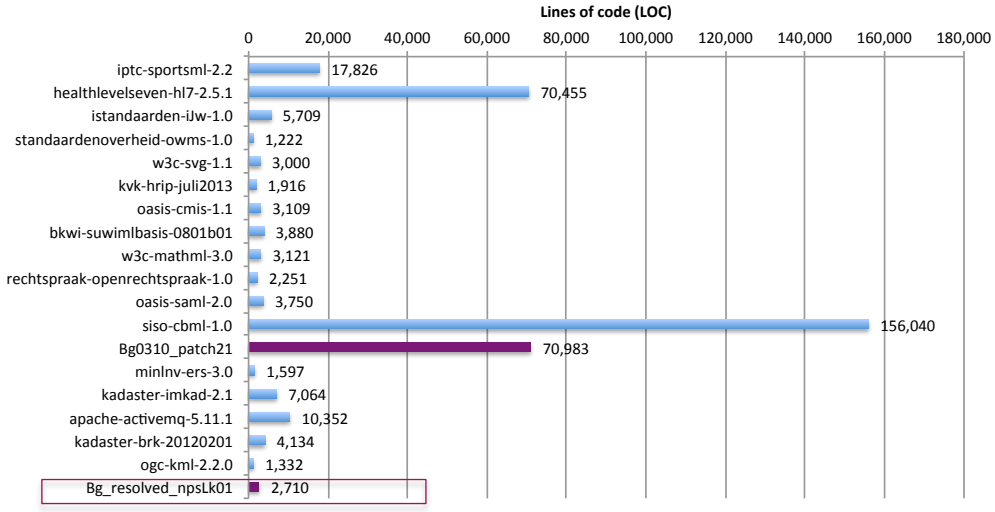


Figuur 5.1 : Vergelijking omvang StUF-BG met de benchmark

5.3.2 Een resolved sub-set van StUF-BG kan significant kleiner worden dan het origineel

Wanneer de XSD resolver wordt gebruikt om een sub-set van de StUF-BG standaard te resolven, kan het resulterende schema significant kleiner worden dan de oorspronkelijke StUF-BG standaard, doordat ongebruikte types en elementen worden verwijderd. Het samenstellen van een sub-set is een handmatige stap.

De omvang van het resulterende schema is afhankelijk van de gekozen sub-set. In het onderstaande voorbeeld omvat de gekozen sub-set enkel het bericht npsLk01 (kennisgevingsbericht voor natuurlijke personen). Ten opzichte van de benchmark van SIG is de omvang van dit schema ongeveer 15% van de gemiddelde omvang. Wanneer de uitzonderlijk grote standaarden *healthlevelseven-hl7-2.5.1* en *siso-cbml-1.0* niet worden meegerekend in het gemiddelde, is de omvang van de sub-set ongeveer 57% van de gemiddelde omvang.



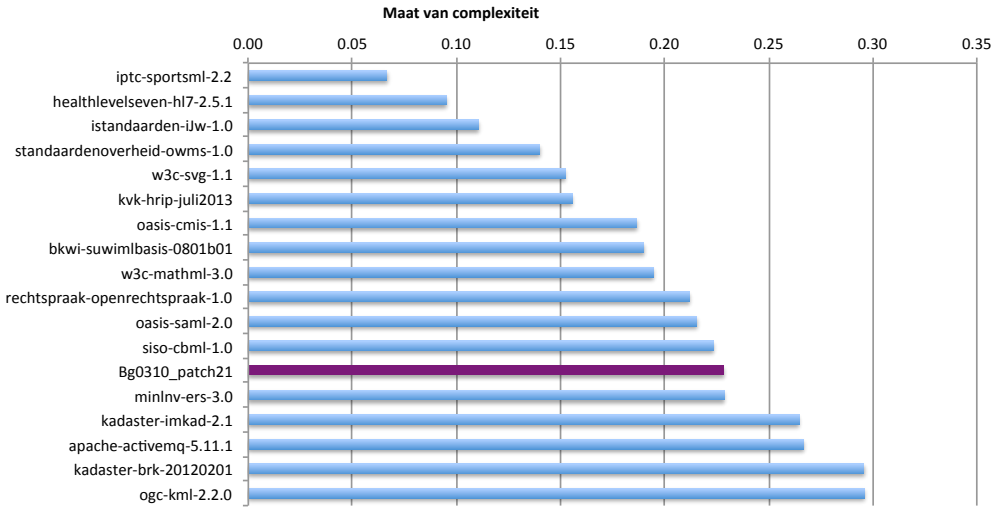
Figuur 5.2: Vergelijking omvang van een resolved sub-set (npsLk01) van StUF-BG met de benchmark

Om de genoemde NPS-Lk01 sub-set te verkrijgen, zijn de volgende handelingen verricht:

- Kopieer *bg0310_msg_mutatie.xsd* naar bestand *bg0310_msg_mutatie_npsLk01.xsd*.
- Verwijder alle onnodige elementen uit *bg0310_msg_mutatie_npsLk01.xsd*.
- Pas de *configuration.xml* van de XSD Resolver aan: voeg een nieuw *configurationSectorModel* toe, laat *pathSectormodel* verwijzen naar *bg0310_msg_mutatie_npsLk01.xsd*. Zet voor dit model *active* op *"yes"*.
- Zet bij alle andere sectormodellen *active* op *"no"*.
- Draai XSD Resolver. Het resulterende XML schema bevat enkel de types en elementen die gebruik worden voor NPS-Lk01 kennisgevingen.

5.3.3 StUF-BG is een complexe standaard

SIG heeft o.b.v. de relatieve McCabe Cyclomatic Complexity (MCC_{rel}) de complexiteit van StUF-BG vergeleken met andere standaarden in de SIG benchmark. Uit deze benchmark blijkt dat StUF-BG een complexe standaard is t.o.v. de andere standaarden in de SIG benchmark. Hierbij is het belangrijk op te merken dat MCC_{rel} een weergave is van de gemiddelde complexiteit per XML node; De waarde van MCC_{rel} staat los van de omvang van de standaard. Indien gebruik wordt gemaakt van een sub-set van de StUF-BG standaard, zal de complexiteit bij benadering gelijk blijven.

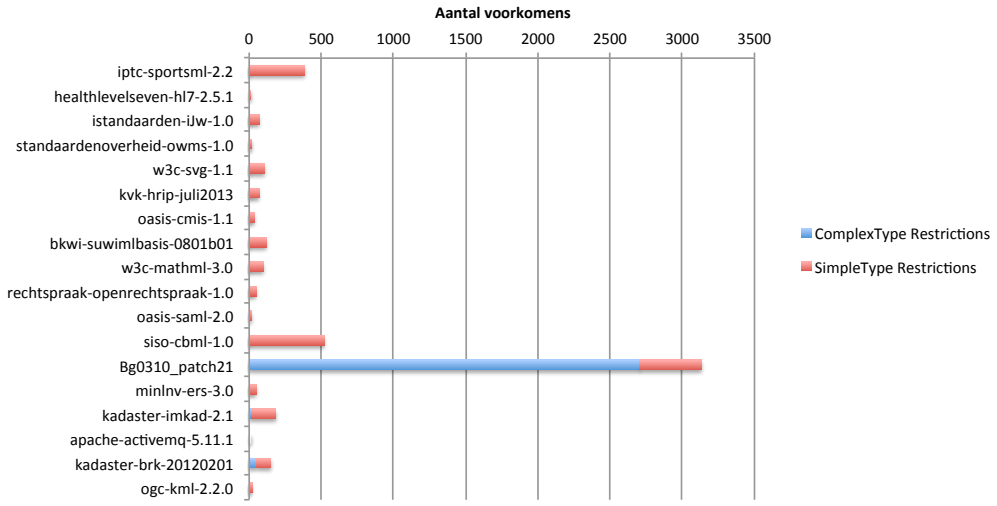


Figuur 5.3: Vergelijking complexiteit StUF-BG met de benchmark

5.3.4 Drie standaarden in de benchmark maken gebruik van complexType restrictions

In StUF-BG wordt 2704 maal gebruik gemaakt van complexType restrictions. SimpleType restrictions worden in alle standaarden gebruikt. Naast StUF-BG maken de volgende standaarden beperkt gebruik van complexType restrictions:

- Kadaster-Brk-20120201 (53x)
- Kadaster-Imkad-2.1 (13x)



Figuur 1.4: Vergelijking gebruik restrictions StUF-BG met de benchmark

5.3.5 Het gebruik van complexType restrictions door StUF-BG faciliteert het behoud van een eenduidig informatiemodel in koppelvlakken

StUF-BG legt het RSGB 2.0 informatiemodel vast in de XML Schema's. Wanneer partijen eigen koppelvlakken definiëren o.b.v. StUF-BG, hoeven zij niet zelf het benodigde gedeelte van het RSGB informatiemodel te definiëren, maar kan m.b.v. complexType restrictions het benodigde gedeelte van het informatiemodel uit de entiteit basistypes worden "geselecteerd". KING heeft aangegeven dat het op deze manier gebruik maken van complexType restrictions bedoeld is om een eenduidig informatiemodel te behouden in koppelvlakken.

5.3.6 Java en .NET code generatoren kunnen niet omgaan met complexType restrictions

In de interview sessies met de gemeente Den Haag is aangegeven dat Java en .NET code generatoren niet om kunnen gaan met restrictions op complexTypes. Het SIG onderzoek bevestigt deze claim. De reden hiervoor is dat complexType restrictions tegengesteld werken aan het OO (Object Oriënted) paradigma waarop Java en C# zijn gebaseerd. Het genereren van code objecten vanuit StUF-BG in Java en C# vereist dat de StUF-BG schema's eerst met de XSD Resolver tool worden bewerkt.

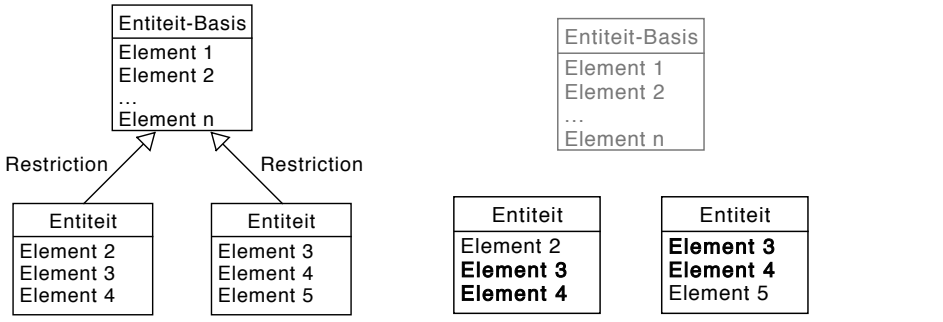
5.3.7 De XSD Resolver wijzigt alleen XSD bestanden

De XSD Resolver van KING wijzigt XSD bestanden en voegt alle schema's met dezelfde *targetNamespace* samen in één bestand. De XSD Resolver past echter enkel de XSD's van StUF-BG aan; na het uitvoeren van de XSD Resolver moet de ontwikkelaar handmatig de WSDL's van StUF-BG aanpassen met verwijzingen naar de resolved XSD schema's. Overigens dient de ontwikkelaar de XSD Resolver eerst goed te configureren, waarbij opgelet moet worden dat de juiste berichten uit de StUF onderlaag in de resolved schema's worden opgenomen.

5.3.8 De XSD Resolver verbreekt relaties tussen entiteiten

De XSD Resolver is een tool van KING dat "resolved XSD's" creëert vanuit de StUF-BG standaard, door o.a. restricties en attribuutgroepen uit de XSD's te verwijderen. Met de resolved XSD's heeft SIG wel code objecten kunnen genereren in Java. Echter, het verwijderen van restricties en attribuutgroepen verbreekt relaties tussen entiteittypen met de volgende gevolgen:

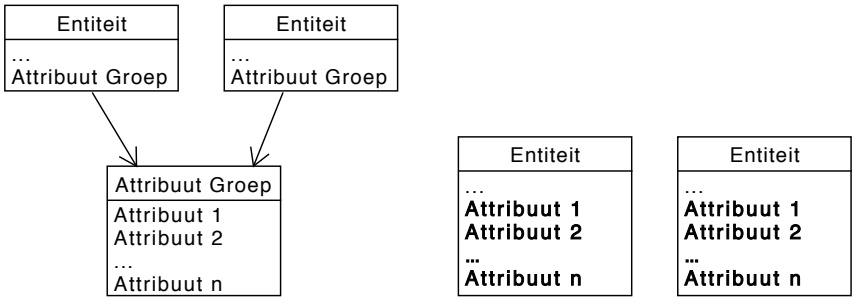
- Het verwijderen van overbodige typen maakt de resolved StUF-BG standaard kleiner en overzichtelijker.
- Door het verbreken van de ervingsrelaties tussen entiteiten ontstaat er duplicatie in de gegenereerde code, doordat meerdere verschillende entiteiten exact dezelfde elementen hebben.
- Door het verwijderen van attribuutgroepen worden attributen gedupliceerd over entiteiten. Hierdoor ontstaat duplicatie in de gegenereerde code.



