

***Metadata and Annotation Two-Stage SSP Provisioning***

***GRA-UML***

**Version 0.1 (Draft for Review)**

|  |  |
| --- | --- |
| Delivery On: | **?** |
|  |  |
| Prepared For:  Prepared By: | **IJIS Institute**  **Model Driven Solutions**  **Open Networks, Inc.** |

Table of Contents

[Introduction 3](#_Toc388905667)

[Objective 3](#_Toc388905668)

[Problem statement 3](#_Toc388905669)

[Approach Overview 3](#_Toc388905670)

[Supporting artifacts 4](#_Toc388905671)

[Structure of metadata in GRA models 5](#_Toc388905672)

[Elements of the UML “Platform independent model” (PIM) 5](#_Toc388905673)

[Actors 5](#_Toc388905674)

[Use cases 6](#_Toc388905675)

[Interfaces 7](#_Toc388905676)

[Components and ports 8](#_Toc388905677)

[Collaborations & Interactions 8](#_Toc388905678)

[The GRA Service and Service Interface Specification 9](#_Toc388905679)

[Structure of the populated annotation metadata 12](#_Toc388905680)

[Execution of Phase-2 provisioning 14](#_Toc388905681)

[Appendix B: Revision History 15](#_Toc388905682)

Table of Figures

# Introduction

## Objective

This document specifies the GRA-UML approach to metadata and provisioning through use of model based annotations.

## Problem statement

GRA-UML has two goals that may be difficult to satisfy together:

* That GRA UML should produce a fully GRA conformant SSP
* That GRA-UML should provide flexibility in how SSPs are produced, what technologies are used and how these technologies are used.

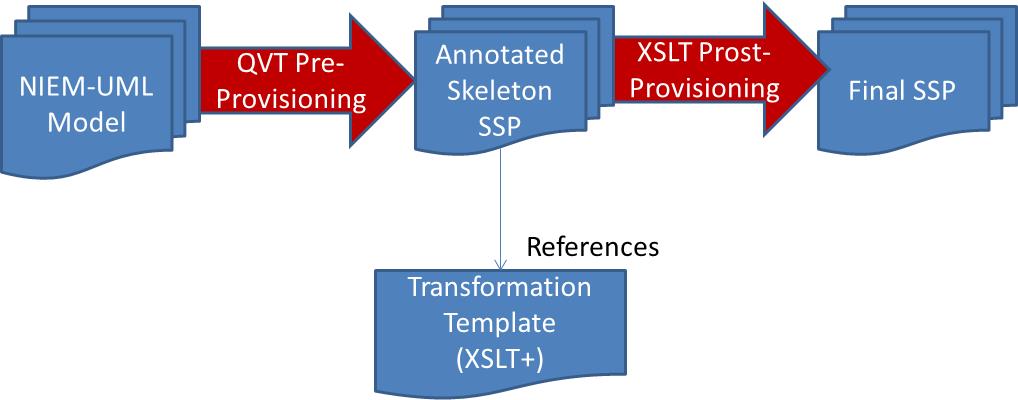
The reason for this flexibility as that GRA is essentially “open ended” with respect to the service interaction profiles used as well as what specific technologies are used within a SIP. In addition, SSP architects frequently have additional requirements or styls with respect to the generated documentation. This presented three alternatives that were not seen as advantageous:

1. That the SSP provisioning would provide for a fixed set of options without substantial extension capability
2. That the SSP would be modeled at the “technology level” and encompass all the complexity of WSDL, policies and other selected options
3. That developers would have to modify the QVT that produces a SSP – an unfamiliar technology to most dealing with SSPs and WSDL.

## Approach Overview

The approach outlined in this document is intended to provide a 4th alternative that provides for extensibility while retaining simplicity for the SSP developer. The approach introduces another kind of developer, the SSP template developer. The template developer uses familiar technologies (e.g. XSLT) to tailor SSPs to a particular style and set of technology choices.

