TASC Technical Specification Development

Policy and processes for developing and communicating maturity of GA4GH Technical Specifications

Document Purpose

The GA4GH is developing data exchange **standards** for federated genomic data sharing. To address this, new **technical specifications** are required, such as the VRS standard, which must be developed and iterated upon through application across community **implementations**. This creates a tension between the need to create **products** with enough stability for initial community adoption, while ensuring that they can evolve with minimal disruption to interoperate smoothly across a diverse set of genomic data resources. Mechanisms for communicating the stability, uptake, and development of technical specifications are therefore of paramount importance to addressing this balance.

A maturity model is a useful mechanism for communicating varying stability across **product features** (e.g. **data classes** or **protocols**) of a GA4GH standard. This is needed to help data producers at each stage of the adoption lifecycle (**Figure 1**) decide on the appropriate time to engage and implement the standard. Product features that have progressed through the maturity model should have an associated progression of support from the GA4GH specification maintainers for message generation, translation, and validation tooling.

The purpose of this document is to clearly define the maturity model and release process for developing and maintaining GA4GH standards, with the goal of enabling timely specification adoption by the community.

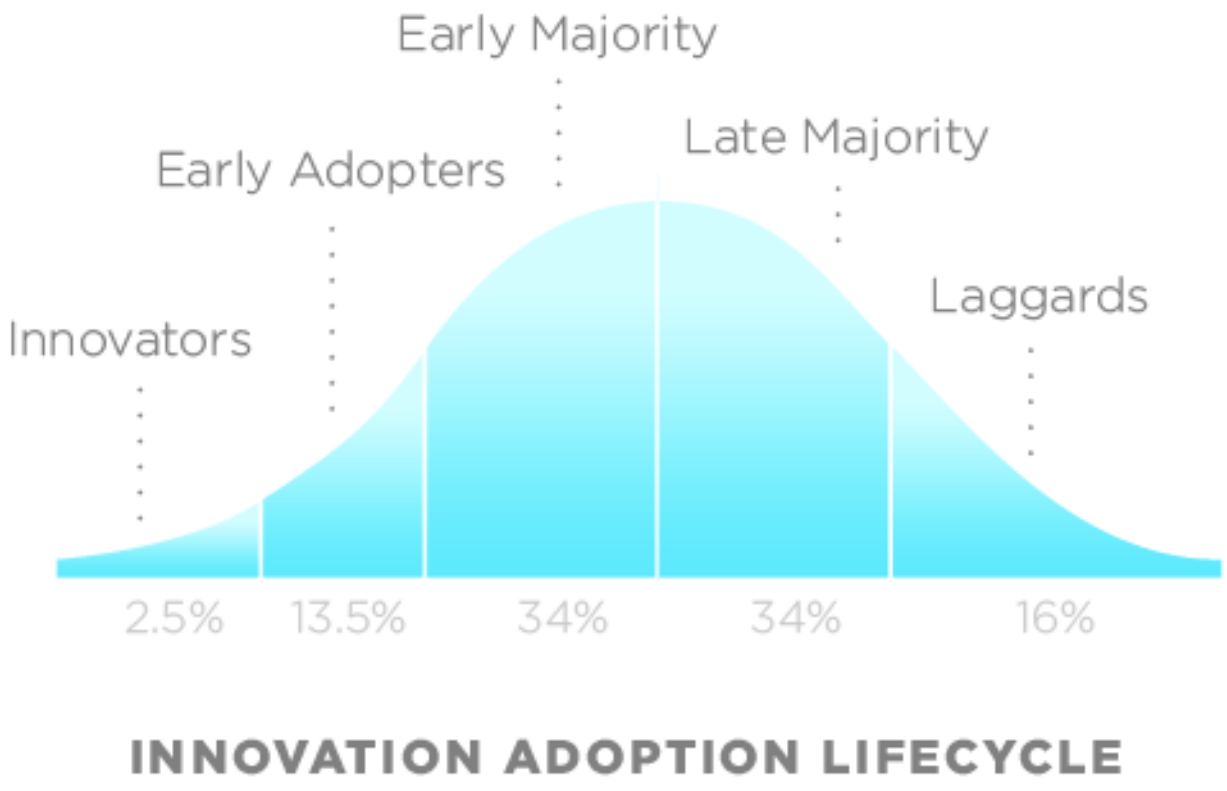

Figure 1 - The Innovation Adoption Lifecycle (**source**)***.*** *The Innovation Adoption Lifecycle illustrates adoption rates (y-axis) for new technologies over time (x-axis). Innovators (leftmost on the time axis) are among the first to adopt a new technology, and laggards (rightmost) are among the last, reflecting the differing needs for innovation and stability by these*

community groups. Adopters in every category along the innovation adoption lifecycle benefit from communication about the maturity of technical specification components generated in GA4GH technical products. Communicating when a component is ready for implementation by groups along the innovation / stability spectrum is a primary goal of the maturity model, enabling adopters to engage at a time that is appropriate for their organizational needs.