Unresolved StUF-BG schema: Entiteiten erven elementen van Entiteit-Basis

Resolved StUF-BG schema: Entiteit-Basis wordt niet meer gebruikt. Elementen in entiteiten (Element 3 & 4) worden gedupliceerd

Figuur 5.5: Resultaat van gebruik resolver tool op elementen



Unresolved StUF-BG schema: Entiteiten gebruiken dezelfde attribuutgroep

Resolved StUF-BG schema: Attribuutgroepen worden verwijderd. Attributen gedupliceerd over entiteiten

Figuur 5.6: Resultaat van gebruik resolver tool op attributen

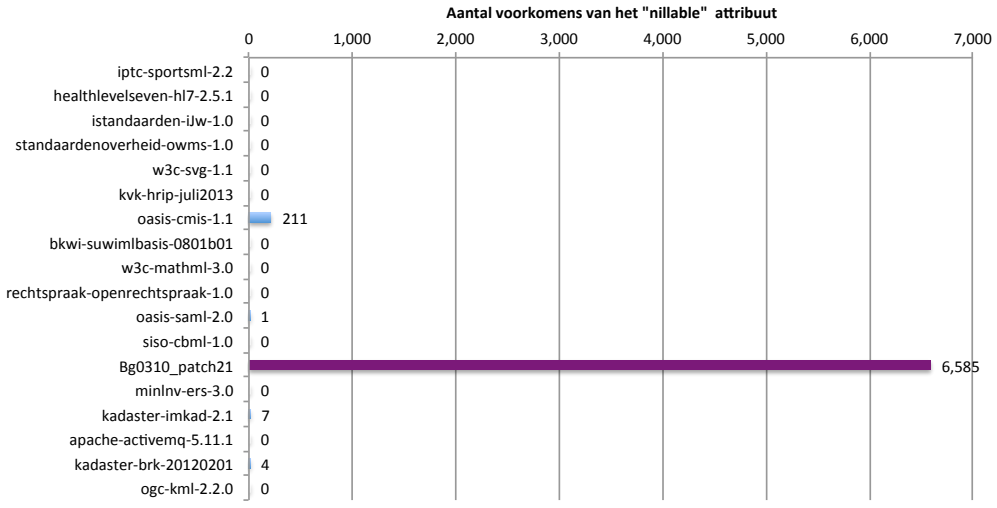
5.3.9 Vijf standaarden in de benchmark maken gebruik van *nillable="true"*

Het *nillable* attribuut kent in StUF meerdere gebruiksdoeleinden:

- I.c.m. met het StUF:NoValue attribuut om aan te geven waarom een verplicht element ontbreekt;
- In de scope van een bevraging, om aan te geven welke informatie wordt opgevraagd.

Het *nillable* attribuut wordt in StUF-BG 6585 maal gebruikt. Naast StUF-BG wordt dit attribuut in de SIG benchmark door vier andere standaarden gebruikt:

- Kadaster-Imkad-2.1 (7x),
- Kadaster-brk-20120201 (4x),
- Oasis-saml-2.0 (1x)
- Oasis-cmis-1.1 (211x).



Figuur 5.7: Vergelijking gebruik nillable binnen StUF-BG en de benchmark

5.3.10 Serializers werken niet bij het gebruik van *nillable="true"* i.c.m. andere attributen

SIG heeft onderzocht hoe serializers van Java en .NET omgaan met het *nillable="true"* attribuut. SIG heeft de volgende serializers gebruikt, in combinatie met gegenereerde code:

- De JAXB serializer van Java i.c.m. de door JAXB gegenereerde StUF-BG code
- De XMLSerializer van .NET i.c.m. de door SvcUtil.exe gegenereerde StUF-BG code
- De XMLSerializer van .NET i.c.m. de door Wsdl.exe gegenereerde StUF-BG code

SIG heeft vastgesteld dat zowel JAXB als XMLSerializer in staat zijn *nil* elementen te serializeren en deserializeren. Echter zowel JAXB als XMLSerializer hebben de volgende problemen bij het gebruik van het *nillable="true"* attribuut wanneer andere attributen ook in een *nil* element (<element nil="true"/>) worden meegenomen:

- Het is niet mogelijk om objecten als *nil* elementen met andere attributen te serializeren.
  - Bij het serializeren van objecten met een *null*-waarde naar een element wordt dit element in XML opgenomen als *nil* element. Echter kunnen andere attributen van het element niet worden meegenomen in de serializatie, ook niet wanneer deze attributen *fixed* en/of verplicht zijn.
  - Om een element met attributen te serializeren, dient het corresponderend object geïnstantieerd te zijn. Geïnstantieerde objecten worden niet als *nil* elementen geserializeerd. Zie figuur 5.8.
- Bij het deserializeren van *nil* elementen waarin andere attributen zijn gespecificeerd, worden deze elementen door de deserializer niet als *nil* element herkend, met als gevolg dat er een object wordt geïnstantieerd. Dit object weet niet of dat het van een *nil* element is afgeleid. Zie figuur 5.9.

Er zijn meer voorbeelden m.b.t. het serializeren van *nillable* elementen in Java en .NET te vinden in appendix D.

```
private static NPSVraagScope createScope() {

    NPSVraagScope scope = new NPSVraagScope();

    VoornamenE voornamen = new VoornamenE();
    voornamen.setNoValue(NoValue.VASTGESTELD_ONBEKEND);

    JAXBElement<VoornamenE> e = jaxbElement(voornamen);
    e.setNil(true);

    scope.setVoornamen(e);

    return scope;
}
```

```
<NPS-vraagScope xmlns:ns2="http://www.egem.nl/StUF/sector/bg/0310" xmlns:ns1="http://www.egem.nl/StUF/StUF0301">
    <voornamen ns1:noValue="vastgesteldOnbekend"/>
</NPS-vraagScope>
```

Figuur 5.8: Geïnstantieerd object met StUF:NoValue waarde wordt niet als nil element opgenomen in geserializeerd bericht.

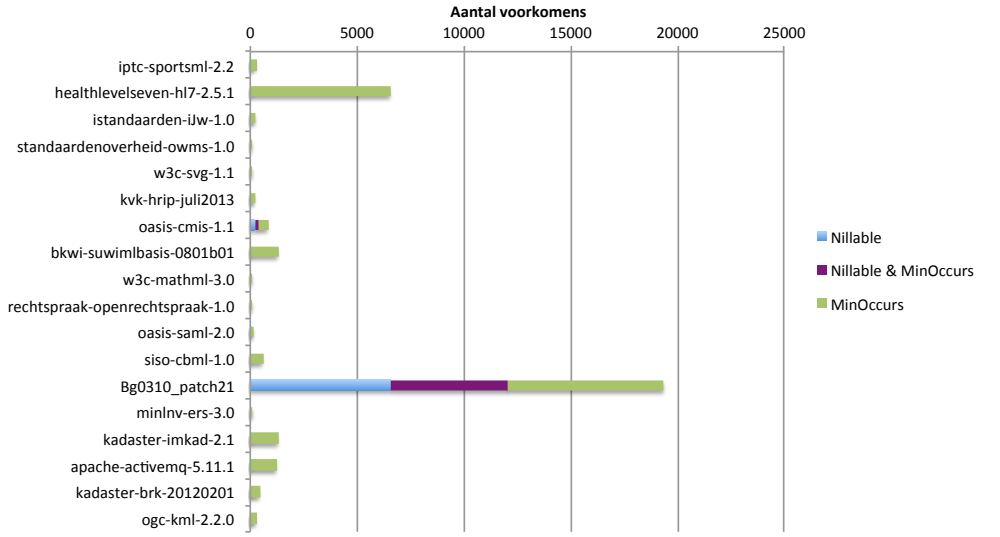
```
<BG:NPS-vraagScope xmlns:BG="http://www.egem.r
    xmlns:StUF="http://www.egem.nl/StUF/StUF03

    <voornamen StUF:NoValue="geenWaarde" nil="
</BG:NPS-vraagScope>


vraagScope.getVoornamen().isNil()
```

Heeft waarde *false*

Figuur 5.9: Bij deserialisatie van nil element met een StUF:NoValue attribuut, wordt het element niet als nil herkend in de code

5.3.11 Twee standaarden in de benchmark maken gebruik van de combinatie *nillable="true"* en *minOccurs="0"*

De combinatie van het *nillable="true"* attribuut met het *minOccurs="0"* attribuut wordt in StUF-BG 5455 maal gebruikt. Enkel Oasis-cmis-1.1 maakt ook gebruik van deze combinatie (203x) van attributen. Hieruit blijkt dat het gebruik van de combinatie *nillable="true"* en *minOccurs="0"* vrijwel niet voorkomt in andere standaarden.



Figuur 5.10: Vergelijking gebruik nillable & minOccurs binnen StUF-BG en de benchmark

5.3.12 Functioneel onderscheid tussen *nillable="true"* en *minOccurs="0"* is niet af te leiden uit de StUF-BG schema's

Volgens de XSD's van de StUF-BG standaard mogen elementen die zowel de *nillable="true"* en *minOccurs="0"* attributen bevatten:

- Volledig ingevuld worden in de XML
- Als *nil* –element voorkomen in de XML
- Geheel worden weggelaten uit de XML

In tegenstelling tot de betekenis van het *nillable* attribuut, dient in StUF een element weggelaten te worden wanneer deze niet verplicht is en:

- Geen waarde heeft OF
- Niet relevant is voor de informatie uitwisseling.

De bovenstaande mogelijkheden zorgen er voor dat het functionele onderscheid dat StUF-BG wil maken tussen het gebruik van een *nil* element, of het volledig weglaten van een element, niet door de StUF-BG schema's wordt afgedwongen. Correcte toepassing van het functionele onderscheid vereist kennis van de StUF richtlijnen en een specifieke implementatie in de programmeertaal.

5.3.13 .NET code generatoren leggen onderscheid *nillable="true"* en *minOccurs="0"* niet vast in de code

Uit het onderzoek van SIG is gebleken dat de alleen de *JAXB* (Java) code generator in staat is het onderscheid tussen *nillable="true"* en *minOccurs="0"* vast te leggen in de gegenereerde code. De standaard code generatoren van .NET (Wsdl.exe en ScvUtil.exe) zijn hier toe niet in staat. De gevolgen hiervan zijn:

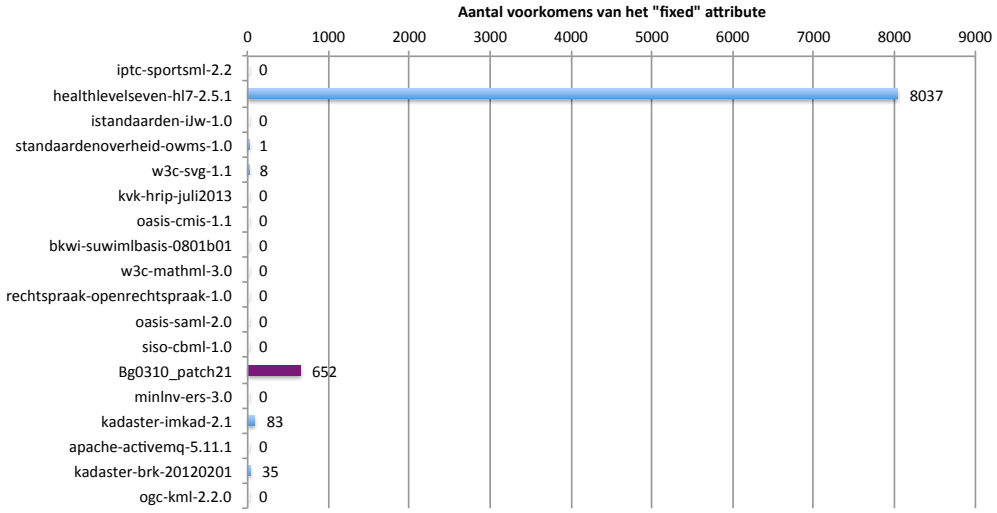
- .NET objecten met een *null*-waarde worden door de *XMLSerializer* altijd geserializeerd tot *nil* elementen. Het is niet mogelijk om te specificeren wanneer een element geheel weggelaten dient te worden.
- Bij het deserializeren van een *nil* of ontbrekend element zal in beide gevallen het corresponderend object in code een *null*-waarde hebben. De code maakt geen onderscheid tussen *nil* elementen en afwezige elementen.

5.3.14 Zes standaarden in de benchmark maken gebruik van *fixed* attributen

Het *fixed* attribuut wordt in StUF-BG gebruikt om aan te geven op welk entiteittype een object betrekking heeft.