The basis of this approach is to introduce “SSP annotations”. SSP annotations are metadata produced from the GRA-UML model that provide business and technology choices attached to a generated “skeleton” SSP. The provisioning process is divided into two segments: 1) The standard generation of a skeleton SSP with annotations using Model Driven technologies, specifically QVT (OMG Query View Transform standard). 2) A post provisioning process that tailors the provisioned SSP based on a user supplied XSLT transformation based on the skeleton SSP and XML annotations.



The QVT pre-provisioning is not intended to be modified; it is “fixed” in the GRA-UML standard as is the structure of the Skeleton SSP. The Skeleton SSP contains an annotation metadata file, essentially a metamodel of GRA. Elements of the annotation file are referenced from elements in the annotated SSP. The transformation template then reads the skeleton SSP and makes whatever adjustments are required based on that templates technology and stylistic choices. Any file in the SSP can be modified and other technologies may be called for complex transformation tasks. Template developers may also use the metadata to produce other artifacts in support of their GRA specification or development tasks.

Note: The exact method for connecting skeleton artifacts with the templates is TBD. It should be easy to use from XSLT. Assume use of NIEM Annotations.

The GRA-UML standard includes a baseline template that supports basic GRA web services. Templates can then add additional technology content, such as policy implementations. Policy implementations should be based on the “InteractionRequirements” as defined in GRA and modeled in the annotation metadata.

Within the metadata portion of a GRA-UML model the architect can select the template specification to use and may also assign specific templates for use at any level of the GRA model, this allows specific technology choices to be made at any level. For example, a template could be provided to support REST web services and select REST services for a specific SIP or interface.

XSLT was chosen for the post-processing steps to do its familiarity and popularity among developers who typically operate at the WSDL/SOAP and integration technology levels. XSLT is also a proven and standard technology. However, XSLT is not particularly suited for traversing complex models, so QVT is a better choice for the pre-provisioning step.

## Supporting artifacts

Please consider the following artifacts in support of this design document:

* GRAAnnotationModel – the GRA annotation metamodel. Information models are provided for generic GRA as well as the basic choices for the WSDL/Soap specific SIPs.
* GRAAnnotationIEPD – a NIEM-UML IEPD that imports the annotation model and defines the production of an NIEM IEPD for GRA annotations.
* GRAAnnotationIEPD – The generated IEPD with schema for the GRA annotation model. Instances of these schema are populated as artifacts in a skeleton SSP and provide the annotation data. The annotation data is referenced (as XML annotations) from all generated skeleton artifacts.
* Correction\_GRA\_Example – an example SSP using annotation metadata.
* Metadata.xml – Corrections metdata file that corresponds to the example (manually produced).

Note that we don’t yet have a fully populated GRAAnnotation.xml file.

## Structure of metadata in GRA models

GRA contains a substantial amount of metadata. The annotation approach augments the GRA metadata with additional “metamodel” metadata to support post-provisioning. The metamodel metadata maps well to the structure of WSDL. The GRA annotation model is a NIEM-UML model. Instances of this model are created in a GRA specification to define the metadata and annotations. Note that this is different than the approach taken in NIEM-UML where stereotypes are used for metadata. UML instances provide similar expressability as well as the ability to easily augment the annotations and model complex structures. UML instances also have a direct correspondence with the annotation metadata a XML file.