Alignment with the GA4GH Product Development Processes

Developing a new GA4GH standard would still require that the product go through the full GA4GH Product Development and Approval processes. This document complements those processes by providing a maturity, release, and versioning strategy that may be referenced by the product proposal for developing, maintaining, and extending standards using a maturity model for product features. The processes outlined here may also be applied to technical specifications that support downstream products but are not themselves GA4GH Standards (e.g. the **GKS Common Library**).

It is not expected that every category of product feature developed for a GA4GH technical specification will be annotated with the maturity model (e.g. validation tests or documentation appendices would likely not be annotated with maturity), though some categories (e.g. data classes and protocols) are expected to always be annotated with maturity levels. Generally, if a product feature would be sufficient for a major or minor version increment in the absence of a maturity system, it should be annotated with a maturity level using this system.

Feature Maturity Levels

Maturity Level Criteria

Level	Criteria	Specification Changes	Support
Draft	A requirements survey has been done, and a draft product feature has been developed and documented.	Not considered safe for stable implementation, changes may occur at any time and are published for evaluation by the implementation community.	Technical artifacts, documentation, and validation tests are produced and made publicly available.
Trial Use	The product feature has been reviewed by multiple data providers. It has been evaluated by the community and endorsed as ready for widespread testing.	Expected infrequently. Changes are made in consultation with all participants in the product group.	At least two reference implementations using the product feature, one of which must be open.
Normative	The product feature has been sufficiently tested to warrant long-term support from the specification maintainers. This is considered stable.	No backwards-incompatible changes will happen within the released major version.	The product feature will be fully supported by product reference implementations.
Deprecated	The product feature was previously released at a trial use or normative maturity, but will be discontinued in a subsequent version.	Use of the product feature will be discontinued in a subsequent minor version (for Trial Use maturity) or major version (for Normative maturity).	This product feature will no longer be supported once removed from the specification.

Table 1 - Product feature maturity level criteria and commitments.

Maturity Advancement Process

Product feature maturity levels are to be reviewed and advanced by consensus among defined decision-makers following Work Stream and GA4GH processes, in consultation with the associated **product group** membership. Factors to be considered for product feature maturity advancement include the criteria

specified in **Table 1**, the degree of adoption observed in the community, feedback provided by adopters, and availability of specification maintainers to provide the level of support required.

Developing a Draft Product Feature

Decision-makers: **Feature developers**, **product owners**

Criteria: *Draft* product feature development work should be based on real use cases across multiple environments (aligned with **GA4GH Product Development 14.5**). Requirements may result directly from a **landscape analysis of the problem domain**, or may emerge in the course of technical specification development. It is expected that the need for product features are first discussed in a community forum (e.g. GitHub Discussions, product group calls).

Process: Follow **the GA4GH product feature development process**. As part of this process, it is expected that consensus among the decision-makers was reached and major design decisions documented. Disagreements are resolved per Work Stream and GA4GH processes.

Advancing from Draft to Trial Use

Decision-makers: **Feature developers**, **product owners**, **product implementers**

Criteria: Advancing a *draft* product feature to *trial use* should include at least two independent product implementers that commit to supporting the *draft* product feature once it has been advanced to *trial use*. At least one of these implementations must be open (aligned with **GA4GH Product Development 14.8.3**). Advancing a product feature to *trial use* also mandates a minor version increment at the next **release**. As part of this process, it is expected that consensus among the decision-makers was reached and major design decisions documented. Disagreement resolution is handled per Work Stream and GA4GH processes.

Process: A **ballot release** is created that describes draft models under evaluation for advancement to trial use. A survey is sent to all **Product Implementers** that have indicated they are implementing one or more features under evaluation for advance to Trial Use. This survey includes:

1. Name of **Product Implementer**
2. Selection of a previously described implementation
3. If (or if multiple, which) product feature(s) are suitable for advance to Trial Use
4. Comments on response (e.g. explicit endorsement or description of gaps)

There is a minimum 1-week review period for Product Implementers to respond, though this may be longer at the discretion of the product owners. More time for individual contributors may be permitted on request.

Advancing from Trial Use to Normative

Decision-makers: **Feature developers**, **product owners**, **product implementers**, **Work Stream leads**

Criteria: A *normative* model should have demonstrated interoperability of multiple data generation and data consumption implementations, and should include implementations beyond those used to advance a model to Trial Use. Advancing a product feature to *normative* also mandates a minor version increment at the next **release**. As part of this process, it is expected that consensus among the decision-makers was reached and major design decisions documented. Community consultation and disagreement resolution are handled per Work Stream and GA4GH processes.

Data Class Inheritance and Property Maturity

Data models may (and often do) include child **data classes** that inherit properties from a parent data class. For example, the **Entity** data class from the GKS Common Library provides shared properties (e.g. *id*, *label*,

extensions) that are inherited by several child data classes across the VRS and Variant Annotation specifications.

To address inheritance used in a data model, we place additional constraints on the maturity of data classes and their properties. First, child data classes may not have maturity levels greater than the upstream data classes they inherit from. Second, the properties of a data class may not exceed the maturity of the data class as a whole. Together, these rules ensure that attention to the maturity of upstream classes is addressed first, and that less mature data classes do not artificially convey stability (in whole or in part) through inheritance of properties of more mature data classes.