StUF-BG maakt 652 maal gebruik van het *fixed* attribuut. In de SIG benchmark wordt het *fixed* attribuut door 5 andere standaarden gebruikt:

- Healthlevelseven-hl7-2.5.1 (8037x)
- Kadaster-imkad-2.1 (83x)
- Kadaster-brk-5.11.1 (35x)
- W3c-svg-1.1 (8x)
- Standaardenoverheid-owms-1.0 (1x)



Figuur 5.11: Vergelijking gebruik fixed binnen StUF-BG en de benchmark

5.3.15 Serializers leggen de waarde van *fixed* attributen niet vast

SIG heeft onderzocht hoe Java en .NET code generatoren omgaan met het *fixed* attribuut. Geen van de code generatoren (zowel Java als .NET) legt de waarde van *fixed* attributen vast in de gegeneerde code. Dit heeft de volgende gevolgen:

- Bij het serializeren van objecten naar XML moet de waarde van een *fixed* attribuut expliciet gezet worden. Wanneer de waarde afwijkt van de voorgeschreven waarde wordt dit niet gecontroleerd door de serializer.
- Bij het deserializeren van XML worden ongeldige waarden die zijn meegegeven in het bericht niet gecontroleerd. Om de geldigheid van een XML bericht te garanderen dient het bericht expliciet gevalideerd te worden tegen het XML Schema.

De volgende standaard libraries kunnen worden gebruikt om de berichten te valideren tegen het XML schema:

- **Java:** Javax.xml.schemas
- **.NET:** System.Xml.Schema

5.3.16 De StUF-BG schema's dwingen het correct gebruik van *StUF:noValue* niet af

Het *StUF:noValue* attribuut wordt door de StUF onderlaag gedefinieerd. Dit attribuut wordt in bepaalde gevallen als extra attribuut meegegeven in *nil* elementen en dient als verklaring voor de afwezigheid van dat element. Echter, de XML Schema's van StUF dwingen het correct toepassen van *StUF:noValue* niet af:

- *StUF:noValue* kan als attribuut worden meegeven aan een element, ongeacht of dit element *nil* is
- *StUF:noValue* kan worden weggelaten bij een element, ook wanneer dit attribuut volgens de StUF richtlijnen gebruikt dient te worden

5.3.17 Er bestaat een StUF-Kletser API voor Java, maar er zijn geen vergelijkbare tools beschikbaar voor .NET

KING heeft een Java API beschikbaar gesteld, genaamd de StUF-Kletser, welk gebruikt kan worden bij de implementatie van interfaces o.b.v. StUF-BG. Echter, de StUF-Kletser is niet kosteloos en er is nagenoeg geen informatie beschikbaar op internet. Voor .NET en andere technologieën zijn er geen soortgelijke API's bekend.

5.3.18 De StUF-Kletser ondersteunt niet het gebruik van code generatie

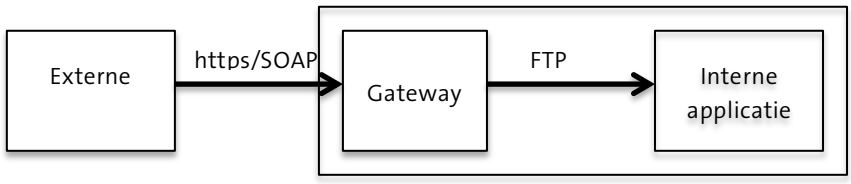
Bij het gebruik van de StUF-Kletser wordt geen gebruik gemaakt van code generatie; Tijdens de ontwikkeling van een StUF-BG implementatie dienen gebruikte entiteittypen handmatig aangemaakt te worden door ontwikkelaars.

5.4 Claim 2: De SOAP binding van StUF-BG is niet specifiek genoeg

De SOAP binding is het mechanisme waarmee berichtenuitwisseling plaatsvindt. In de SOAP binding wordt o.a. beschreven welk transportprotocol gebruikt wordt voor de berichtenuitwisseling, maar kan ook beschreven worden welke WS-* veiligheidsvoorzieningen gebruikt worden ter beveiliging.

5.4.1 Gebruik van stuurgegevens is verplicht maar het beoogd gebruik is onduidelijk

Opname van stuurgegevens in StUF berichten is wel verplicht, echter de StUF standaard is onduidelijk over het gebruik van stuurgegevens als autorisatie mechanisme bij een SOAP implementatie. KING heeft aangegeven dat stuurgegevens niet bedoeld zijn als veiligheidsvoorziening. Opname van stuurgegevens bij StUF berichten is bedoeld om benodigde sturingsinformatie end-to-end te communiceren. D.w.z. dat applicaties ook onderling StUF berichten met elkaar kunnen uitwisselen (bijv. m.b.v. files) zonder gebruik te maken van SOAP. In het onderstaande voorbeeld worden StUF berichten tussen de gateway en interne applicatie middels FTP gecommuniceerd, waarbij de interne applicatie over de informatie in de stuurgegevens moet beschikken. Indien stuurgegevens in de header van een SOAP bericht staan, zijn extra voorzieningen noodzakelijk om de stuurgegevens beschikbaar te stellen aan de interne applicatie.



Figuur 5.12: Stuurgegevens bij gebruik van een file-based interface (FTP)

De in deze paragraaf beschreven onduidelijkheid in beoogd gebruik heeft er mede toe geleid dat binnen SBS de niet versleutelde stuurgegevens voor autorisatie worden gebruikt. Dit is een onveilige situatie, die in de Haagse implementatie is weggenomen door de door de provider ingevulde gegevens te vervangen door informatie uit een 'trusted source' voordat een StUF bericht van een afnemer bij SBS wordt geaccepteerd. Hiermee wordt afgedwongen dat de afzender in de stuurgegevens ook werkelijk is wie hij zegt dat hij is.

5.4.2 Gebruik van WS-* voorzieningen wordt niet beschreven door STUF

In tegenstelling tot stuurgegevens, maakt WS Security (bijvoorbeeld m.b.v. SAML 2.0 tokens) encryptie van gevoelige gegevens en authenticatie tussen partijen wel mogelijk. De Protocol binding van StUF beschrijft echter niet hoe WS-* voorzieningen toegepast kunnen worden.

5.5 Claim 3: De betekenis van algemene kennisgevingsberichten is afhankelijk van de context

De gebeurtenissen, die in algemene kennisgevingsberichten worden doorgegeven, zijn niet direct af te leiden uit de structuur van het bericht. Ter illustratie het volgende kennisgevingsbericht van overlijden.

```
<object a:entiteittype="NPS" a:sleutelOntvangend="1461360" a:verwerkingssoort="W">
    ...
    <voorletters>F.</voorletters>
    <voornamen>Friedrich</voornamen>
    <geslachtsaanduiding>M</geslachtsaanduiding>
    <geboortedatum>19450303</geboortedatum>
    <overlijdensdatum a:noValue="geenWaarde" xsi:nil="true"/>
    <inp.overlijdenplaats a:noValue="geenWaarde" xsi:nil="true"/>
    <inp.overlijdenLand a:noValue="geenWaarde" xsi:nil="true"/>
    ...
</object>
<object a:entiteittype="NPS" a:sleutelOntvangend="1461360" a:verwerkingssoort="W">
    ...
    <voorletters>F.</voorletters>
    <voornamen>Friedrich</voornamen>
    <geslachtsaanduiding>M</geslachtsaanduiding>
    <geboortedatum>19450303</geboortedatum>
    <overlijdensdatum>20141117</overlijdensdatum>
    <inp.overlijdenplaats>'s-Gravenhage</inp.overlijdenplaats>
    <inp.overlijdenLand>6030</inp.overlijdenLand>
    ...
</object>
```

Figuur 5.13 Voorbeeld van een generieke kennisgeving van overlijden

5.5.1 "Oude" en "Huidige" situatie niet af te leiden uit de structuur van een kennisgevingsbericht

In de technische structuur van kennisgevingsberichten worden zowel de “oude” en “huidige” situatie opgenomen onder een element genaamd “object”. In deze structuur wordt het onderscheid tussen de twee situaties niet vastgelegd.

5.5.2 Volgorde voorkomen "Oude" en "Huidige" situatie voorgeschreven door StUF richtlijnen

In de StUF richtlijnen staat beschreven dat - indien een kennisgeving een wijziging betreft - het eerste voorkomen van het element “object” de “oude” situatie aanduid en het tweede voorkomen de “huidige” situatie.

5.5.3 Opname van benodigde elementen in kennisgevingsberichten is afhankelijk van StUF richtlijnen

Kennisgevingsberichten kunnen meer gegevens bevatten dan daadwerkelijk gewijzigd zijn. De StUF richtlijnen documenteren een aantal regels waarin de noodzakelijkheid wordt bepaald om kerngegevens, dan wel andere gegevens, op te nemen in het bericht. Dit wordt echter niet afgedwongen door de schema’s van de StUF-BG standaard.

5.5.4 Aanleiding kennisgeving moet worden afgeleid uit het bericht

Kennisgevingsberichten kunnen meer gegevens bevatten dan daadwerkelijk gewijzigd zijn. Het vaststellen van de gebeurtenis die de aanleiding is voor het versturen van het bericht, wordt

vastgesteld door elementen in de “oude” en “huidige” situatie met elkaar te vergelijken. Kennisgevingenberichten zijn echter niet beperkt tot het doorgeven van een enkele gebeurtenis. Ter illustratie: een kennisgevingsbericht mag zowel een kennisgeving van overlijden bevatten, alsmede een kennisgeving van naamswijziging.

5.6 Claim 4: StUF-BG kan alleen gedeeltelijk geïmplementeerd worden door de gemeente Den Haag. Dit vereist extra afstemming met afnemers

De gemeente Den Haag heeft aangegeven dat de StUF-BG standaard te groot is om in zijn geheel geïmplementeerd te worden. Hoewel SIG gedurende het onderzoek geen technische bevindingen heeft gedaan die deze claim ondersteunen, heeft SIG de claim voorgelegd aan KING. De reactie van KING heeft tot de volgende bevindingen geleid:

5.6.1 StUF-BG hoeft niet in zijn geheel geïmplementeerd te worden

KING heeft aangegeven dat het niet de verwachting heeft dat een leverancier StUF-BG in zijn geheel implementeert; het is de bedoeling dat leveranciers uit de StUF-BG schema’s een sub-set benodigde berichten selecteren.

5.6.2 StUF-BG is halffabricaat welk dient als basis voor koppelvlakken

KING heeft aangegeven dat StUF-BG niet gezien dient te worden als een op zichzelf staande standaard interface definitie, maar dat het een halffabricaat is. Leveranciers kunnen de StUF-BG standaard gebruiken als basis waarop eigen koppelvlakken gebouwd kunnen worden.

5.6.3 Er is geen centraal beheer van koppelvlakken

KING zou graag zien dat koppelvlakken door providers publiekelijk beschikbaar zijn en worden gedeeld. Momenteel is dit niet het geval. Dit leidt ertoe dat individuele providers hun eigen koppelvlakken definiëren, waardoor afnemers met hun koppelingen moeten afstemmen over individuele ‘maatwerkstekkers’ van verschillende providers.

5.7 Claim 5: Implementatie van StUF-BG levert performance en beschikbaarheidsrisico’s op

De gemeente Den Haag heeft aangegeven dat het implementeren van de vraag/antwoord berichten van StUF-BG beschikbaarheidsrisico’s kan opleveren voor de provider. Ter illustratie heeft de gemeente Den Haag de volgende voorbeeld bevraging gegeven, waarop verwerking tot uitval van SBS kan leiden:

- Opvragen alle personen die wonen op een adres met huisnummer tussen 0 en200.

```
<BG:npsLv01>
    ...
    <BG:parameters>
        <StUF:sortering>4</StUF:sortering>
    </BG:parameters>
    <BG:vanaf StUF:entiteittype="NPS">
        <BG:verblijfsadres>
            <BG:aoa.huisnummer>0</BG:aoa.huisnummer>
        </BG:verblijfsadres>
    </BG:vanaf>
    <BG:totEnMet StUF:entiteittype="NPS">
        <BG:verblijfsadres>
            <BG:aoa.huisnummer>200</BG:aoa.huisnummer>
        </BG:verblijfsadres>
    </BG:totEnMet>
    <BG:scope>
        <BG:object xsi:nil="true" StUF:scope="alles" />
    </BG:scope>
</BG:npsLv01>
```

Figuur 5.14: Voorbeeld opvragen alle personen die wonen op een adres met huisnummer tussen 0 en 200

5.7.1 De technische structuur van StUF-BG beperkt de mogelijke vraagberichten niet

De structuur van bevragingen legt geen beperkingen op aan mogelijke selectiecriteria. In het voorbeeld dat gegeven wordt door de gemeente Den Haag, vereist verwerking van de gestelde vraag dusdanig veel systeemmiddelen dat het tot uitval van het systeem kan leiden.

5.7.2 De door StUF-BG gedefinieerde sorteringen bepalen de te ondersteunen selectiecriteria

StUF-BG definieert per bevragingsbericht een aantal sorteringen. De sorteringen die voor een gegeven bevragingsbericht zijn gedefinieerd, dienen ter specificatie van de selectiecriteria die door een bevraagd systeem ondersteund dienen te worden. Het gebruik van sorteringen t.b.v. de te ondersteunen selectiecriteria wordt beschreven in de StUF richtlijnen (StUF301.pdf, pagina 84).

5.7.3 Gedefinieerde sorteringen mogen in koppelvlakken ingeperkt worden

Indien een provider niet in staat is alle in StUF-BG gedefinieerde sorteringen te ondersteunen, mogen koppelvlakken de sorteringen inperken. Dit kan m.b.v. *simpleType restrictions*. Door gedefinieerde sorteringen in koppelvlakken in te perken, kan een provider aan haar afnemers communiceren welke selectiecriteria worden ondersteund. Bevragingen die selectiecriteria bevatten die niet door een beantwoordend systeem worden ondersteund mogen geweigerd worden.

5.7.4 De gebruikte voorbeeld bevragingen van de gemeente Den Haag hoeven niet ondersteund te worden

Het voorbeeld dat gegeven is door de gemeente Den Haag bevat een onvolledige combinatie van selectiecriteria en hoeft daarbij niet in volledigheid ondersteund te worden. Volgens de StUF richtlijnen, geven sorteringen een volgorde waarop geselecteerd dient te worden. Indien een sortering wordt gebruikt, waarbij een selectiecriterium niet wordt gespecificeerd, mogen opvolgende criteria genegeerd worden. In het gegeven voorbeeld van de gemeente Den Haag kan de *NPS-sortering* nummer 4 worden gebruikt:

```
<StUF:sorteringObject>
    <StUF:nummer>4</StUF:nummer>
    <StUF:element>verblijfsadres/gor.straatnaam</StUF:element>
    <StUF:element>verblijfsadres/aoa.huisnummer</StUF:element>
    <StUF:element>verblijfsadres/aoa.huisletter</StUF:element>
</StUF:sorteringObject>
```

Figuur 5.15: Een in StUF-BG gedefinieerde sortering voor bevragingen op natuurlijke personen

De SBS van gemeente Den Haag hoeft de gegeven bevraging niet te ondersteunen omdat bij het selecteren van alle personen die wonen op adressen met huisnummer 0 t/m 200, het selectiecriterium *verblijfsadres/gor.straatnaam* niet wordt gespecificeerd. Omdat het eerste selectiecriterium in de sortering niet wordt gespecificeerd, mogen de opvolgende criteria *verblijfsadres/aoa.huisnummer* en *verblijfsadres/aoa.huisletter* genegeerd worden.

6 Bevindingen

In dit hoofdstuk worden de bevindingen door SIG, op basis van de gevonden resultaten uit hoofdstuk 5, met betrekking tot de bruikbaarheid van StUF-BG beschreven.

6.1 Implementatie van StUF-BG vereist extra inspanning

SIG verwacht dat het implementeren van StUF-BG koppelingen extra inspanning vereist.
De extra inspanning is het gevolg van:

- StUF-BG is een halffabricaat is, waardoor extra stappen noodzakelijk zijn om een sub-set te selecteren en/of koppelvlakken te definiëren.
- StUF-BG werkt niet goed met standaard .Net en Java ontwikkel-tooling, waardoor extra inspanning noodzakelijk is om alternatieve oplossingen te vinden en/of te implementeren.
- De complexe technische structuur van de StUF-BG standaard een vereist complexe implementatie.
- Conformiteit van een StUF-BG implementatie is mede afhankelijk van StUF(-BG) richtlijnen, waardoor zorg gedragen moet worden dat alle nodige (en talrijke) richtlijnen uit de documentatie van StUF geïmplementeerd worden.
- Code gegenereerd vanuit resolved StUF-BG schema's bevat duplicatie, waardoor inspanning vereist wordt om te implementeren en te onderhouden, of extra inspanning wordt vereist om oplossingen te vinden waarmee het implementeren van gedupliceerde voorzieningen vermeden kan worden.
- Implementaties voor generieke kennisgevingen moeten in staat zijn gebeurtenissen af te leiden uit kennisgevingsberichten. Dit voegt een extra dimensie toe aan de complexiteit van een StUF-BG implementatie.
- Een veilige StUF-Bg koppeling vereist de implementatie van additionele veiligheids-voorzieningen (buiten de officiële standaard)

In de volgende paragrafen worden de bovenstaande punten nader toegelicht.

6.1.1 StUF-BG is een halffabricaat, beoogd gebruik van StUF-BG vereist aanpassing van de StUF-BG XML Schema's en WSDL's

Alvorens een StUF-BG implementatie te bouwen, dient bij het beoogd gebruik van StUF-BG eerst een sub-set van de XML Schema's te worden geselecteerd en eventueel een koppelvlak te worden gedefinieerd. Dit is mogelijk door de benodigde XML Schema's en WSDL's (handmatig) te bewerken.
Het beoogd gebruik leidt in feite tot een nieuwe sub-standaard en vereist extra inspanning alvorens een koppeling kan worden geïmplementeerd.
Bij de implementatie van StUF-BG in SBS zijn bovenstaande stappen niet uitgevoerd en worden de complete StUF-BG schema's meegeleverd, waardoor impliciet de complete StUF-BG standaard volledig wordt ondersteund. Deze aanpak is volgens KING niet in lijn met het beoogd gebruik van de standaard.

6.1.2 StUF-BG vereist een complexe implementatie

De Relatieve McCabe complexiteit (MCC_{rel}) in XML Schema's biedt een maat voor de mogelijke variëteit van XML berichten, relatief t.o.v. de omvang van het gehele XML Schema. Een systeem dat een XML standaard implementeert, zal in haar implementatie de mogelijke variaties moeten ondersteunen. Een complexe standaard zoals StUF-BG, zal een complexe implementatie vereisen.

6.1.3 Standaard .NET en Java tooling werkt niet met StUF-BG

Het onderzoek van SIG heeft bevestigd dat de StUF-BG standaard niet werkt met de standaard ontwikkel-tooling gebruikt door de gemeente Den Haag en haar leveranciers. SIG heeft de volgende problemen vastgesteld:

- *ComplexType* restrictions werken tegengesteld aan het OO-paradigma, waardoor de complete StUF-BG standaard niet vastgelegd kan worden in OO-talen zoals Java en C#. Het gebruik van StUF-BG vereist de XSD Resolver om *complexType* restrictions uit de standaard te verwijderen.
- Code generatoren zijn niet in staat een aantal van de door StUF-BG gebruikte constructies (complexType restrictions, nillable, combinatie nillable & minOccurs en fixed) vast te leggen in code.
- Standaard .NET en Java Serializers zijn niet in staat StUF-BG berichten op de juiste wijze te serializeren en deserializeren.

Doordat standaard .NET en Java tooling op meerdere aspecten ontoereikend is bij het gebruik van StUF-BG, zijn er alternatieve oplossingen vereist. Mogelijke oplossingen zijn:

- Vinden van andere (open source of COTS) tooling,
- Ontwikkeling van eigen tooling
- "Workaround"-oplossingen binnen StUF-BG implementaties.
- Gebruik van een .NET of Java library (zoals de StUF-Kletser). In dit geval dienen implementaties van entiteittypen handmatig gemaakt te worden door ontwikkelaars

6.1.4 Conformiteit van StUF-BG implementatie is afhankelijk van StUF(-BG) richtlijnen

Het correct gebruik van StUF-BG wordt niet afgedwongen door de XML Schemas' van StUF-BG, waardoor:

- Incorrect gebruik van het *StUF:noValue* attribuut kan leiden tot berichten die technisch wel valide zijn, maar niet voldoen aan de richtlijnen van StUF.
- Correcte en veilige implementatie van bevragingen niet door de StUF-BG standaard wordt afgedwongen. Mogelijke selectiecriteria worden niet door de XML Schema's beperkt, maar zijn afhankelijk van conformiteit aan de StUF-BG richtlijnen.
- Correcte implementatie van kennisgevingen niet door de XML Schema's wordt afgedwongen. De XML Schema's maken geen onderscheid tussen de "oude" en "huidige" situatie en de gegevens die opgenomen dienen te worden in een kennisgevingsbericht zijn afhankelijk van verscheidene regels die zijn vastgelegd in de StUF-BG richtlijnen.

Een implementerend systeem moet naast de ondersteuning van de technische structuur van StUF-BG, ook de nodige richtlijnen van de StUF-BG standaard implementeren, waarbij de gedocumenteerde richtlijnen uit meer dan 100 pagina's bestaan. Veelal worden meerdere richtlijnen in een enkele paragraaf tekst opgenomen. Hierdoor kunnen richtlijnen moeilijker ter naslag worden opgezocht, waardoor de kans dat richtlijnen over het hoofd gezien worden door een ontwikkelaar vergroot wordt. Indien niet alle benodigde richtlijnen worden geïmplementeerd, leidt dit tot implementatie fouten en/of dient dit afgestemd te worden tussen providers en afnemers.

6.1.5 Duplicatie in resolved StUF-BG schema's leidt tot duplicatie in een implementatie

In de resolved schema's van StUF-BG ontstaat duplicatie tussen typen die in de oorspronkelijke StUF-BG schema's dezelfde entiteit-basistypen delen of dezelfde attribuutgroepen gebruiken. Hierdoor ontstaat er in de gegeneerde code ook duplicatie. Het gebruik van gedupliceerde code vereist het implementeren van gedupliceerde voorzieningen, dan wel alternatieve benaderingen om duplicatie van voorzieningen te vermijden.

6.1.6 Een implementatie moet de gebeurtenis uit generieke StUF-BG kennisgevingsberichten afleiden

Bij het gebruik van generieke kennisgevingsberichten wordt de gebeurtenis afgeleid uit het bericht zelf. Indien een specifieke gebeurtenis nadere afhandeling vereist, dient een

implementerend systeem een dergelijke gebeurtenis zelf af te leiden uit de ontvangen kennisgevingsberichten.

6.1.7 Een veilige StUF-BG implementatie moet extra veiligheidsvoorzieningen implementeren

Omdat de stuurgegevens in de huidige vorm niet geschikt zijn voor authenticatie en/of autorisatie, is extra inspanning vereist om een veilig authenticatie en autorisatie mechanisme te implementeren in StUF-BG koppelingen. Daarbij is ook extra inspanning vereist wanneer de implementatie van een dergelijke veiligheidsvoorziening moet worden afgestemd met providers of afnemers.

6.2 Op StUF-BG gebaseerde koppelingen zijn niet direct herbruikbaar

Het onderzoek van SIG heeft uitgewezen dat het gebruik van StUF-BG op meerdere vlakken afstemming vereist tussen providers en afnemers. Hierdoor is er geen sprake van een standaard StUF-stekker waarmee afnemers kunnen communiceren met willekeurige providers. In de volgende paragrafen wordt beschreven welke vlakken afstemming vereisen.

6.2.1 Definiëren van eigen koppelvlakken leidt tot verschillende sub-standaarden

Providers en afnemers dienen o.b.v. StUF-BG eigen koppelvlakken te definiëren. Hierdoor worden o.b.v. StUF-BG eigen sub standaarden gecreëerd. Zolang er geen sprake is van centraal beheer van koppelvlakken, waarbij koppelvlakken worden gedeeld tussen providers, bestaat de kans dat afnemers voor verschillende providers verschillende koppelingen moeten implementeren.

6.2.2 Gebruik van WS-* voorzieningen vereist afstemming

Stuurgegevens zijn niet geschikt voor autorisatie of authenticatie. WS-* voorzieningen zijn wel geschikt voor deze doeleinden, maar het gebruik wordt in de StUF richtlijnen niet beschreven. Providers zijn vrij om eigen invulling te geven aan het gebruik van authenticatie en autorisatie middelen, waardoor afnemers een koppeling moeten afstemmen met elk van hun providers.

6.2.3 Gebruik van Stuurgegevens vereist afstemming

Hoewel het beoogde doel van stuurgegevens wel is gedocumenteerd in de StUF richtlijnen, is er onduidelijkheid bestaat over de juiste toepassing. Partijen geven verschillende invulling aan het gebruik van de stuurgegevens, waardoor afstemming noodzakelijk wordt tussen providers en afnemers.

6.3 Gebruik van de StUF-BG standaard vereist een aanzienlijke aanloop

SIG heeft vastgesteld dat het gebruiken van StUF-BG een aanzienlijke aanloop vereist voor partijen die onbekend zijn met de standaard. In de volgende paragrafen wordt de toedracht tot deze aanloop uiteengezet.

6.3.1 StUF-BG bevat constructies die weinig voorkomen in andere standaarden

StUF-BG bevat constructies die niet of weinig voorkomen in andere standaarden in de SIG benchmark. Ontwikkelaars die deze constructies niet kennen moeten deze constructies eerst begrijpen, alvorens te leren werken met de StUF-BG standaard. Met name het gebruik van constructies door StUF-BG die niet werken met standaard tooling zijn voor ontwikkelaars uitdagend, omdat zij ook moeten leren hoe zij deze constructies op de juiste wijze kunnen gebruiken en welke code ervoor moet worden geschreven.

6.3.2 Problemen met het gebruik van standaard tooling verlengen de aanloop

Doordat standaard tooling niet goed werkt met StUF-BG en omdat er geen eenduidige set (referentie) tools wordt beschreven waarmee de StUF-BG standaard wel kan worden gebruikt, wordt elke nieuwe partij geconfronteerd met dezelfde bruikbaarheidsproblemen. Voordat een

partij in staat is effectief aan de slag te gaan met StUF-BG, dient zij eerst de nodige oplossingen te vinden om de door StUF-BG gebruikte constructies te kunnen gebruiken.

6.3.3 Gebruik van StUF-BG vereist kennis van de StUF richtlijnen

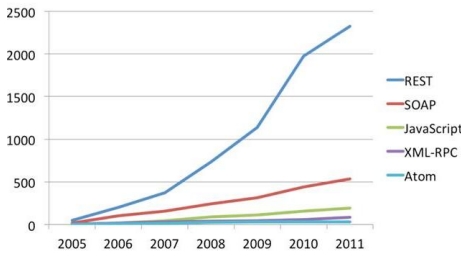
De XML Schema's van de StUF-BG standaard alleen zijn niet toereikend voor het gebruik van de standaard. De StUF-BG standaard wordt vergezeld door een StUF handleiding die meer dan 100 pagina's (meer dan 56.500 woorden) met richtlijnen voorschrijft. Het aantal richtlijnen in dit document is dusdanig groot, dat SIG deze niet heeft kunnen tellen. Het correct gebruik van StUF-BG vereist dat ontwikkelaars deze richtlijnen kennen en begrijpen.

6.4 Mogelijk gebruik van StUF-BG bij nieuwe architectuurstijlen en formaten

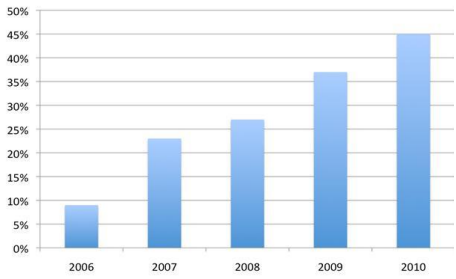
Hoewel niet gerelateerd aan de claims van gemeente Den Haag m.b.t. de bruikbaarheid van de StUF-BG standaard, heeft SIG - ter onderbouwing van de onderzoeksvragen naar de toepasbaarheid van StUF-BG op innovatieve vlakken - onderzoek verricht naar de bruikbaarheid van StUF-BG i.c.m. nieuwe architectuurstijlen en formaten.
SIG heeft geconstateerd dat dergelijk gebruik op delen van de StUF-BG standaard mogelijk is, maar op dit moment niet door de huidige versie van de StUF-BG standaard en het beheerproces worden ondersteund. Het gebruik van StUF-BG i.c.m. nieuwe architectuurstijlen en formaten is tot op heden nog niet gedaan. Om dit te realiseren is een koppelvlak nodig, waarbij niet volledig gebruik kan worden gemaakt van de bestaande structuur van StUF-BG en eigen invulling door de ontwerper van het koppelvlak noodzakelijk is. Deze aanpak wordt in de volgende paragrafen toegelicht.

6.4.1 Het gebruik van REST en JSON groeit sterk

SIG ziet in de markt het gebruik van de REST architectuurstijl en het JSON formaat sterk groeien in de afgelopen jaren. Zie figuur 6.1 en 6.2.



Figuur 6.1: Toename van beschikbare API's (Bron: Programmable Web)



Figuur 6.2 Percentage API's dat gebruik maakt van JSON (Bron: Programmable Web)

REST is een architectuurstijl die gebruik maakt van het http protocol, waarin beschikbare informatie geïdentificeerd wordt middels unieke URI's. Informatie kan worden opgevraagd, toegevoegd, bewerkt of verwijderd worden middels http GET, PUT, POST en DELETE requests. In tegenstelling tot het SOAP protocol, dat onlosmakelijk is gebonden aan het gebruik van XML, is REST onafhankelijk van een specifiek uitwisselingsformaat. Gegeven de significante groei in het gebruik van REST bij API's, heeft SIG gekeken naar de mogelijkheden om StUF-BG te gebruiken bij een dergelijke resource-georiënteerde aanpak.

JSON staat voor Javascript Object Notation. JSON is – net als XML - een data-uitwisselingsformaat. Echter, JSON is compacter dan XML[2]. Waar in XML informatie wordt vastgelegd middels elementen met waarden, wordt in het JSON formaat informatie vastgelegd middels sleutel-waarde paren. In tegenstelling tot XML, waar elementen voorzien kunnen worden van attributen, biedt de structuur van JSON geen vergelijkbare constructie.

6.4.2 Definiëren van een REST koppelvlak met StUF-BG

De StUF richtlijnen eisen dat koppelvlakken zoveel mogelijk gebruik maken van de aangereikte entiteiten. Echter bij het definiëren van eigen koppelvlakken zijn partijen vrij om eigen invulling te geven aan de standaard waar nodig. StUF is niet beperkt tot SOAP bindingen. De StUF-BG Schema's kunnen gebruikt worden als basis voor REST koppelvlakken, waarbij

- Entiteiten en relaties geïdentificeerd worden middels URI's.
- Relaties niet direct worden opgenomen in de gebruikte entiteiten, maar dat entiteiten in het koppelvlak worden uitgebreid met voorzieningen voor het meegeven van URI's naar gerelateerde informatie.
- Bevragingen middels URI's geïdentificeerd worden, waarbij selectiecriteria middels pad parameters en/of querystring parameters worden meegegeven.
- Kennisgevingen op entiteiten middels http PUT, POST of DELETE requests worden uitgevoerd.

In appendix E is een voorbeeld gegeven van een REST georiënteerd StUF-BG koppelvlak.

6.4.3 Vastleggen van StUF-BG entiteiten in JSON formaat

StUF entiteiten die vastgelegd worden in code objecten, kunnen geserializeerd worden tot - en gedeserializeerd worden uit – het JSON formaat. Hoewel het JSON formaat geen attributen kent zoals aanwezig in XML, kunnen zulke attributen wel middels alternatieve representaties worden opgenomen in JSON. In appendix F is een dergelijk voorbeeld opgenomen.

Er is een ruim assortiment aan tools voor zowel Java (Bijv. Google *Gson*) en .NET (Standaard tool *DataContractJsonSerizalizer*) beschikbaar die deze functionaliteit biedt.

[2] Nurseitov, Nurzhan, et al. "Comparison of JSON and XML Data Interchange Formats: A Case Study." *Caine* 9 (2009): 157-162.

7 Conclusies

In dit hoofdstuk worden de conclusies geformuleerd door de bevindingen van SIG te relateren aan de doelstellingen die genoemd zijn door de gemeente Den Haag. Er wordt aangegeven welke gevolgen de bevindingen van SIG hebben op de ontwikkeling van software die gebruik maakt van de StUF-BG standaard.

7.1 Interoperabiliteit

De extra afstemmingen en de extra inspanningen bij het gebruik van StUF-BG hebben een negatieve impact op de interoperabiliteit. Een voorinvestering in een koppeling, die de StUF-BG standaard implementeert, is hierbij niet voldoende, aangezien er voor het realiseren van elke nieuwe aansluiting (zowel voor providers als afnemers) extra inspanningen en afstemming noodzakelijk is.

De interoperabiliteit van een interface wordt beïnvloed door het gemak waarmee verschillende partijen (afnemers en de providers) de interface kunnen gebruiken zonder dat daarvoor extra inspanning en/of afstemming noodzakelijk is. Belangrijk hierbij is een eenduidig contract (WSDL en XSD's) en zo min mogelijk additionele randvoorwaarden, die zijn vastgelegd in de documentatie, op websites, etc.. SIG heeft het volgende vastgesteld:

- Koppelvlakken o.b.v. StUF-BG zijn in feite nieuwe sub-standaarden, die niet centraal worden afgestemd en beheerd. Iedere provider kan hierdoor zijn eigen standaard creëren en afnemers moeten hun aansluitingen apart afstemmen met individuele providers. Er is hierbij geen sprake van één standaard StUF-BG stekker, maar van meerdere StUF-BG sub-stekkers, die niet centraal worden beheerd en publiekelijk beschikbaar zijn.
- Pogingen om de StUF-BG volledig te ondersteunen (zoals bij SBS) leiden tot impliciete servicecontracten, waarbij de implementatie van het systeem niet in staat is te voldoen aan het gehele servicecontract. Hierbij is er ook geen sprake van een standaard StUF-Stekker omdat met afnemers moet worden afgestemd welke functionaliteit wordt ondersteund door de provider. Bij deze aanpak ontstaan risico's bij nieuwe versies van de software, doordat niet duidelijk wordt welke impact de wijzigingen hebben op het impliciete service-contract.
- Een beoogde implementatie op basis van StUF-BG vereist extra inspanning. Doordat koppelingen nieuwe sub-standaarden zijn, worden afnemers niet beperkt tot het implementeren van één enkele StUF-BG koppeling, maar moeten voor meerdere providers meerdere koppelingen bouwen (of bestaande koppelingen aanpassen). Zolang er geen sturing is door KING om gestandaardiseerde koppelvlakken op basis van StUF-BG te definiëren en beheren, zal de inspanning die gepaard gaat met de implementatie van StUF-BG zich niet beperken tot de eenmalige investering,
- Het gebruik van additionele (WS-*) veiligheidsvoorzieningen en het gebruik van de stuurgegevens bij een SOAP implementatie vereisen afstemming tussen afnemer en provider, omdat deze veiligheidsvoorzieningen niet worden beschreven door de StUF standaard.

7.2 Kostenreductie

De genoemde bevindingen zullen een negatieve invloed hebben op de verwachte kostenreductie ten gevolge van het gebruik van de StUF-BG standaard. In het uiterste geval zullen er zelfs extra kosten worden gemaakt bij gebruik van StUF-BG in vergelijking met een maatwerk koppeling.

Het bouwen van koppelingen tussen softwaresystemen vergt in het algemeen een aanzienlijke inspanning. Mede doordat er afhankelijkheden zijn met derden, is er een risico op uitloop. Om

deze redenen vormen koppelingen een grote kostenpost op de begrotingen van een software projecten. Door gebruik te maken van een standaard interface wordt verwacht dat kosten lager zijn dan bij implementatie van een maatwerk interface. Dit is mogelijk doordat er minder nieuwe kennis hoeft te worden opgebouwd en er gebruik kan worden gemaakt van eerder gebouwde onderdelen. Bovendien mag verwacht worden dat een standaard het gebruik van veel gebruikte ontwikkel-tooling ondersteunt, waarmee de productiviteit van het ontwikkelteam wordt vergroot.

Uit het onderzoek van SIG is gebleken dat ontwikkelde koppelingen o.b.v. StUF-BG in feite sub-standaarden zijn, die niet zonder extra inspanning en kosten herbruikbaar zijn bij het aanmaken van een nieuwe aansluiting. Dit kunnen zowel impliciete sub-standaarden zijn (zoals bij SBS) of expliciete sub-standaarden (bij aanmaken van een nieuw koppelvlak). Dit wordt versterkt doordat de StUF-BG standaard het gebruik van stuurgegevens en beveiliging ter invulling laat aan providers die hun interfaces aanbieden.

Bovendien werkt de standaard .NET en Java tooling niet met StUF-BG, waardoor extra inspanningen en kosten moeten worden gemaakt bij de ontwikkeling van de interface software.

7.3 Bevorderen marktwerking

De door SIG genoemde bevindingen zorgen er voor dat het gebruik van de StUF-BG standaard voor interfaces tussen applicaties niet laagdrempelig is en een aanzienlijke aanloop vereist voor nieuwe spelers. Het gedwongen gebruik van een standaard als StUF-BG voor de koppeling tussen gemeentelijke applicaties zal daardoor niet leiden tot een vergroting van de marktwerking, waarbij nieuwe spelers eenvoudig alternatieve applicaties ontwikkelen die koppelen o.b.v. de StUF-BG standaard.

Een grote marktwerking bij applicaties, die de StUF-BG standaard ondersteunen, betekent dat veel (ook kleinere) partijen verschillende oplossingen kunnen aanbieden die gebruik maken van een StUF-BG interface om aan te sluiten op het gemeentelijke applicatielandschap. Dit wordt vereenvoudigd doordat de gebruikte standaard laagdrempelig en niet complex is, zodat met een geringe investering de aansluiting kan worden gerealiseerd.

Uit het onderzoek van SIG is gebleken dat:

- Het implementeren van een interface o.b.v. de StUF-BG standaard een aanzienlijke aanloop vereist, waardoor aanzienlijke investeringen gedaan moeten worden voordat een partij effectief aan de slag kan met StUF-BG.
- Het implementeren van een interface o.b.v. de StUF-BG standaard extra inspanning kost, mede door de complexiteit van de standaard. Bovendien zijn gebouwde koppelingen niet altijd eenvoudig herbruikbaar en niet direct onderling uitwisselbaar.
- Het gebruik van het testplatform niet gratis is, waardoor de drempel voor het gebruik door kleine partijen groter wordt.

7.4 Bevorderen innovatie

Zonder actieve sturing en goedkeuring van KING als beheerder van de standaard is de kans klein dat innovaties binnen en met StUF-BG zullen plaatsvinden. Adoptie van de StUF-BG standaard heeft in de markt tot dusver niet geleid tot de toepassing van StUF-BG bij nieuwere architectuurstijlen en formaten. Omdat dergelijke toepassingen tot heden ontbreken in de markt, concludeert SIG dat, hoewel delen van StUF-BG technisch kunnen worden gebruikt in nieuwere architectuurstijlen en formaten, de StUF-BG standaard zelf niet bevorderend is voor deze innovatie.

Een standaard zal innovatie ondersteunen, wanneer deze gangbare en moderne architectuurstijlen en formaten voor gegevensuitwisseling ondersteunt. Hierdoor kan een standaard eenvoudiger worden toegepast bij moderne consumers. StUF-BG ondersteunt een tweetal SOAP bindingen (naast een 'file-based koppeling'). SOAP is een technologie die is ontstaan aan het einde van de vorige eeuw en gericht op architectuurstijlen uit die tijd. SOAP kent vooral toepassingen binnen Enterprise omgevingen. De afgelopen jaren zijn er op het gebied van web services trends ontstaan richting nieuwere architectuurstijlen zoals REST en formaten zoals JSON, welk met name worden gebruikt in communicatie met mobile devices.

Het gebruik van de StUF-BG standaard is niet beperkt tot SOAP en XML, middels het definiëren van koppelvlakken kunnen onderdelen van de StUF-BG standaard ook gebruikt worden met nieuwere architectuurstijlen en formaten (zoals REST en JSON). Een dergelijke architectuurstijl of formaat is tot op heden binnen de StUF-BG standaard niet ontwikkeld, er bestaan op dit moment geen voorbeelden of referentie-implementaties.

Het ontsluiten van een StUF-compliant web-service in (open web) omgevingen waarbij gecommuniceerd wordt met moderne consumers en/of 'the Internet Of Things' middels moderne architectuurstijlen en formaten is technisch mogelijk en realiseerbaar.

In de huidige inrichting van het StUF-BG beheerproces, ligt het ontwikkelen en beheren van nieuwe koppelvlakken bij de partij die de StUF-BG standaard gebruikt en implementeert. Het toepassen van StUF-BG voor een REST of JSON koppelvlak zal dus moeten worden uitgevoerd door de ontwerper van het koppelvlak zelf. Zoals geconstateerd in dit onderzoek wordt hierbij een nieuwe op StUF-BG gebaseerde sub-standaard gecreëerd.

8 Aanbevelingen

8.1 Aanbevelingen voor de Gemeente Den Haag

SIG adviseert de gemeente Den Haag de volgende acties uit te voeren.

8.1.1 Definieer een koppelvlak voor SBS

SIG adviseert een eigen koppelvlak voor SBS te (laten) definiëren o.b.v. de benodigde sub-set van de StUF-BG schema's. Dit stelt de gemeente Den Haag in staat richting haar afnemers expliciet te maken welke functionaliteiten van de StUF-BG standaard ondersteund worden en – daar waar nodig – de standaard uit te breiden met eigen functionaliteiten en definities.

Door deze maatregel zal er minder afstemming noodzakelijk zijn tussen provider en afnemers. Dit zal een kosten verminderend effect hebben bij het koppelen van de systemen aan SBS.

8.2 Aanbevelingen die de Gemeente Den Haag aan KING kan doen

Ter bevordering van het gebruik van de StUF-BG standaard legt de Gemeente Den Haag een aantal nadrukkelijke wensen voor aan KING. Om een toekomst-bestendige standaard te realiseren verdient het aanbeveling **alle** hieronder genoemde aspecten op te nemen in de nieuwe standaard. KING heeft daar toe al stappen gezet, maar dient een explicietere strategie te volgen. Daar bij moet worden opgemerkt dat er bij het realiseren van deze aanbevelingen *breaking changes* kunnen optreden, waardoor een verbeterde versie van de standaard niet meer backward compatible is met oudere versies. Dit zal op korte termijn extra werk met zich meebrengen, maar zal een standaard opleveren, die beter voldoet aan de doelstellingen op het gebied van interoperabiliteit, kostenreductie, marktwerking en innovatie. Een expliciete door KING gedragen strategie is hierbij noodzakelijk omdat anders de oude standaard de vigerende standaard zal blijven.

8.2.1 Werk de richtlijnen van StUF(-BG) verder uit

De StUF richtlijnen verschaffen op een aantal vlakken onvoldoende duidelijkheid over het beoogd gebruik van de StUF-BG standaard. Dit leidt bij gebruikers van de standaard tot misverstanden. Deze kunnen worden voorkomen door de volgende verbeteringen van de StUF richtlijnen:

1. Maak duidelijk dat StUF-BG een halffabricaat is en niet in zijn geheel geïmplementeerd hoeft te worden. Maak hierbij expliciet duidelijk dat StUF-BG bedoeld is als basis voor specifieke koppelvlakken.
2. Beschrijf de stappen en tools die nodig zijn om schema's en WSDL's te maken die de benodigde sub-set van de StUF-BG standaard bevatten. Beschrijf de concrete stappen voor het definiëren van koppelvlakken. Voeg een referentie-implementatie toe, waarop ontwikkelaars kunnen terugvallen, indien de documentatie niet voldoende duidelijkheid geven.
3. Werk het beoogd gebruik van stuurgegevens beter uit in de StUF richtlijnen. Publiceer concrete voorbeelden voor het gebruik van stuurgegevens.
4. Maak duidelijk hoe stuurgegevens gebruikt kunnen worden in een SOAP implementatie i.c.m. WS-* veiligheidsvoorzieningen. Beschrijf het gebruik van WS-* in de StUF protocolbindingen.

Het nader specificeren van de bovengenoemde punten zal er toe leiden dat er minder afstemming noodzakelijk is tussen partijen. Dit zal leiden tot de volgende verbeteringen t.o.v. de geformuleerde doelstellingen:

- Interoperabiliteit: doordat er op uniforme wijze gebruik wordt gemaakt van stuurgegevens en WS-* veiligheidsvoorzieningen

- Kostenreductie: doordat providers middels hun koppelvlakken duidelijker kunnen communiceren welke functionaliteiten worden ondersteund. Hierdoor is minder afstemming tussen providers en afnemers nodig.

8.2.2 Zorg dat alle van StUF-BG afgeleide koppelvlakken publiek beschikbaar zijn

Er zijn geen centrale voorzieningen beschikbaar voor het publiek maken van op StUF-BG gebaseerde koppelvlakken, waardoor koppelvlakken niet worden gedeeld. Providers definiëren unieke koppelvakken, ook wanneer bestaande koppelvlakken dezelfde functionele eisen vervullen. Wanneer providers ieder een uniek koppelvlak aanbieden, moeten afnemers die met meerdere providers koppelen aparte koppelingen bouwen, waardoor er geen sprake is van een standaard StUF stekker.
De aanwezigheid van een centrale database met koppelvlakken zal onnodige inspanningen besparen, doordat bestaande koppelvlakken worden hergebruikt. Hoewel in het StUF Testplatform tot op zekere hoogte bestaande koppelvlakken inzichtelijk worden gemaakt, wordt dit platform niet door alle leveranciers gebruikt en is het gebruik niet gratis.

Deze aanbeveling zal leiden tot de volgende verbeteringen m.b.t. de geformuleerde doelstellingen:

- Interoperabiliteit: Providers kunnen bestaande koppelvlakken hergebruiken. Dit faciliteert de herbruikbaarheid van koppelingen.
- Kostenreductie: Door koppelvlakken te delen, kan er bij providers inspanning bespaard worden wanneer zij een bestaand koppelvlak kunnen gebruiken, doordat zij niet zelf een nieuw koppelvlak hoeven te definiëren.

8.2.3 Maak een library voor afnemers waarmee de drempel voor het gebruik van StUF(-BG) wordt verlaagd

Uit het onderzoek van SIG is gebleken dat de voornaamste bruikbaarheidsproblemen liggen in het werken met de StUF-BG XML schema's en de uit StUF-BG gegenereerde code.
Om de adoptie van de StUF standaard te bevorderen is het belangrijk dat er voorzieningen zijn die het gebruik van StUF faciliteren en de aanloop verminderen door de complexiteit van de StUF-BG standaard af te schermen. Hierdoor zullen afnemers met minder inspanning en investeringen koppelingen kunnen bouwen met providers.

Afnemers zouden niet geconfronteerd moeten worden met de inspanningen rondom het gebruik van de XSD-Resolver, WSDL's, code generatoren, serializers en het valideren van berichten tegen de StUF richtlijnen. Ter realisatie van dit advies, stelt SIG voor libraries te ontwikkelen waarmee afnemers koppelingen kunnen bouwen met minimale inspanning en zonder gebruik te hoeven maken van de WSDL's en XML Schema's die geïmplementeerd worden door hun providers. Een dergelijke library moet voldoen aan de volgende eisen:

- De library moet in meerdere grote technologieën beschikbaar zijn, in ieder geval Java en .NET
- De library moet bruikbaar zijn zonder het genereren van code. Hierbij moeten gebruikers in staat zijn eigen keuzes te maken voor de representatie van domeinobjecten in hun code.
- Het gebruik van de library moet laagdrempelig zijn, dit wordt gerealiseerd doordat de library:
  - gratis is;
  - bekend is bij belanghebbenden;
  - gemakkelijk te vinden is op internet;
  - goed gedocumenteerd is in het gebruik.
- De library moet voldoen aan de volgende functionele eisen:
  - er moeten fouten worden teruggegeven wanneer een bericht niet voldoet aan de StUF richtlijnen;

- er moet ondersteuning worden geboden voor het gebruik van WS-* veiligheidsvoorzieningen;
- de library moet onafhankelijk van SOAP gebruikt kunnen worden en bijv. ook met een REST implementatie te gebruiken zijn.

Implementatie van de beschreven library zal leiden tot de volgende verbeteringen m.b.t. de geformuleerde doelstellingen:

- Kostenreductie: De inspanning die gepaard gaat met het bouwen van koppelingen wordt gereduceerd. Het bouwen van koppelingen wordt hierdoor minder kostbaar.
- Marktwerking: Doordat afnemers niet geconfronteerd worden met de problemen die standaard ontwikkelomgevingen en tooling hebben met StUF-BG, wordt de aanloop om de standaard te gebruiken verkort. Dit vermindert de investering voor nieuwe partijen, waardoor de drempel voor het gebruik van de standaard verlaagd wordt.
- Innovatie: Het gebruik van de library bevordert het gebruik van StUF-BG bij moderne consumers wanneer de library onafhankelijk wordt gemaakt van een specifiek communicatie protocol, zodat deze ook het gebruik van nieuwere architectuurstijlen zoals REST faciliteert.

SIG heeft tijdens het onderzoek kennis gemaakt met de documentatie van de StUF-Kletser library van KING. De observatie van SIG is dat de StUF-Kletser voldoet aan aantal van de gestelde eisen, echter de library moet verder worden ontwikkeld om aan alle eisen te voldoen, waaronder een gratis beschikbaarheid.

8.2.4 Vervang de minder gangbare constructies in StUF(-BG)

Naast het beschikbaar maken van een library voor het afschermen van de complexiteit van StUF-BG voor de gebruikers is het wenselijk om de minder gangbare constructies in StUF-BG te vervangen door constructies, die in ieder geval in Java en C# eenvoudig tot goed werkende software leiden. Hierbij moet er op worden gelet dat na vervanging van de minder gangbare constructies het gebruik van standaard tooling zoals Code generatoren en de standaard .NET en Java Serializers mogelijk wordt.

8.2.5 Zorg dat de StUF-BG schema's zoveel mogelijk het correct gebruik van de standaard afdwingen

De schema's van StUF-BG dwingen het correct gebruik van de StUF-BG standaard niet altijd af, waardoor correcte interpretatie en gebruik van de StUF-BG standaard mede afhankelijk is van gedocumenteerde richtlijnen. Deze documentatie is groot en onoverzichtelijk, waardoor een verhoogde kans op implementatiefouten en/of onvolledige implementaties ontstaat.