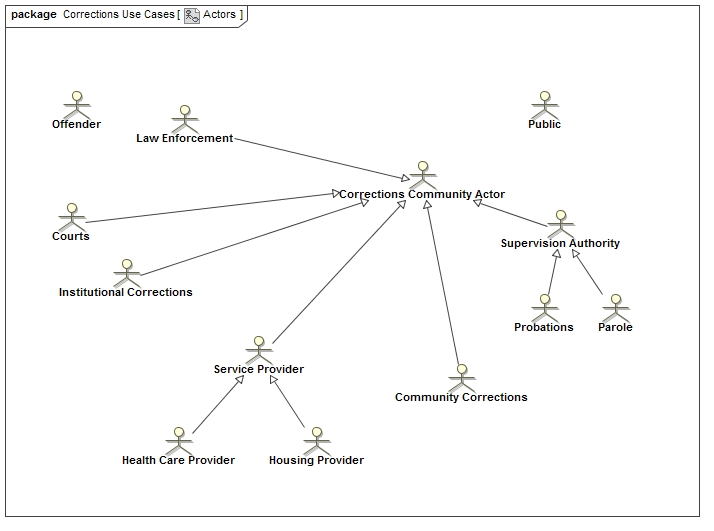
As a design goal “normal” UML structures that are typically used to define services can be used without any special stereotypes or undue rules. This allows the same UML elements to be used for purposes other than producing GRA specifications, such as producing implementations. It also provides for a separation of concerns between the logical services model and the GRA specific requirements and metadata. GRA metadata elements *realize* the UML elements that implement the logical model (Realization is a standard UML construct that shows as an arrow between elements).

## Elements of the UML “Platform independent model” (PIM)

GRA\_UML is designed to interpret specific UML constructs. These constructs are in keeping with the SoaML services modeling standard but use of SoaML stereotypes is optional. The following elements have meaning to GRA-UML and are shown with preliminary examples:

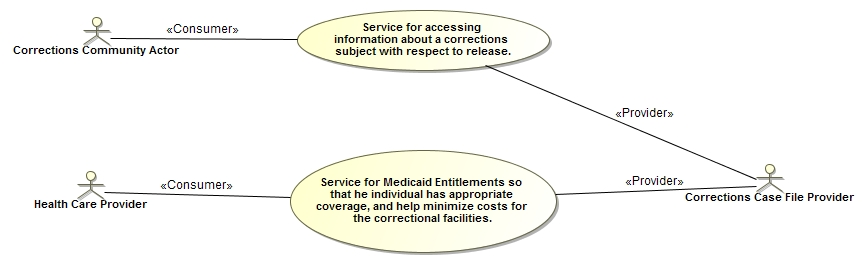
### Actors

Actors are the business participants in the SSP’s subject community. They are typically people or organizations.



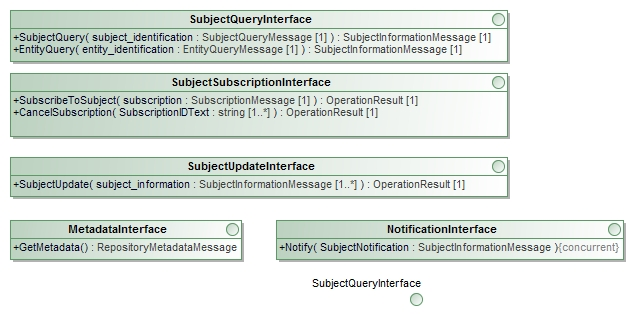
### Use cases

Use cases define the “real world effect” of services as use cases involving actors. Each use case (oval) is a real world effect and a business interaction.



### Interfaces

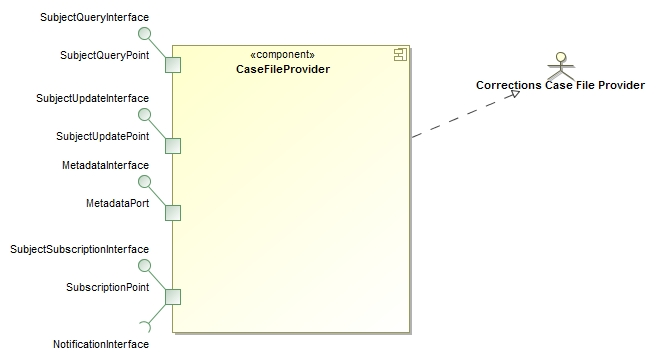
UML interfaces define the operations and signal receptions that service providers implement to expose a service.



The arguments of operations directly reference NIEM-UML message types as their content. (note: may need to reference non NIEM-UML parts).

### Components and ports

UML components represent the service provided by an actor. Each component has a set of ports that provide or use UML interfaces.