These rules also allow for extending more mature data classes with new properties that exist in a less mature state.

Communicating Maturity Level

Minimally, primary documentation sites (e.g. **vrs.ga4gh.org**) will annotate data classes, data class properties, protocols, and other important documentation with their corresponding maturity levels.

In JSON Schema, this is accomplished using the maturity property for data classes (see the JSON Schema **maturity annotation for the VRS Allele** class) or data class properties (see this JSON Schema **property-level maturity annotation in the VA-Spec Cohort Allele Frequency** profile).

Product Feature Development Process

Development of GA4GH standards involves a community-oriented process that iterates on the general pattern of:

1. Discuss Issues
2. Gather Requirements
3. Propose Solutions
4. Develop Product Feature

Discuss Issues

Emerging discussion topics should first be created as Discussions in the repository of the associated GA4GH product (e.g. the VRS Discussion board at **github.com/ga4gh/vrs/discussions**). The **product owners** monitor these discussions and coordinate their addition to the agenda on community calls.

Gather Requirements

On a community call, the discussion topic is first announced as a future call agendum by the **product owners**, and a request made for asynchronous discussion and community-driven requirements gathering on the GitHub Discussion thread.

As part of requirements gathering, the community is surveyed for interest in implementing the feature, and any potential product feature implementers are expected to provide:

1. Name of **Product Implementer**
2. A description of their **Implementation**:
    a. A short description of the implementation that will use / is using the standard
    b. If the implementation is maintained by a Driver Project
        i. If so, which?
    c. If the implementation is open / public
3. If (or if multiple, which) product feature(s) will be used by the implementation
4. If the implementer will contribute **product feature developer** effort

An example form for collecting this information is the **GKS Product Implementer Form**.

It is expected that there will be at least two product implementers supporting a product feature, and that at least one implementation will be open. It is also expected that at least one product implementer will contribute product feature developer effort.

Once the above criteria are met, the development process may advance.

Propose Solutions

Product implementers should propose solutions on the GitHub Discussion thread. On a subsequent call, the topic is raised for review of requirements and discussion of proposed solutions. Action items may include advancing to solution implementation, furthering investigation of requirements, or continued discussion on a subsequent call. When one or more solutions are identified as ready to advance to implementation, a GitHub Issue is created (e.g. the VRS Issue board at **github.com/ga4gh/vrs/issues**) and assigned to one or more **feature developers**.

Develop Product Feature

The assigned **feature developers** will develop the product feature on a separate feature branch reflecting the associated GitHub Issue, and make a Pull Request for community review. The **product owners** are responsible for review and recommend action on Pull Requests within 2 weeks. Once merged, the product feature is developed.

Specification Releases and Versioning

Versioning

Versions are used to identify releases of technical specifications, *not* to individual product features.

Technical specification development is intrinsically linked to policy surrounding major and minor version identification, which follow semantic versioning v2 (SemVer; **semver.org**) practices for API versioning. Version syntax follows SemVer syntax. Examples of how product features at different maturity levels are applied to the SemVer major/minor/patch syntax as follows:

Major Version Increment

- Backwards-incompatible changes to a *normative* product feature
- Backwards-incompatible changes to property names of a previously-released *normative* data class
- Backwards-incompatible changes to the definition of a previously-released *normative* data class
- Backwards-incompatible changes to the digests of previously-released *normative* data class (as applicable)
- Addition of required fields to previously-released *normative* data class

Minor Version Increment

- Backwards-incompatible changes to a *trial use* product feature
- Addition of optional fields to data models at the *trial use* or *normative* level
- Release of a new product feature at the *trial use* or *normative* level
- Backwards-incompatible changes to property names of a previously-released *trial use* data class
- Backwards-incompatible changes to the definition of a previously-released *trial use* data class
- Backwards-incompatible changes to the digests of previously-released *trial use* data class (as applicable)
- Addition of required fields to previously-released *trial use* data class

Patch Version Increment

- A new product feature at the *draft* maturity level

- Any changes made to *draft* product features
- Addition of implementation guidance, tests, or other supporting product features that do not directly affect data compatibility

Versioning of approved GA4GH standards should additionally follow the procedures for **GA4GH Product Updates**. Specifically, advancement of data classes to the *trial use* or *normative* levels must be accompanied by a minor release increment, and therefore may only be included in a release following an appropriate community and PRC consultation process (**GA4GH Product Development 32**).

Releases

A **release** of a technical specification contains all of the content of the specification repository. This includes all features, including **data models** (source and derived artifacts), linked upstream dependencies, documentation, implementation guidance, validation tests, and examples. Releases provide a comprehensive and static snapshot of a technical specification that may be referenced for adoption by downstream products and implementations.

Pre-releases

In order to support continuous development of a technical specification, pre-release snapshots are allowed and must use the SemVer syntax for pre-releases. Pre-release snapshots may be created for purpose at any time by the product leads. Examples of pre-release snapshots following this process may be found in the VRS repository.