SIG adviseert de afhankelijkheid van de StUF-BG richtlijnen te verminderen, door correct gebruik van de standaard zoveel mogelijk af te dwingen in de structurele opbouw van de XML schema's. Vermindering van deze afhankelijkheid zal in het gebruik van StUF-BG leiden tot de volgende verbeteringen m.b.t. de geformuleerde doelstellingen:

- Kostenreductie: Een verminderde afhankelijkheid met de StUF-BG richtlijnen zal ertoe leiden dat het bouwen van koppelingen minder inspanning vereist.
- Marktwerking: Het verminderen van de gedocumenteerde richtlijnen verkort de aanloop voor nieuwkomers om de standaard te leren gebruiken. Dusdanig zal dit een drempel verlagend effect hebben.

8.2.6 Neem voorzieningen voor resource-georiënteerde entiteitrelaties op in de StUF-BG schema's

Om de toepassing van nieuwere architectuurstijlen te bevorderen bij StUF-BG, adviseert SIG dat er in de StUF-BG schema's voorzieningen worden opgenomen om op resource-georiënteerde wijze entiteitrelaties op te nemen in berichten. Hiermee wordt het gebruik van StUF-BG op innovatieve vlakken bevorderd.

BIJLAGEN

A Bronnenoverzicht

A.1 Publieke bronnen

A.1.1 Gebruiksrichtlijnen

- XSD-Resolver.pdf
- stuf_best_practices.pdf
- stuf0301.pdf
- KeuzenVerStUFfing_RSGBv10.pdf

A.1.2 Beheer documentatie

- 7._StUF_Beheermodel_2014_05_28_met_renvooi.pdf
- 2009-3-StUF mini primer.pdf
- 20150402_Presentatie_Roadmap_bg0320_en_zkn0320_agendapunt4.pdf
- 2009-1-StUF artikel.pdf
- CS04-11-08C_notitie_StUF.pdf
- Samengevoegde_reacties_StUF.pdf
- StUF_expertadvies_na_consultatie.pdf
- StUF_expertadvies.pdf
- Toetsingsprocedure_en_criteria_1_4.pdf
- 20141802-Gebruikershandleiding_1_4_1.pdf

A.2 Ontvangen van externe partijen

A.2.1 Gemeente Den Haag

- VraagAntwoord StUF-v2.docx
- scenario's WSG IDM DHB_versie 0 5.docx
- memo StUF november 2012 v0.4
- presentatie StUF 4 december definitief.pptx
- Standpunten gemeente De Haag betreffende StUF v3.doc
- StUF RFCs- definitief v1.1-reacties.doc
- StUF_SOA.pdf
- Uitslag Enquete toekomstvisie StUF regiegroep 5 juni.pdf
- Discussiestuk StUF webservices eigenaardigheden en issues.pdf
- Correspondentie Cathy met Wigo4IT inzake kennisgevingen.pdf
- extract StUF wijzigingen 1-6-2015.xlsx
- Verslag_regiegroep_5_februari_2014_(concept).pdf

A.2.2 SLTN InterAccess

- NPClients Library
- Wiki-pagina met instructies voor het generen van StUF-BG code in .NET

A.2.3 Wigo4it

- Werkinstructie van StUF naar BizTalk StUF.DOCX
- Werkinstructie van StUF-ZKN naar BizTalk StUF-ZKN.DOCX

A.2.4 KING

- bg0310_msg_mutatie_npsLk01.xsd
- stufkletserJavaDoc.zip
- KING feedback op onderzoek SIG 20150825.pdf

B McCabe Cyclomatic Complexity

B.1 McCabe Cyclomatic Complexiteit in XML Schema's

Om een maat te geven voor de complexiteit, relatief aan de omvang van het XML schema hanteert SIG de volgende berekening:

MCC_{rel} = NODES / MCC

Waarbij:

NODES: Het aantal XML Nodes in het schema. Hierbij is de definitie van een XML Node conform aan de definitie die gehanteerd wordt door het W3c[3].

MCC: Totale McCabe Cyclomatic Complexiteit, conform de definitie die gehanteerd wordt in Lammel et al[4].

[3] http://www.w3schools.com/dom/dom_nodes.asp

[4] Lammel, Ralf, Stan Kitsis, and Dave Remy. "Analysis of XML schema usage."*Conference proceedings XML*. Microsoft Atlanta, Georgia, 2005.

C Benchmark XML standaarden

C.1 Standaarden

C.1.1 Overheid standaarden

Standaard	Versie	Sleutel
iStandaarden Jeugdwet (iJw)	1.0	Istandaarden-ijw-1.0
Overheid.nl Web Metadata Standaard (OWMS)	1.0	Standaardoverheid-owms-1.0
Kamer van Koophandel HRIP	Juli 2013	Kvk-hrip-juli2013
Kadaster Informatiemodel	2.1	Kadaster-imkad-2.1
Kadaster Basisregistraties	20120201	Kadaster-brk-20120201
E-Logboek Visserij	3.0	Minlnv-ers-3.0
Openrechtspraak	1.0	Openrechtspraak-1.0
BKWI SuwiML Basis	0801-b01	Bkwi-suwimlbasis-0801b01

C.1.2 Internationale standaarden

Standaard	Versie	Sleutel
IPTC SportsML	2.2	Iptc-sportsml-2.2
Health Level-7 (HL-7)	2.5.1	Healthlevelseven-hl7-2.5.1
OASIS Content Management Interoperability	1.1	Oasis-cmis-1.1
OASIS Saml 2.0	2.0	Oasis-saml-2.0
SISO Coalition Battle Management Language	1.0	Siso-cbml-1.0
Apache ActiveMQ	5.11.1	Apache-activemq-5.11.1
OGC Keyhole Markup Language	2.2.0	Ogc-kml-2.2.0
W3c Scalable Vector Graphics	1.1	W3c-svg-1.1
W3c MathML	3.0	W3c-mathml-3.0

D Serializeren van *nil* elementen in Java en .NET

D.1 Serializeren met JAXB

D.1.1 Element “voornamen” geheel weglaten in geserializeerd bericht

```
private static NPSVraagScope createScope() {

    NPSVraagScope scope = new NPSVraagScope();
    scope.setVoornamen(null);

    return scope;
}
```

```
<NPS-vraagScope xmlns:ns2="http://www.egem.nl/StUF/sector/bg/0310" xmlns:ns1="http://www.egem.nl/StUF/StUF0301"/>
```

D.1.2 Element “voornamen” wordt als *nil* element geserializeerd

```
private static NPSVraagScope createScope() {

    NPSVraagScope scope = new NPSVraagScope();

    VoornamenE voornamen = null;

    JAXBElement<VoornamenE> e = jaxbElement(voornamen);
    e.setNil(true);

    scope.setVoornamen(e);

    return scope;
}
```

```
<NPS-vraagScope xmlns:ns2="http://www.egem.nl/StUF/sector/bg/0310" xmlns:ns1="http://www.egem.nl/StUF/StUF0301">
    <voornamen xsi:nil="true" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"/>
</NPS-vraagScope>
```

D.1.3 Element “voornamen” met StUF:NoValue wordt niet als *nil* element geserializeerd

```
private static NPSVraagScope createScope() {

    NPSVraagScope scope = new NPSVraagScope();

    VoornamenE voornamen = new VoornamenE();
    voornamen.setNoValue(NoValue.VASTGESTELD_ONBEKEND);

    JAXBElement<VoornamenE> e = jaxbElement(voornamen);
    e.setNil(true);

    scope.setVoornamen(e);

    return scope;
}
```

```
<NPS-vraagScope xmlns:ns2="http://www.egem.nl/StUF/sector/bg/0310" xmlns:ns1="http://www.egem.nl/StUF/StUF0301">
    <voornamen ns1:noValue="vastgesteldOnbekend"/>
</NPS-vraagScope>
```

D.2 Serializeren met .NET XMLSerializer

D.2.1 Element “voornamen” moet *nil* zijn.

```
private NPSvraagScope CreateScope()
{
    var scope = new NPSvraagScope();

    scope.voornamen = null;

    return scope;
}
```

```
<NPS-vraagScope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/
    XMLSchema">
    <voornamen xsi:nil="true" xmlns="http://www.egem.nl/StUF/sector/bg/0310" />
</NPS-vraagScope>
```

D.2.2 Element “voornamen” met StUF:NoValue wordt niet als *nil* element geserializeerd

```
private NPSvraagScope CreateScope()
{
    var scope = new NPSvraagScope();

    Voornamene voornamen = new Voornamene();
    voornamen.noValue = NoValue.vastgesteldOnbekend;
    voornamen.noValueSpecified = true;

    scope.voornamen = voornamen;

    return scope;
}
```

```
<NPS-vraagScope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/
    XMLSchema">
  <voornamen d2p1:noValue="vastgesteldOnbekend" xmlns:d2p1="https://www.egem.nl/StUF/StUF0301" xmlns="http://
      www.egem.nl/StUF/sector/bg/0310" />
</NPS-vraagScope>
```

E Voorbeeld: StUF-BG met REST koppelvlak

E.1 Koppelvlak definities

StUF-BG

```
<complexType name="NPSNINING-basis">
    ...
    <sequence>
        ...
        <element name="inp.a-nummer" type="xs:string" ... />
        ...
        <element name="inp.heeftAlsEchtgenootPartner" type="BG:NPSNPSHUW-basis" ... />
        ...
    </sequence>
    ...
</complexType>
```

REST Koppelvlak

```
<complexType name="NPSNINING-resource-basis">
    <complexContent>
        <restriction base="BG:NPSNINING-basis">
            <sequence>
                ...
                <element name="inp.a-nummer" type="xs:string" ... />
                ...
            </sequence>
            ...
        </restriction>
    </complexContent>
</complexType>


<complexType name="NPSNINING-resource">
    <complexContent>
        <extension base="NPSNINING-resource-basis">
            <sequence>
                ...
                <element name="inp.heeftAlsEchtgenootPartner" type="xs:anyURI" ... />
                ...
            </sequence>
        </extension>
    </complexContent>
</complexType>

<element name="nps" type="NPSNINING-resource" />
```

Relatie inp.heeftAlsEchtgenootPartner niet meegenomen in de restriction

Nieuwe definitie van inp.heeftAlsEchtgenootPartner met type "anyURI"

Voorbeeld root element

E.2 Ophalen van resource

F Voorbeeld: StUF-BG met JSON formaat

F.1 Voorbeeldbericht NPS bevraging

```
<complexType name="NPS-vraagScope">
    <sequence>
        ...
        <element name="voornamen"
                 type="BG:Voornamen-e"
                 nillable="true"
                 minOccurs="0"/>

    </sequence>
    <attribute ref="StUF:entiteittype" use="required" fixed="NPS"/>
    ...
</complexType>
```

Figuur F.1: XML Schema Definitie van NPS-vraagScope in resolved StUF-BG

```
public class NPSVraagScope {

    @XmlElementRef(
            name = "voornamen",
            namespace = "http://www.egem.nl/StUF/sector/bg/0310",
            type = JAXBElement.class, required = false)
    protected JAXBElement<VoornamenE> voornamen;


    @XmlAttribute(name = "entiteittype",
            namespace = "http://www.egem.nl/StUF/StUF0301",
            required = true)
    protected String entiteittype;
```

Figuur F.2: Java class representatie van NPS-vraagScope

```
{
    …
    “vraag_scope”: {
        “element” : {
            …
            “voornamen”: {
                “element”: {
                    …
                },
                “attribuut_is_nil” : false
            }
        },
        “attribuut_entiteit_type”: “nps”,
        …
    }
    …
}
```

Eigenschappen die behoren tot het omsluitend object worden in JSON opgenomen als een object genaamd “element”

Attributen die behoren tot het omsluitend object worden in JSON opgenomen als sleutel-waarde paar met prefix “attribuut”

G KING feedback op onderzoek SIG

Memo

Van	KING, Henri Korver
Aan	Software Improvement Group (SIG)
Afdeling	KING/E-Diensten
Onderwerp	Feedback op onderzoeksresultaten StUF-BG
Datum	25-8-2015
Bijlagen	

Inleiding

De Software Improvement Group (SIG) heeft in opdracht van gemeente Den Haag de bruikbaarheid van StUF-BG onderzocht. In het kader hiervan heeft SIG een interview gehad met Henri Korver (KING) en Maarten van den Broek (MessageDesign). SIG heeft vervolgens aan KING gevraagd de slides ("Presentatie KING Validatie.pdf") waarin de onderzoeksresultaten worden gepresenteerd te valideren alvorens het eindadvies op te stellen voor de gemeente Den Haag.

De feedback van KING staat op hoofdlijnen in dit document. Meer gedetailleerd lichten we de opmerkingen en aanvullingen graag in een gesprek toe aan de hand van de sheets.

De onderzoeksvraag

Het onderzoek richt zich op het toetsen van de "claims" van de gemeente Den Haag. Deze zijn

- StUF-BG is niet goed bruikbaar voor de programmeurs/beheerders
- De standaard tooling werkt niet met de geleverde contracten, doordat:
    - StUF-BG groot en complex is (C1a)
    - StUF-BG constructies bevat die niet voorkomen in andere bekende standaarden (C1b)
- De SOAP binding van StUF-BG is niet specifiek genoeg (C2)
- De betekenis van kennisgevingsberichten is afhankelijk van de context (C3)
- StUF-BG kan alleen gedeeltelijk geïmplementeerd worden door de gemeente Den Haag. Dit vereist extra afstemming met afnemers (C4)
- Implementatie van StUF-BG levert performance en beschikbaarheidsrisico's op (C5)

Reactie van KING

Hoge kwaliteit van interoperabiliteit en een goede toepasbaarheid van standaarden zijn voor gemeenten belangrijk om informatie uit te wisselen, in ketens te kunnen werken, de ICT markt goed te laten functioneren en een samenhangend applicatielandschap te realiseren.
Voor interoperabiliteitsstandaarden is het belangrijk dat niet alleen de semantiek van de informatie-uitwisseling eenduidig is maar ook dat standaarden juist en snel toegepast kunnen worden voor applicatie-integratie in softwareproducten.

Bij de ontwikkeling en het onderhoud van software moeten moderne en gangbare software-ontwikkelplatformen, tools en talen kunnen worden ingezet. KING juicht dan ook toe dat de gemeente Den Haag onderzoek laat uitvoeren naar de bruikbaarheid van de StUF-standaarden.

De onderzoeksvraag is gericht op StUF-BG en op het toetsen van de genoemde claims. De claims van Den Haag komen voor wat betreft complexiteit, toepasbaarheid en noodzaak voor afstemming overeen met het beeld van KING. Daarom heeft KING ca. 5 jaar geleden een nieuwe standaardisatie-aanpak opgezet. Daarin is onderscheid gemaakt in "Eindproduct", "Halffabricaat "Grondstof" standaarden en gradaties van interoperabiliteit. Eindproducten zijn scherper, minder complex en sneller en beter implementeerbaar en testbaar. In de regiegroep gegevens en berichtenstandaarden, waar de gemeente Den Haag ook aan deelneemt, sturen we op deze aanpak, de inbouw van eindproduct standaarden door leveranciers en vormgeven van nieuwe ontwikkelingen.

Het uitgevoerde onderzoek betreft de bruikbaarheid van StUF-BG. StUF-BG is een halffabricaat dat eerst geraffineerd moet worden tot eindproduct standaarden c.q. concrete koppelvlakken voor gegevensuitwisseling in een bepaalde toepassing of domein. Afgelopen jaren zijn voor veel gemeentelijke ketens eindproduct standaarden ontwikkeld of zijn in ontwikkeling. Deze zijn:

1. BAG-WOZ
2. BAG-GBA
3. Betalen- en Invorderenservices
4. Documentcreatieservices
5. Prefill eFormulierservices
6. RSGB Bevragingen (in ontwikkeling in samenwerking met Gemeente Den Haag)
7. Toezicht- en Handhavenservices
8. Wabo-BAG services
9. Zaak- en Documentservices
10. StUF-koppelvlak Jeugdzorg (CORV)
11. Regie- en zaakservices
12. StUF-EF
13. StUF-WOZ
14. StUF-HR
15. StUF-LVO
16. StUF-RIHa
17. MijnOverheid Lopende Zaken koppelvlak
18. StUF BAG-BGT
19. BRK (in ontwikkeling)

Deze eindproduct standaarden zijn minder complex en hiermee kunnen software ontwikkelaars direct aan de slag. Voor de meeste is ook compliance ingericht en is testinstrumentarium beschikbaar. Voor deze eindproduct standaarden hoeft een gemeente dus geen eigen maatwerk en beheer in te richten en zijn met 170 leveranciers afspraken gemaakt over adoptie van de voor hun relevante standaarden. Ontwikkeling en beheer vindt plaats door een tiental organisaties.

Onderstaande figuur geeft inzicht hoe de StUF-familie is opgedeeld in drie "productlagen" :

- grondstoffen (StUF 3.01, protocolbindingen),
- halffabricaten (StUF-BG 3.10, StUF-ZKN 3.10) en
- eindproducten (geoBAG, CORV, etc.).

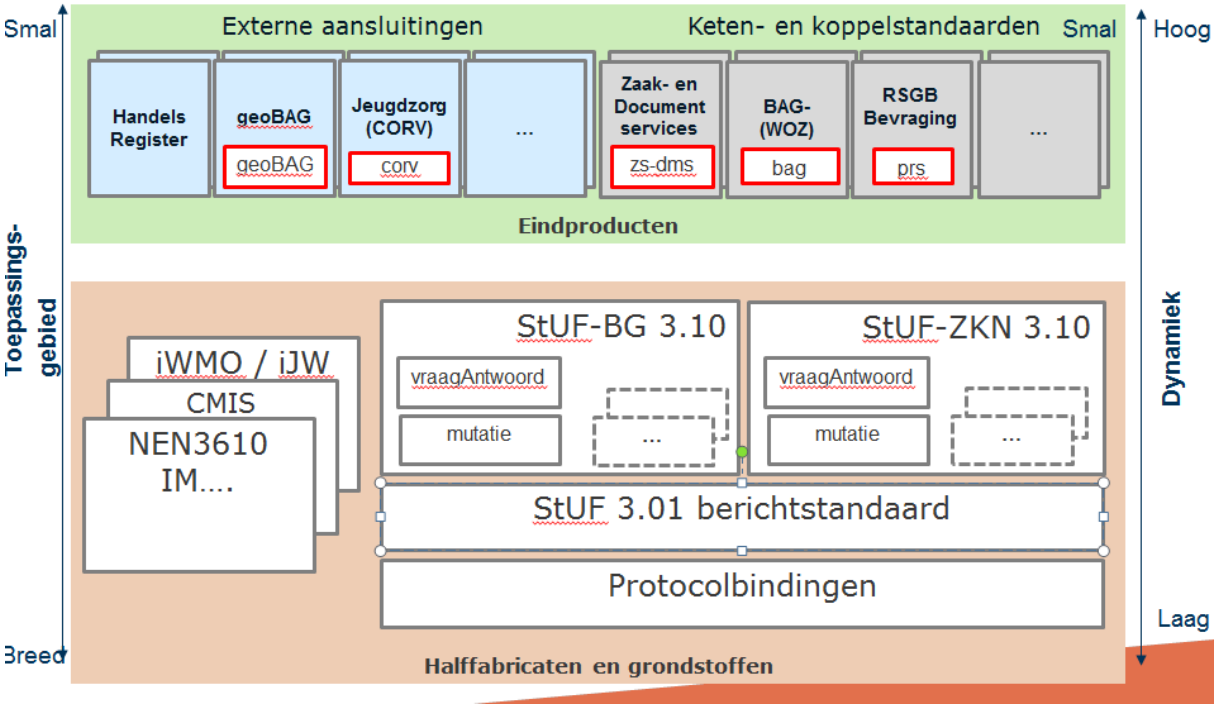


Figure 1: StUF-familie

In de familieplaat is te zien dat StUF-BG een halffabricaat is waarop vele concrete eindproducten (keten- en koppelvlakstandaarden) zijn gebaseerd. Mede op initiatief van gemeente Den Haag en in samenwerking met KING wordt gewerkt aan het opstellen van het op StUF-BG gebaseerde koppelvlak (eindproduct) "RSGB Bevragingen".

Dus het antwoord op de vraag: "Is StUF-BG (eenvoudig) bruikbaar voor programmeurs of beheerders?" is inderdaad negatief. Ons advies is de ingezette standaardisatielijn te volgen en StUF-BG niet direct te gebruiken maar altijd eerst te kiezen voor een functioneel passende eindproduct zoals Prefill eFormulierservices, RSGB Bevragingen, etc.

Omdat momenteel gewerkt wordt aan een nieuwe standaard “RSGB bevragingen” die gemeente Den Haag als één van de eerste zal gaan toepassen is een interessante aanvullende onderzoeksvraag om deze eindproduct standaard op bruikbaarheid te toetsen in relatie tot de claims van Den Haag.

Feedback op het onderzoek

In onderstaande tekst geven we op hoofdlijnen feedback op de vijf claims van Den Haag.

Ad De standaard tooling werkt niet met de geleverde contracten, doordat StUF-BG groot en complex is (C1a)

StUF-BG is inderdaad groot en complex omdat het een ruw product (halffabrikaat) is dat nog moet worden versneden tot kleine en eenvoudige koppelvlakken (eindproducten) zoals bijvoorbeeld geoBAG, BAG-WOZ, Prefill eFormulierservices, etc. Eindproduct-standaarden zijn qua aantal regels code een fractie van de omvang en complexiteit van StUF-BG en de eenduidigheid neemt toe. Standaard tooling (codegenerators Java, .NET/C#) is veel beter toepasbaar op deze eindproducten. Hoewel deze tooling ook bruikbaar is voor het generieke StUF-BG. Bijv. Centric genereert vanuit StUF-BG C#-code met de standaard tooling van .NET. Er waren wel een paar work-arounds nodig maar het schijnt verder goed te werken.

Ad De standaard tooling werkt niet met de geleverde contracten, doordat StUF-BG constructies bevat die niet voorkomen in andere bekende standaarden (C1b).
De restriction-constructie kan gerefactored worden met het Resolver-tool. De constructie nillable=”true” is zeer waardevol en wordt gebruikt in allerlei GML-standaarden. Met een paar eenvoudige work-arounds kan .NET (C#) met deze constructie goed uit de voeten. Uit de Java-community zijn tot nu toe geen klachten over deze constructie vernomen.

Ad De SOAP binding van StUF-BG is niet specifiek genoeg (C2).
Elke concrete toepassing van StUF-BG in een bepaald domein vraagt weer een specifieke invulling van het SOAP-protocol. Dus de verscherping van het SOAP-protocol zal ook hier weer op koppelvlakniveau moeten plaatsvinden. Ook hierin heeft KING recentelijk een initiatief genomen door het “Standaard koppelvlak Digikoppeling adapter intern” op te stellen. Dit is een standaard invulling van het SOAP-protocol gebaseerd op WS-Addressing voor applicaties die berichten via een Digikoppeling Adapter willen versturen.

Ad De betekenis van kennisgevingsberichten is afhankelijk van de context (C3).
StUF-BG is generiek en kent inderdaad geen context in de zin van gebeurtenissen (overlijden, verhuizing, vernummering, samenvoeging verblijfsobject, etc.). De gebeurtenissen worden in eindproduct standaarden gespecificeerd. Bijv. vernummering en samenvoeging verblijfsobject zijn gedefinieerd in het koppelvlak BAG. Daarin zijn de bericht en servicespecificaties functioneel en inhoudelijk aangescherpt.

Ad StUF-BG kan alleen gedeeltelijk geïmplementeerd worden door de gemeente Den Haag. Dit vereist extra afstemming met afnemers (C4).
Deze inperking vindt normaal gesproken plaats binnen eindproduct standaarden. Gemeente Den Haag zou prima kunnen aangeven welk gedeelte van StUF-BG ze hebben geïmplementeerd door in de WSDL’s van StUF-BG alle operations weg te laten die niet worden ondersteund. Binnen de operations die wel worden ondersteund kan met restriction worden aangeven welke elementen binnen de StUF-BG-berichten worden ondersteund. Door deze stappen te volgen heeft gemeente Den Haag haar eigen koppelvlak gedefinieerd op basis van StUF-BG. Voor de afnemers is het dan duidelijk wat het systeem wel of niet ondersteunt.

Ad Implementatie van StUF-BG levert performance en beschikbaarheidsrisico’s op (C5).
In StUF-BG zijn sorteringen gespecificeerd om zoekpaden te kunnen indexeren voor performance. Ook hier is StUF-BG een halffabricaat en moeten er in het uiteindelijke koppelvlak keuzes gemaakt worden welke sorteringen (indexeringen) wel of niet worden ondersteund. Als er een zoekopdracht wordt gegeven die niet wordt ondersteund zal er een foutmelding worden teruggegeven. Bijv. in het koppelvlak RSGB Bevragingen zullen de toegestane sorteringen en selecties uitgebreid gedocumenteerd zijn. Als de gekozen sorteringen op de juiste manier worden geïmplementeerd kunnen er geen performance- of beschikbaarheidsrisico’s optreden.

Voor feedback op detailniveau gaan we graag het gesprek aan met de deskundigen van Software Improvement Group (SIG) en de gemeente Den Haag.

Software Improvement Group
Amstelplein 1
P.O. Box 94914
1090 GX Amsterdam
The Netherlands

+31 20 314 09 50
info@sig.eu
@sig_eu
www.sig.eu

GETTING SOFTWARE RIGHT