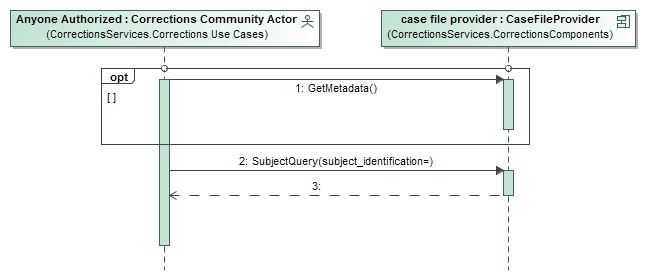


### Collaborations & Interactions

Collaborations are used to define the “community” for which the SSP is intended, providing a property for each actor and service component playing a role in that community.



Under the community collaboration is a set of interactions that define the choreography of exchanges between the actors to implement the use cases.



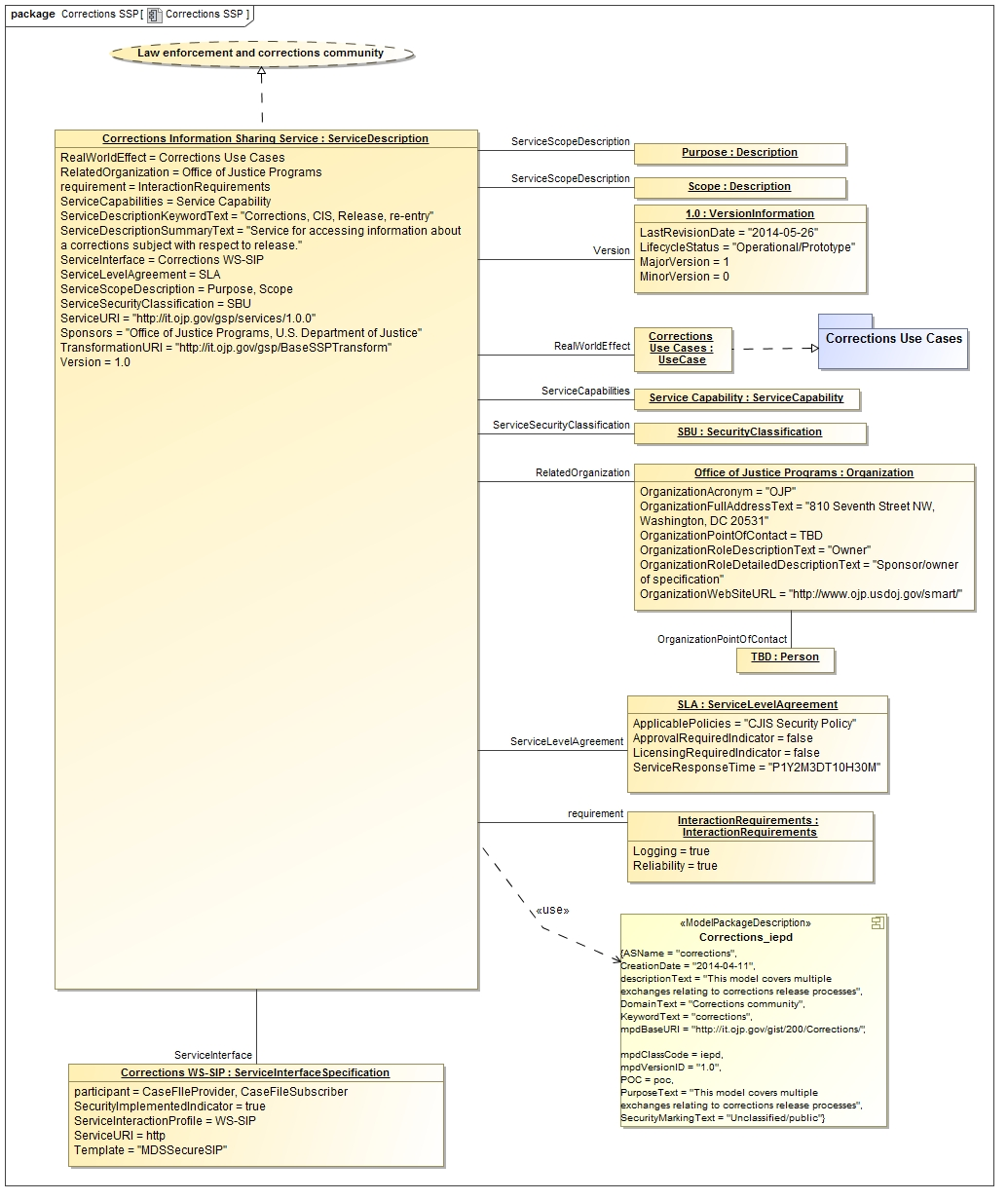
## The GRA Service and Service Interface Specification

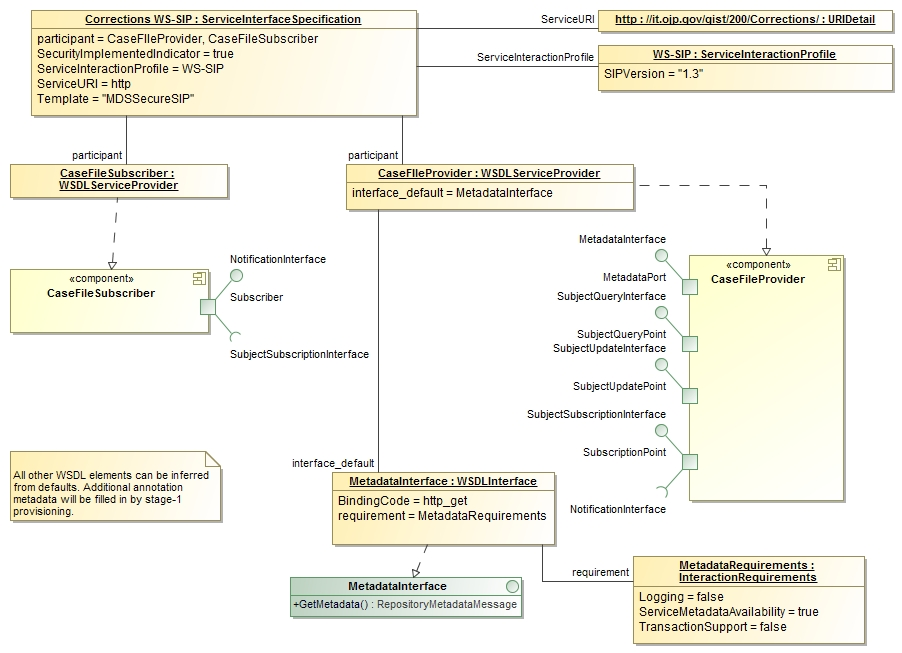
The GRA top-level services are defined as instances of GRA annotation metadata classes. Where appropriate these classes realize the UML element that provides the foundation for that GRA element.

The GRA metadata classes are used at 2 levels:

1. As the specification in the users model where only information that can’t be determined from the UML PIM or template defaults is provided.
2. As a fully populated annotation metadata file. The Stage-1 provisioning process “fills out” the mission annotation metadata elements and generates the annotation metadata instance file. The Stage-1 provisioning also annotates each element in the generated SSP with the metadata annotation element that support it, this provides the stage-2 provisioning with sufficient information to tailor each element and create the final SSP.

The following is an example SSP and Service interface specification using this approach. Note that detailed descriptions are contained in the UML elements documentation and are not currently shown on the diagram due to their size.



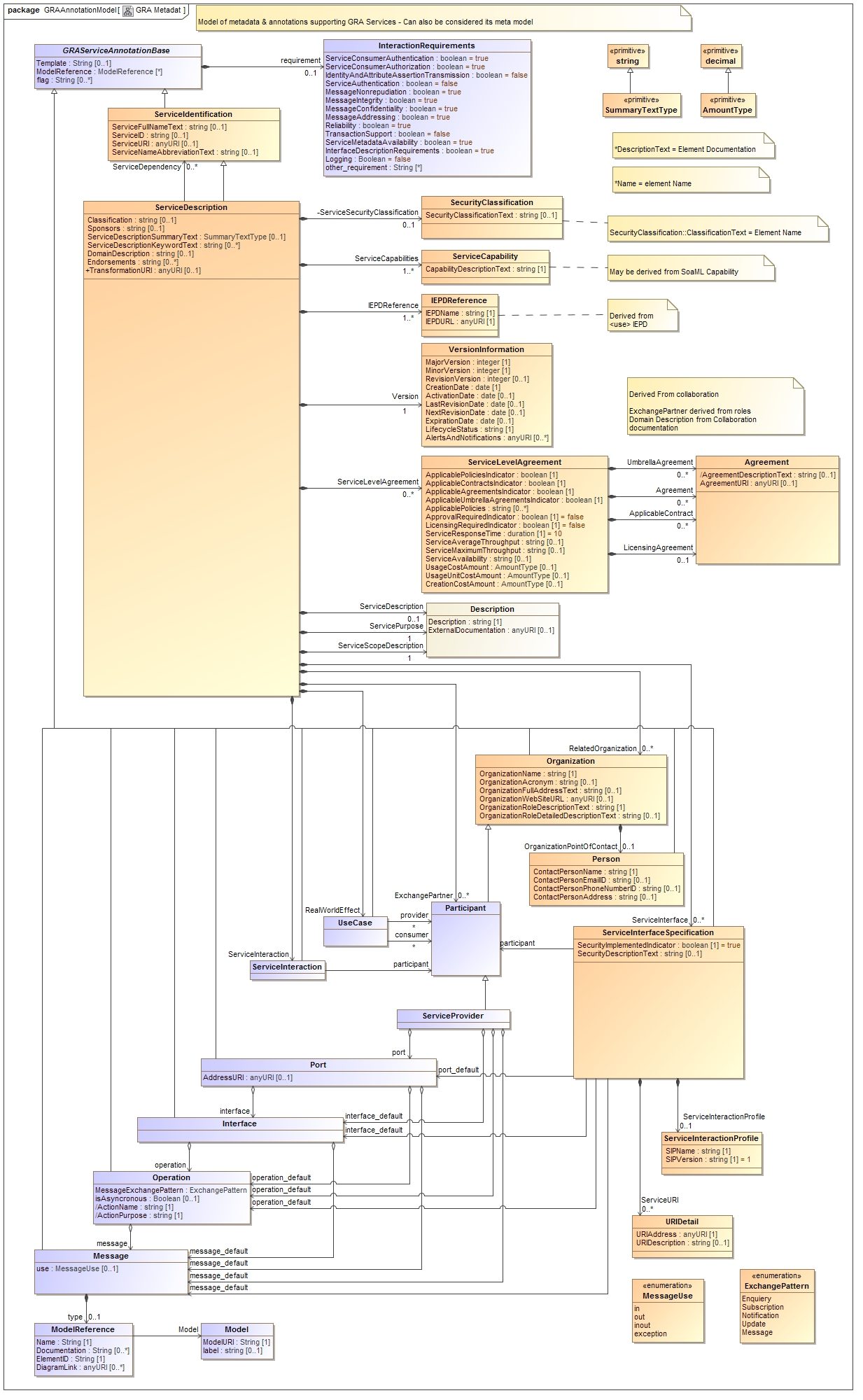


As stated above, the model only contains the information that can’t be inferred from the model or template. This includes the GRA “business information” as well as any specific technology choices. For example, in the above a technology choice was made to use “http\_get” to implement the metadata interface instead of the default (SOAP). The properties in the above metadata elements corresponds very directly to the GRA metadata XSD.

Besides the standard metadata the GRA architect has the option of providing additional parameters to further customize the phase-2 transformation. These are the “template” and “flag” properties of each metadata element. The “Template” parameter “calls” a specific template in the transformation instead of using the default matching. This allows complete control over how a element is interpreted be evoking custom new tempting logic. The “flag” parameter defines a simple tag in the metadata that can signal, the template to a user specified option. The template is then expected to process the element conditionally based on the flag. Unknown flags are ignored. (note: Support “mode” as well?)

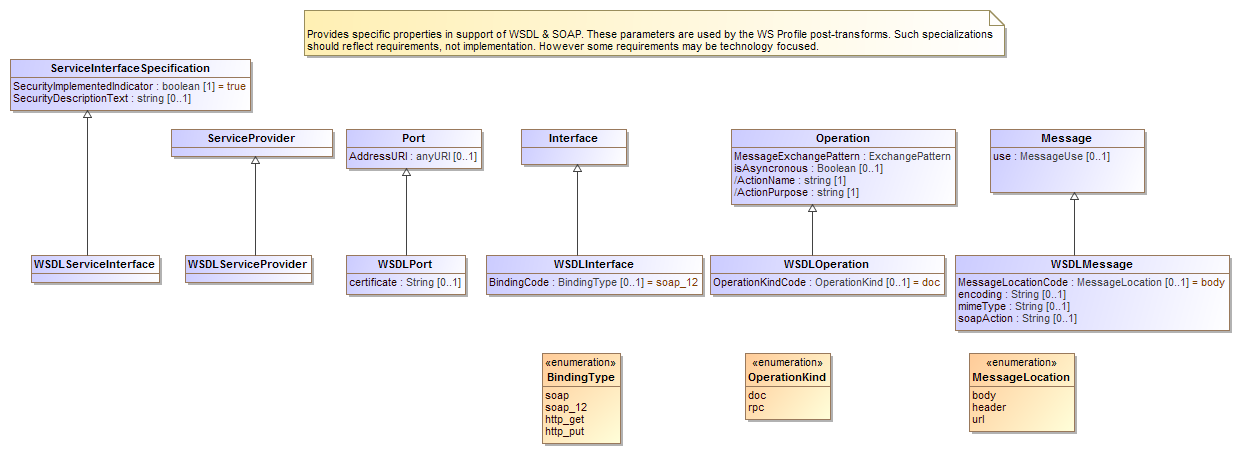
## Structure of the populated annotation metadata

While the modeler only needs to specify non-default information, the Pahse-2 transformation has access to the complete GRA “meta model” as annotations. This full meta model and metadata structure is found in the SSP under “artifacts/annotations.xml” and is produced by phase-1. The structure of the complete annotation file is defined in UML and is also available as an XML schema. This structure is:



Note that the orange classes are primarily defined by the GRA metadata structure where as the blue classes extend this structure to include information required to populate and extend the technology artifacts (e.g. WSDL). Note that the modeler MAY define any of the metadata elements, any elements not defined are populated from the logical UML model.

The above model defines the metadata for “generic” SSPs, however most SSPs use WSDL and perhaps SOAP. Specific extensions to the metadata classes are provided to define WSDL/SOAP specific choices. Use of these extension is optional, defaults may also be used. The WSDL/SOAP extensions also illustrate how template developers can introduce specific metadata elements for their technology choices. The following model defines the WSDL/SOAP metadata classes.



## Execution of Phase-2 provisioning

As part of the GRA specification a Transformation template is specified, in the example case it is “http:it.ojp.gov/gsp/BaseSSPTransform”. This is an XSLT transform that is executed on each artifact of the skeleton SSP to produce the final SSP. The skeleton SSP combined with the annotation metadata provide the parameters for these transforms. There are essentially no rules about what can or cannot be done in a phase-2 transformation, if a user transformation is specified it is the responsibility of the provider to ensure that a GRA conformant SSP is produced.

# Appendix B: Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| Date | Version | Editor | Change |
| 5/26/2014 | 0.1 | Cory Casanave | Initial rev |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |