API4INSPIRE
D1 - Methodology for evaluation of standard based APIs

Fraunhofer
IOSB

Version 1.1
2020-02-26

Contents

Tables

Figures

1. Definitions and Acronyms

1.1. Acronyms

API: Application Programming Interface

CSV: Comma Separated Values

DB: Database

desktop GIS: Geographic Information System running on a desktop

DWBP: W3C Data on the Web Best Practices

ELISE: European Location Interoperability Solutions for e-Government

FROST: FRaunhofer Opensource SensorThings-Server

GeoJSON: An open standard format designed for representing geographical features.

GIS: Geographic Information System

HTML: HyperText Markup Language

INSPIRE: Infrastructure for Spatial Information in the European Community

INSPIRE MIF: INSPIRE maintenance and implementation framework

INSPIRE MIG: INSPIRE maintenance and implementation group

JSON: JavaScript Object Notation

MS: Member State

OGC: Open Geospatial Consortium

OGD: Open Government Data

OWS: OGC Web Services

REST: Representational state transfer

SF-X: Simple Features Level X

SOAP: Simple Object Access Protocol

SDK: Software Development Kit

SDWBP: Spatial Data on the Web Best Practices

STA: SensorThings API

SWG (OGC): Standards Working Group

W3C: World Wide Web Consortium

WFS2: Web Feature Service version 2

XML: Extensible Markup Language

1.2. Definitions

Administrative Stakeholders: Those responsible for providing resources for the provision and use of APIs

Data Providers: Organisations that make data available.

Data Users: People or organisations that use data made available by data providers.

Degrees of Freedom: directions in which recommendations can lead to concrete impacts and improvements

Evaluation Criteria: individual criteria derived from the Five Level Evaluation Model by Moilanen

Evaluation Types: methods of interrogating stakeholders: Heuristic Expert Evaluation, Peer Review, Interview, Questionnaire

Five Level Evaluation Model: API evaluation model by Jarkko Moilanen

Peer Review: evaluation approach for API development evaluation defined by Farooq

Stakeholder Perspectives: different stakeholder roles: development, deployment, usage

2. Introduction

This document reports on the methodology used to evaluate the introduction of standard based APIs into a wider administrative data provision framework as already established within the European Community. The methodology is generic and reusable for various stakeholder groups including both data providers and data users. The methodology includes the criteria deemed as relevant for evaluation of the transition towards the newly proposed OGC APIs, namely the OGC-API Features, and for dynamic data, the SensorThings API for both data providers and users, as well as methods for gaining insights pertaining to these criteria.

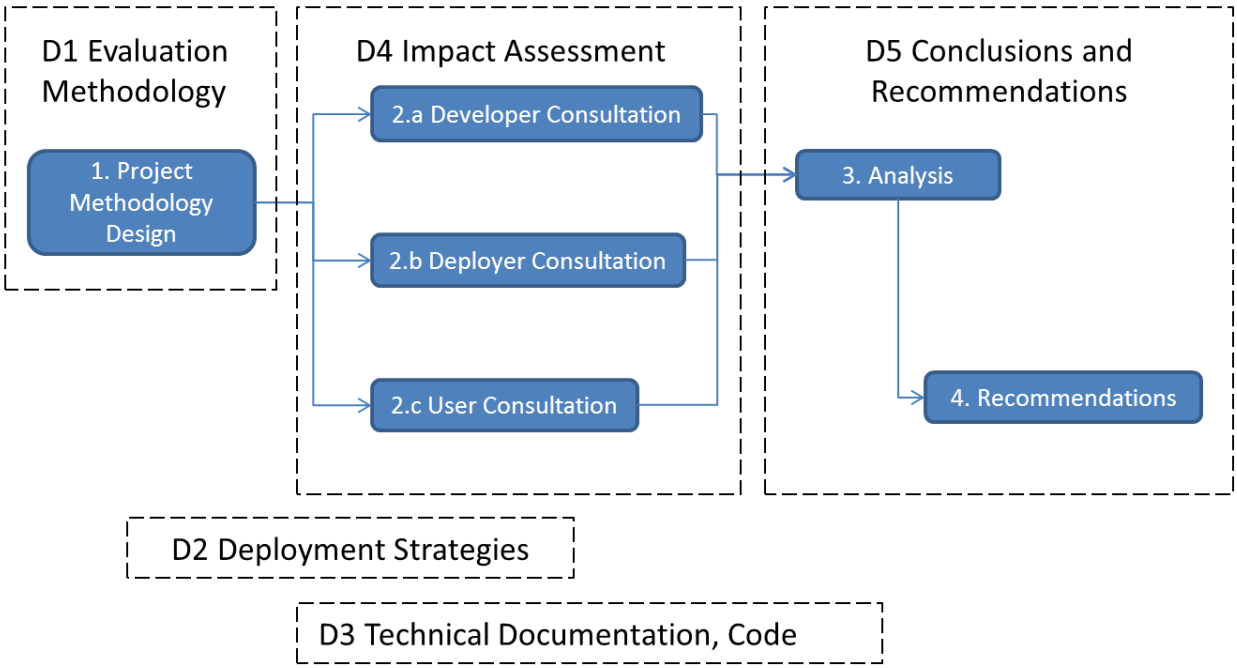
The evaluation methodology described has been designed to weigh costs and benefits against each other, highlighting both strengths and weaknesses of the APIs being evaluated. Pertaining to benefits, this includes flexibility, developer friendliness, and ease of discoverability, access and use. From a technical point of view, alignment with the current architecture of the Web and the Spatial Data on the Web Best Practices should be assured. Cost considerations, such as infrastructural changes, need for additional expertise, updated tooling, training, reengineering of existing practices and security aspects shall be included. While quantifiable metrics would be preferable, the effort entailed in gaining truly representative values would require a level of complexity that outweighs the assumed benefits of such quantification; thus we have chosen to focus on qualitative metrics that can be easily abstracted to different operational environments.

The outcomes of this evaluation will provide guidance for the diverse stakeholders involved in this complex data ecosystem that incorporates such initiatives as INSPIRE. This includes the different roles involved in data provision and use, ranging from administrative stakeholders tasked with securing the necessary resources for deployment of APIs over domain experts concerned that their data is handled and provided correctly to developers designing new applications based on the available APIs. The ecosystem further encompasses not only the governance aspects entailed in maintaining and updating such an infrastructure, but also enabling community initiatives and platforms, as well as the standards bodies defining and updating the underlying standards.

2.1. Overview

In this section, we provide an overview of the evaluation methodology, the subject of this deliverable. At the same time, we show how this methodology is embedded within the wider context of the evaluation process. In Figure 1: Evaluation Process Overview we illustrate how the individual steps of the evaluation process are aligned with the project deliverables.

Figure 1: Evaluation Process Overview



In an initial step, we have designed the project methodology. Relevant stakeholder groups have been analyzed, evaluation criteria defined and evaluation methods specified and described. Based on this foundation, the evaluation process has been formulated. This is the topic of the current document “D1 Methodology for evaluation of standard-based APIs”.

In parallel with the design of the project methodology, work on deployment strategies has commenced with the various data providers linked with this project. Based on the available data resources potential use cases are being defined, guiding our selection of data sources to be exposed, whereby we aim to assure collections of related data assuring spatiotemporal consistency. This work will be documented and provided as “D2 Deployment Strategies”.

Based on the deployment strategy, we shall begin implementing and deploying both the OGC API – Features and SensorThings API together with our data providers. During this process, feedback on issues encountered will be documented, in order to better understand what support data providers are currently missing; this feedback will provide complementary information to that collected through questionnaires and interviews during the formal consultation process. All technical documentation together with source code and operational prototypes will be made available as “D3 Technical Documentation, Code”.

The impact of the introduction of the new APIs will be investigated through stakeholder consultations. The methodology described in this document will be utilized in order gain insights into the experiences of the developers, deployers and users involved. This process will be staggered, with developer and deployer consultations taking place earlier on in the process as the APIs are being made available. Once

the APIs are openly available, users will be invited to experiment with these resources in the context of various events and further feedback gained. This work will be documented in "D4 Impact Assessment".

After the consultations of the impact assessment have been finalized, the outcomes of the assessment will be analyzed together with inputs gained during the development and deployment process to provide insights into the strengths and weaknesses of the APIs under evaluation. As part of this analysis, effort required for API deployment will be compared to that required for the provision of more traditional services; a similar comparison will be performed pertaining to data usage aspects. Recommendations will be provided both towards the stakeholder groups identified on how to make best use of their resources for API provision as well as to the responsible standardization bodies on desired extensions to the API standards. All materials identified as relevant for the development, deployment and use of the OGC APIs will be collected and structured on the project GitHub[1]; the content can later be transferred to an alternative platform if required. These resources will be extended in order to provide a solid foundation for webinar participants to rapidly acquire the skills required to both provide and utilize APIs. All outputs will be provided as "D5 Conclusions and recommendations for MS authorities".

2.2. Goals

In order to gain a better understanding of the evaluation target, we must first clearly specify the goals of the evaluation process. The high level evaluation goal pertains to gaining a better understanding of the impacts and benefits of both

i. establishing the emerging APIs as valid INSPIRE download services as well as
ii. where deemed appropriate based on the underlying data models and foreseen use cases, introducing simplifications within the INSPIRE data models as foreseen within the INSPIRE MIF activity 2017.2 on alternative encodings within the INSPIRE and beyond domain.

Based on these two high level targets, we began the process of refining these to their component parts. This analysis provides an overview of the "degrees of freedom" available within this complex data ecosystem, which in turn provides a robust framework for weighing costs and benefits pertaining to each of these dimensions, and providing well-founded recommendations.

An evaluation is always a comparison - the important question is against what explicitly are we evaluating? Based on the wider context, it became clear that the introduction of the OGC API – Features and SensorThings API together with potential data model simplification must be compared to the existing SOAP-Like OGC Web Services, Data Models and Ecosystems. Estimates of costs and benefits must be seen on a relative scale in comparison to these pertaining to existing technologies. This relative approach also allows us to provide valid findings on a smaller sample set. Providing absolute numbers for effort would require a very large sample set and result in fuzzy intervals, as effort varies greatly with the

[1] https://github.com/DataCoveEU/API4INSPIRE

level of expertise. Providing effort relative to known tasks allows each reader to apply these values to their own experiences.

A final goal of this project pertains to the identification of gaps in information and supporting materials encountered in the course of the evaluation process. All identified gaps must be documented, required materials collected or prepared and made available. Based on these supporting materials, training webinars will be designed.

2.3. Evaluation Dimensions - Degrees of Freedom

For a well-founded evaluation, in addition to an in-depth understanding of the requirements, we must also know what we are evaluating against, i.e. what is the current baseline, as well as our possible axes of intervention, i.e. in which directions can we recommend modifications or further actions. The requirements we are evaluating against ensue from the previous sections. Our degrees of freedom are as follows:

- **API Specification.** While the specification for the SensorThings API Standard is fairly settled, with the V1.1 update forthcoming, the OGC API – Features remains a work-in-progress, with the core specification newly finalized and essential functionality such as filter/query still on the horizon. This challenging newness of the OGC API – Features can be seen as a benefit, as insights gained within this evaluation can be directly fed back into the OGC through the various members within the project team. The same holds true for the SensorThings API standard, although too late for the V1.1 update, having one of the OGC SWG chairs within this project can only be advantageous.
- **Data Model.** Data model simplification has been a recurring request within INSPIRE. As these simplification options have gone hand in hand with experimentation with (Geo)JSON-based approaches, it logically follows to integrate such thoughts within the API evaluation process. INSPIRE Themes suited to simplification to SF-0 and/or SF-1 will be selected, whereby initial work has commenced on the INSPIRE Theme Transport Networks - Air. Current candidates include Transport Networks - Roads and Natural Risk Zones - Flooding. Pertaining to SensorThings API, we will reflect on the additional attributes provided within the extended properties fields to determine if these are essential to our use cases and propose simplifications. Of relevance here is also enabling stakeholders more engagement pertaining to the data model. Some options include well-governed feedback processes or the creation of simplified models with standardized extension points.
- **Support Ecosystem.** Just as important for rapid uptake and use as the API Specification and the Data Model is the existence and completeness of the support ecosystem. A central access point to all required specifications, a vibrant community providing timely feedback, a wide spectrum of examples and code snippets can go a long way towards boosting productivity and user satisfaction. Pertaining to the two software systems integral to this exercise, GeoServer[2] [3] and

[2] https://github.com/geoserver/geoserver

FROST[4], there is already a vibrant community established providing rapid developer support as well as fixes and extensions to the software; the same is true for the underlying standards OGC API – Features[5] [6] and SensorThings API[7], with open GitHubs available for providing insights pertaining to necessary standard updates. This support ecosystem also includes transformation tools like HALE allowing data providers a simple way of transforming and providing their data or conversion tools such as LD-Proxy that can copy data from a service implementing traditional OWS to JSON based services.

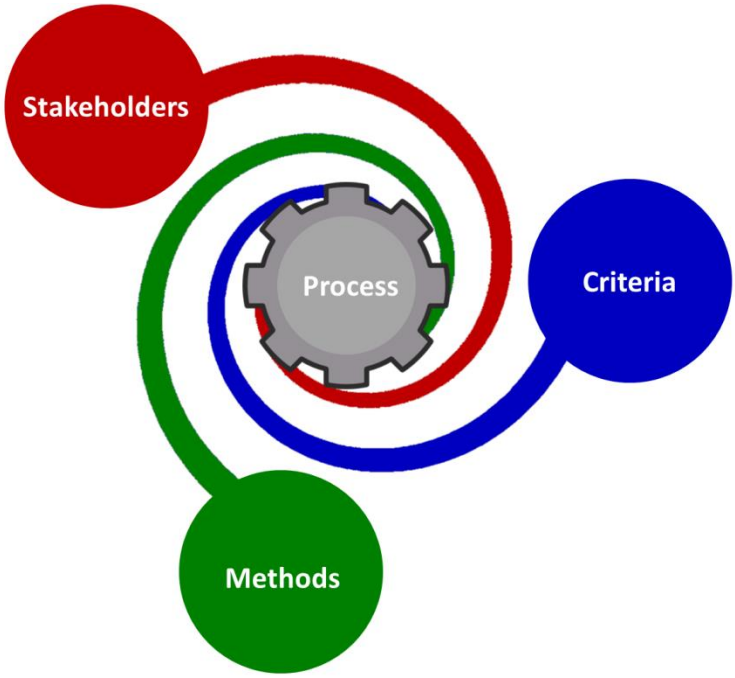
2.4. Evaluation Components

The following components compose our evaluation methodology:

- **Stakeholder Perspectives:** based on an analysis of stakeholders and their requirements towards provision and use of data, these have been defined in order to abstract the various roles taken by the stakeholders of this process. For more details on stakeholders, please see section “3 Stakeholder Analysis”.
- **Evaluation Criteria:** these formalize requirements towards APIs, providing a framework suited for gauging levels of maturity of an API specification and deployments as well as providing estimates of resource requirements for their achievement. For more details on the Evaluation Criteria, please see section “4 Evaluation Criteria - Five Level Open Data API evaluation model”
- **Evaluation Methods:** based on a review of state-of-the-art methods for the evaluation of APIs and other software systems, a set of different methods to be utilized for interrogating representatives of the Stakeholder Perspectives on the Evaluation Criteria was selected. For more details on these methods, please see section “5 Evaluation Methods”.

These evaluation components then feed into the Evaluation Process as described in the section below, whereby the Evaluation Methods applied to the Evaluation Criteria will be executed with representatives of the Stakeholder Perspectives both during the implementation and deployment of the services further described in “D2 Deployment strategies for standard-based APIs” as well as in the course of events more strongly addressing usage aspects.

[3] geoserver-users@lists.sourceforge.net
[4] https://github.com/FraunhoferIOSB/FROST-Server
[5] https://github.com/opengeospatial/ogcapi-features
[6] https://github.com/opengeospatial/oapi_common
[7] https://github.com/opengeospatial/sensorthings



2.5. Evaluation Process

Based on our project goals, the evaluation dimensions identified as well as the evaluation components defined, our evaluation process will commence. This process will be tightly linked with the development and deployment processes further described under our deliverable “D2 Deployment strategies for standard based APIs” in order to gain insights pertaining to the development and deployment processes. Once the APIs have been deployed under the tasks comprising “D3 Technical Documentation, Code”, the evaluation steps pertaining to the user perspective will proceed, including the various events described for this purpose. This will be further documented in “D4 Assessment of the impact of APIs”

Once the evaluation process has been completed, all outputs will be merged and analyzed, all gaps identified will be documented. All evaluation inputs will be reflected against the degrees of freedom identified in order to guide recommendations, quantify effort and define the contents of the supporting materials (technical guidance, software tools, tutorials, etc.) to be provided.

Based on the evaluation analysis, the outputs will be summarized and made available in various forms in order to meet the information requirements of the stakeholder groups involved.

- For administrative stakeholders, information on costs and benefits to be expected will be provided, allowing them to evaluate the potential of adoption of these technologies within their own organizations.
- For data providers, the necessary materials required to empower Member State authorities to deploy their data via the new APIs, as well as to understand the benefits and pitfalls of data simplification options will be provided.

- For data users, different levels of information will be required in order to address the different types of stakeholders.

A Webinar will be held in order to widely disseminate the outcomes of this project. All information resources and code collected during the project will be structured, documented, and made openly available. All relevant materials are being collected on the project GitHub.

3. Stakeholder Analysis

In this section, we provide an overview of the various stakeholder groups involved in providing and using data within the INSPIRE and beyond context. The requirements on the different stakeholder groups pertaining to both provision and use of data are analyzed, ensuing implications presented in section 3.1.1 Stakeholder Overview.

Based on this overview of stakeholder requirements, different aspects of data provision and use have been identified and described, providing deeper insights into the current status quo, and helping to highlight both challenges and opportunities put on stakeholders through the INSPIRE process. These are described in sections 3.1.2 Provision Spectrum and 3.1.3 Usage Spectrum.

For the purpose of this analysis, we have refined the various aspects pertaining to the different stakeholder groups as well as their roles into a set of stakeholder perspectives. Thus, we can differentiate between a data provider making data available in comparison to the same data provider accessing data, either from within their own organization or from external sources. These are described in section 3.1.4 Stakeholder Perspectives, and serve to guide the evaluation process.

While these Stakeholder Perspectives cover the operational aspects of data provision and usage, we must also consider the administrative aspects. Without organization buy-in, most data provision and usage endeavors are doomed to fail due to resource issues. Thus, we also address administrative stakeholders as those tasked with ensuring the necessary resources for the creation and maintenance of the entire data ecosystem.

3.1.1. Stakeholder Overview

In an initial step, in order to gain a better overview of the actual requirements, we reviewed the various stakeholders to be expected pertaining to provision and use of data within INSPIRE and beyond. A related aspect is understanding the approach to data access associated with the different stakeholder types.

It is important to be aware of the fact that the same stakeholder can assume multiple roles, causing the provision vs. usage model to blur. Simultaneously, exactly this blurring of roles is where we expect the greatest benefit, as those data providers also using the API data will provide the most in depth inputs as to the synergies to be leveraged. In Table 1: Stakeholder Overview we provide an overview of the different stakeholders involved together with information on their data provision and usage modalities. From this we derive implications based in the individual requirements to help guide the further evaluation process.

Table 1: Stakeholder Overview

	Provision	**Use**	**Implications**
Governmental Organizations providing data under INSPIRE	Provision with all requirements mandatory. Sometimes use of standard software, sometimes in-house developments for provision.	In-house use usually bypasses web services, directly accesses data sources. Often recurring processes already covered by in-house solutions.	Expect simple solutions for the continuation of their daily work, either integrated process for regular tasks, or standard formats that can be easily integrated into desktop GIS
Other Governmental Organizations (no INSPIRE requirements)	No provision requirements, but may also provide data regardless. Sometimes use of standard software, sometimes in-house developments for provision, whereby provided data models may be non-standardized, driven by backend DB or other requirements.	Access to data from other organizations (both within and without MS). Sometimes recurring processes already covered by in-house solutions; other times one-off studies.	Wish for simple solutions to enable their daily work. To date mostly depended on standard formats that can be easily integrated into desktop GIS.
Industry	No provision requirements, but may also provide data regardless. Sometimes use of standard software, sometimes in-house developments for provision, whereby provided data models may be non-standardized, driven by backend DB or other requirements.	Access to data from other organizations. Sometimes recurring processes already covered by in-house solutions; other times one-off studies.	

NGOs	No provision requirements, but may also provide data regardless. Sometimes use of standard software, sometimes in-house developments for provision, whereby provided data models may be non-standardized, driven by backend DB or other requirements.	Access to data from other organizations. Sometimes recurring processes already covered by in-house solutions; other times one-off studies.	Wish for simple solutions to enable their daily work. In-house processes for recurring analyses, some dependence on standard formats that can be easily integrated into desktop GIS, but also the realization that finding new answers implies creating new data tools
Scientific Organizations	No provision requirements, but may also provide data regardless. Sometimes use of standard software, sometimes in-house developments for provision, whereby provided data models may be non-standardized, driven by backend DB or other requirements.	Access to data from other organizations Usually new studies, thus expecting effort from data analysis & alignment	Some dependence on standard formats that can be easily integrated into desktop GIS, but strong realization that cutting edge research requires cutting edge tools. Limited funding for such tools, but willing students, staff and research personnel.

Citizen Science Initiatives	No provision requirements, but may also provide data regardless. Sometimes use of standard software, sometimes in-house developments for provision, whereby provided data models may be non-standardized, driven by backend DB or other requirements.	Access to data from other organizations Usually new studies, thus expecting effort from data analysis & alignment	Some dependence on standard formats that can be easily integrated into desktop GIS, but realization that research requires tools. Limited funding for such tools, but willing to accept some effort.
Interested Public	No data provision	Access to data from various organizations in a simple format, easy to view & understand, simple download formats	Simple (HTML) viewers with accompanying documentation would enable access to these resources. Simple download formats would allow non-professional users to interact with the primary data
EC Institutions	No provision requirements, but data centers sometimes try to align to INSPIRE data models and services.	Reporting data flows are increasingly being aligned with INSPIRE data models and services.	Modifications and extensions of the core data and service models must be supported.
Standardization Bodies	No provision requirements, but a vested interest in providing the relevant standards required for enablement.	No usage requirements, but a vested interest in providing the relevant standards required for enablement.	Standardization bodies must be engaged in order to ensure dynamic evolution as required by the other stakeholder groups.

3.1.2. Provision Spectrum

In order to provide a complete system of costs and benefits to the data ecosystems being analysed, we must first disaggregate the various aspects of data provision and usage. Pertaining to data provision, the following approaches will be investigated:

- **Configuration of an existing server for the provision of the desired API.** In the case of OGC API - Features, we have selected GeoServer[8] as the software of choice as this is widely deployed amongst data providers. In addition, developments towards provision of OGC API – Features are well progressed, providing us a relatively mature tool for the provision of this API. In the case of SensorThings API, we will utilize the FROST[9] implementation. In some of these deployments, we will provide the full scope of data as specified within the INSPIRE Data Specifications; in others, we will experiment with the simplification options proposed by the INSPIRE MIF activity 2017.2 on alternative encodings. This work will be done in close cooperation with our associated data providers, ensuring that all APIs deployed are in line with their requirements.
- **Development of a dedicated system for provision of OGC API – Features.** In some cases, data providers will prefer to create their own implementation of the OGC API – Features directly on their local data sources in lieu of deploying unfamiliar systems. In order to gain a better overview of the costs involved in such an undertaking, a green-fields development is being done on the Austro Control Air Transport Network data by a group of motivated students. This work will give us insights into the challenges faced by “outsiders” not intimately familiar with the OGC and INSPIRE domains. Once this development has been validated on the initial data source, it will be deployed on other data provider’s data sources.
- **API by proxy on existing WFS2.** In some cases, data is already provided using WFS2, but the server software used for this cannot be upgraded to also support OGC API - Features. In this case proxy software, such as LD-Proxy[10], may be a solution. This proxy software fetches data from the WFS2 server, and exposes the data through the OGC API - Features.

3.1.3. Usage Spectrum

Based on the stakeholder overview provided above, the following types of data usage become clear:

- **Direct access to data source (DB).** While this mostly pertains to in-house databases, there are also situations where entire databases are transferred between organizations on external discs. These solutions almost always require dedicated tooling tailored to the underlying databases. This can be a good solution for recurring tasks or when the dataset is too large for on-line transfer.

[8] http://geoserver.org/
[9] https://www.iosb.fraunhofer.de/servlet/is/82077/
[10] https://interactive-instruments.github.io/ldproxy/

Disadvantage is that external access to the dataset is not possible, and if both sides make changes to the data the two versions of the data set will go out of sync.

- **File-based data sources.** To date, many recurring and ad-hoc analyses have been performed based on data shared through semi-standardized file formats. An advantage of these formats is their simple accessibility through standard desktop GIS applications. Disadvantages are that such data cannot be interactively queried but can only be accessed as a monolithic block. In addition, the structure of the domain data contained is often not standardized.
  Examples of file-based data sources are Shapefiles[11], SpatiaLite[12] databases, GeoPackage, CSV and binary formats (NetCDF, image formats).
- **Access to SOAP-Like OGC Web Services (OWS) with simple features.** Various existing OWS services, be they non-harmonized INSPIRE download services or other services made available under the Open Government Data (OGD) umbrella, provide an interesting complement to the more structured harmonized INSPIRE Services. While these services rarely utilize complex data standards in the provision of available data, instead usually providing a fairly direct image of their backend data sources, they provide access to domains not covered by other directives in a simple and accessible manner. OWS providing simple features enables integration with standard desktop GIS applications.
  Disadvantage is that for each data source, one must first analyse the data structure and tailor tools to correctly interpret the data.
- **Access to SOAP-Like OGC Web Services (OWS) with complex features.** Various organizations provide access to data via OWS; in some cases the INSPIRE Data Models are utilized for data provision, while in other cases either community standards or proprietary models are used. Mature data models and service specifications allow for the transparent, accessible and reusable provision of rich data models.
  The complexity of the data models and service specifications can also be seen as a disadvantage as the full width and breadth of the available data is experienced as overload. Integration into classic desktop GIS becomes difficult, as recently illustrated by the CanIUse INSPIRE project[13]; an overview of which technologies can handle which encoding types has been provided there[14].
- **Access to REST-Based APIs.** REST-Based APIs are a relatively new development in web service spectrum, providing developers with easy access to web resources. Currently, we are undergoing a wild-west phase of API development and deployment, with various semi-standardized approaches emerging; a development strongly welcomed by the developer community. REST-Based, standardized APIs pertaining to both the spatial data as well as spatio-temporal observations have newly emerged, and are the object of evaluation in this project. Alternatively, data users can configure various on-the-fly transformation tools such as LD-Proxy or pygeoapi in order to enable API-Like access to existing WFS2 based resources.

[11] https://en.wikipedia.org/wiki/Shapefile
[12] https://en.wikipedia.org/wiki/SpatiaLite
[13] https://github.com/INSPIRE-MIF/caniuse
[14] https://inspire-mif.github.io/caniuse/generator/out.html

In addition to providing insights into current data usage modalities, these usage types also serve to illustrate the status quo pertaining to data provision. By classifying the data providers being interrogated during the evaluation process by their current data provision modalities, we can sharpen our insights as to how to best support and guide data providers at various levels of maturity in the most efficient usage of available resources towards data provision.

3.1.4. Stakeholder Perspectives

Based on this overview of relevant stakeholders and their requirements and approaches to data provision and use, we have derived the following **stakeholder perspectives** to guide us during the evaluation process:

1. **API Development** (Dev) [Provision]: Concerning the role of a software engineer, responsible to develop an implementation of the API Standard.
2. **API Deployment** (Deploy) [Provision]: Concerning the role of a data provider, responsible to publish through the API by utilizing and configuring existing implementations.
3. **API Use** (Use) [Usage]: Concerning the role of API consumers. These may be developers in the same organization owning the data, responsible for creating an application making use of in-house data. These may also be people outside the organization who are interested in the data, provided by the API. Access to the API is done either programmatically or via tools.

In addition to these three stakeholder perspectives, requirements pertaining to administrative stakeholders as described above will be accounted for.

4. Evaluation Criteria - Five Level Open Data API evaluation model

The technical evaluation criteria are based on the Five Level Open Data API evaluation model by Jarkko Moilanen[15], which is in turn derived from Sir Tim Berners-Lee's 5-star deployment scheme for Linked Open Data. As the original 5-Level model does not reflect the standardization aspects inherent to INSPIRE and the ELISE project, additional criteria have been integrated to reflect ensuing requirements; other criteria have been modified to better correspond with the standards evaluation context. The criteria have also been compared to those included within the ISO 25010 Standard on Systems and software engineering - Systems and software Quality Requirements and Evaluation (SQuaRE); the Suitability criteria was adopted from this source. These additions and modifications are noted in the individual criterion descriptions below.

As implied by the 5-Level methodology name, the criteria are grouped into five levels, which reflect increasing levels of maturity of both API and supporting ecosystem. Levels one through three pertain more to the maturity of the infrastructure provided by the data providers, with level one corresponding to data that has been provided with no regard to existing standards or conventions, level two pertaining to a standardized architecture with basic functionality and level three aligned with data provider requirements under INSPIRE as we know it. Levels four and five pertain more to the encompassing ecosystem, stemming from various communities.

Table 2 provides an overview of the Evaluation Criteria by level. More detail on the individual criteria is given in the following subsections.

We take these criteria as a foundation to evaluate the APIs through the stakeholder perspectives described in more detail above. The responses collected for these criteria will provide the basis for us to assess the relevance of these aspects for stakeholders assuming roles encompassed by the perspectives described above. The questions formulated for each criterion will also serve to evaluate how the effort accrued in its fulfilment compared to previous technologies, allowing us to determine potential savings through the usage of APIs.

15 https://www.europeandataportal.eu/sites/default/files/2013_api_simple_five_level_open_data_api_evaluation_model.pdf

Table 2: Overview of Evaluation Criteria

Level 1: All find	Level 2: All use	Level 3: All trust	Level 4: All involved	Level 5: All develop
Single Entry Point Is all information available from a single source (the portal), either directly or through links?	**JSON or XML** Does it support the use of JSON and/or XML?	**Query and Analytics API *** Does the API include Querying and Analytics?	**SDK Availability** Are API SDK's available for one or more environments?	**Code Visible** Is code visible/can be cloned?
Documentation Is updated documentation available?	**Data License** Are data license details given through the API?	**Error Handling** Is Error Handling in place and documented?	**Code Examples** Are there examples of code in one or more commonly used programming languages?	**Bug Tracker** Can Bugs, issues and suggestions be reported in a public place and is dialogue public?
Example Requests Are there examples of API requests seen as part of the documentation?	**Terms of Use** Are Terms of Use clear and easily accessible?	**Performance and Cache *** Can the API offer sufficient performance and does it support caching?	**Community** Is there a growing community to consult if needed?	**API License/reuse** Is the API's license known, are parts of the API covered by patent claims, and does it allow further development and re-use?
Example Data Are there examples of the API request returned data?	**Embedded Metadata** Does returning data include metadata?	**Background Support** Is the development and maintenance of the API supported by a big, stable entity or company?	**Playground** Is there an API Playground for testing and getting familiar with the API?	**Development Roadmap** Is the API's development roadmap known and is it visible for all?
Discoverability * Is it possible to discover deployed instances of the API based on the resources provided?	**Authentication** Does the API support authentication/authorization?	**Availability *** Is it easy to integrate into existing workflows and toolsets?	**Linked Documentation** Is documentation linked to code examples and back again?	**Linked Data Ready *** Is the data provided structured in a Linked Data ready manner, i.e. JSON-LD?
	API Standardization * Is the API itself standardised, and is this specification openly available	**API Data Validation *** Can the data returned by the API be validated?	**API Evolution *** Can a developer, data provider or user provide feedback on issues with the data model or API functionality and are custom extensions to the API foreseen?	**Test Framework available *** Can conformance to the API be formally tested?
	Suitability * Is the API suitable for the intended use?			

Note: those criteria extended to or modified from the original criteria list have been marked with an asterisk (*)

The effort required to meet each criterion is estimated on a scale of 1 to 5, with one being the least effort, and 5 being the highest. For some criteria, the effort can be shared with a larger community, or in some cases, community effort is required to meet the criterion. If this is the case, it is also indicated; in most cases, the effort will pertain to developers and deployers. Complementary to the effort involved in provision, we also quantify the benefit to users when the criteria are met. The effort and benefit estimations provided in this document stem from an initial heuristic quantification performed by the project development staff. These values will be successively refined with feedback received during the evaluation process. Effort and benefit estimates per criterion will be further refined by stakeholder role corresponding to the stakeholder perspectives described above, enriched through aspects described for the provision and usage spectra. In addition, the effort required to fulfill a criterion will be different from the effort accrued if the criterion is not fulfilled; both of these viewpoints will be integrated.

In the following sections, we present an overview of the five levels and the evaluation criteria therein. For details on how these will be posed as questions to each of our stakeholder perspectives, please see Annex A - Extended evaluation criteria.

4.1. Level 1: All find

This level corresponds to very basic non-standardized data provision, data is available together with simple examples. This level focuses on in-house use of APIs. Requires minimal effort in provision but is difficult to impossible to use.

Relevant questions: Is the API mainly designed for in-house use? Did the API happen by accident? Did the API evolve without relevant oversight or adherence to standards? Was the API designed based on a large set of realistic use-cases?

4.1.1. Single Entry Point

Is all information available from a single source (the portal), either directly or through links?

Finding all data one needs is a key requirement for any task. Ideally, there is a single place (the portal) through which all required information can be reached. For development this pertains to the API description, for deployment the data model definition, and for API use the data exposed through the API and the metadata required for understanding the data.

Effort required: 1.
Benefit provided: 4.

4.1.2. Documentation

Is updated documentation available?

Documentation is the main source to get information about the API. It should answer all the questions arising for all interested in the API, be it the software developer implementing the server,

the publisher mapping his data to the data model, or the user requesting data. The documentation should be clear and precise, so that no “trial and error” is needed.

Effort required: 2.
Benefit provided: 5.

4.1.3. Example Requests

Are there examples of API requests as part of the documentation?

The interaction point with an API are requests, created by a client-(application), sent to the API implementing server, processed and the result is sent back. The request contains all information, which describes the users information needs. Example requests help with understanding the documentation from all perspectives.

Effort required: 2; Community input possible.
Benefit provided: 5.

4.1.4. Example Data

Are there examples of the data returned by API requests?

After processing the received request, the server returns the requested data. It’s advantageous if the example data matches the example requests, and is described in detail.

Effort required: 2; Community input possible.
Benefit provided: 5.

4.1.5. Discoverability

Is it possible to discover deployed instances of the API based on the resources provided?

What means are available for discovering deployed instances of the API? Are dedicated services such as the OGC Catalogue Service Web (CSW) required for discovery? Can deployed instances of the API be discovered via normal search engines following W3C Data on the Web Best Practices (DWBP)[16] and Spatial Data on the Web Best Practices (SDWBP)[17]?

Note: this criterion was added to reflect the organizational and standardization background against which this evaluation is being performed.

Effort required: 2.
Benefit provided: 5.

[16] https://www.w3.org/TR/dwbp/
[17] https://www.w3.org/TR/sdw-bp/

4.2. Level 2: All use

This level corresponds to basic standardized data provision. Data is available in a standardized format, basic standards based access functionality is provided. This level focuses on providing data to external users. Medium effort in provision, use is possible, but not optimal.

Relevant questions: Has the API been reviewed by external users? Has the API been designed for providing data to external users?

4.2.1. JSON or XML

Does the API support the use of JSON and/or XML?

Data needs to be represented in a serialized form to be transmitted. For web-based APIs, JSON and XML are established as de-facto-standard, since they're human readable and many implementations exists for various programming languages, which allows easy integration and reuse. These aspects are relevant for server development and API use, but not for the deployment.

Effort required: 1.
Benefit provided: 4.

4.2.2. Data License

Are data license details given through the API?

The main value of an API is provided through the data that is published through the API. Therefore, it's important that the license of the data is also available through the API itself. The data license is usually a legal document, and in most cases this document will be linked to from within certain API responses. Ideally, the data license is based on a standard licensing scheme such as CC BY.

Effort required: 1.
Benefit provided: 3.

4.2.3. Terms of Use

Are Terms of Use clear and easily accessible?

In addition to data license, the terms of use should be available. A data provider should be able to specify how and with which constraints an API service can be used, and it should be clear to users of the API where to find this information. The terms of use are usually described in a legal document, and in most cases this document will be linked to from within certain API responses.

Effort required: 1.
Benefit provided: 3.

4.2.4. Embedded Metadata

Does returning data include metadata?

Getting data is often not sufficient. Additional metadata is needed to use and interpret the data correctly. E.g. what is represented by the data? What are the units? The API must allow the developer to enable the provider to provide the metadata required by the user to interpret the data.

Effort required: 3 – 4.
Benefit provided: 4.

4.2.5. Authentication

Does the API support authentication/authorization?

Not all data should be available publically. To limit access to authorized users, authentication/authorization (auth*) mechanisms exist. To ease integration, the auth* methods should be based on existing, well-known protocols (e.g. Oauth[18] or OpenID-Connect[19]). After authenticating, the authorization mechanisms define exactly which data the user is allowed to see.

Effort required: 2 – 4.
Benefit provided: 3.

4.2.6. API Standardization

Is the API itself standardised, and is this specification openly available?

Note: this criterion was added to reflect the organizational and standardization background against which this evaluation is being performed.

For an API to become widespread, it is helpful if the API is formally adopted as a standard by a well-respected, international standards body. Furthermore, an API standard rarely stands alone. Most standards refer to other standards for specific aspect. For example, in most standards, whenever a date or time is used, the encoding is done according the ISO 8601 standard. Referencing existing standards usually reduces the effort required for implementing the standard and makes it easier for clients to use the standard, since they can use common libraries that implement these referenced standards. It also reduces the effort of mapping data models, since standardised building blocks are likely to already be in use.

Effort required: 5; Community input required.
Benefit provided: 5.

[18] https://en.wikipedia.org/wiki/OAuth
[19] https://en.wikipedia.org/wiki/OpenID_Connect

4.2.7. Suitability

Is the API suitable for the intended use?

Note: this criterion was added based on the criteria available from within the ISO 25101 Standard.

For an API to be deployed and used, it has to be suitable for the intended use case. From the deployment perspective, this means the API has to be implemented in server software that fits in the deployment landscape of the data provider and that can be connected to, or loaded with, the data that the data provider intends to publish. From the user perspective, this means that the user must be able to request, in a suitably efficient manner, the data that he needs.

Effort required: 3.
Benefit provided: 4.

4.3. Level 3: All trust

This level corresponds to mature standardized data provision. Encompassing data is available in complex standardized format, powerful and mature standards based access functionality is provided. Fairly large effort in provision, use is well supported.

Relevant questions: Has the API been designed for use in a diverse set of use cases, in a diverse set of environments? Can the API support the complexity of data models required for real-world use cases? Has the API been designed for use with large data sets?

4.3.1. Query and Analytics API

Does the API include Querying and Analytics?

Note: this criterion was modified from the original to include query functionality together with analytics

APIs are often used to get data from a service. Usually only a subset of the available information is of interest. Therefore, the API should contain querying and analytic capabilities. First, this includes a suitably powerful filter mechanism to limit the response to those parts of the data that the user is interested in. Second, this includes a mechanism in the API to do basic analytical calculations, e.g. some aggregation functions.

Effort required: 4.
Benefit provided: 5.

4.3.2. Error Handling

Is Error Handling in place and documented?

While using an API errors might occur. Either there's an issue with the server itself, or the data sent by the user isn't correct or doesn't match the available data. To allow a user to handle these errors, the API should specify how errors are reported back to the user. At the same time, the error message should not expose sensitive internal information about the server deployment.

Effort required: 2.
Benefit provided: 4.

4.3.3. Performance and Cache

Can the API offer sufficient performance and does it support caching?

Note: this criterion was modified from the original to include performance functionality together with caching

APIs may have inherent performance bottlenecks that become apparent when deploying and using the API with large data sets or a large number of users. To increase performance and efficiency, caching mechanisms may be used.

Effort required: 2 – 4.
Benefit provided: 3.

4.3.4. Background Support

Is the development and maintenance of the API supported by a big, stable entity or company?

Deciding to use a specific API requires investments on all sides (dev, deploy, use). Thus, for the future development of the API itself and the implementation is important, to be sure that the API will still be relevant in the future. A big entity or company in the background increases this probability.

Effort required: 1 – 5; Community input possible.
Benefit provided: 4.

4.3.5. Availability

Is it easy to integrate into existing workflows and toolsets?

Many users and domain experts have well established workflows that use existing tools and (desktop) software packages. Switching to a different data source or API may break these workflows, which would greatly reduce the incentive for users to switch to the new API. These tools can include

those used by providers to import data from external or primary sources into the local data infrastructure, or those used by data users to visualise or otherwise use the data.

Note: this criterion was modified from the original to include availability of relevant tooling.

Effort required: 4; Community input required.
Benefit provided: 4.

4.3.6. API Data Validation

Can the data returned by the API be validated?

For interoperability it is important that the data returned by an API conforms to the data model defined by the API. It is beneficial if there is a way to automatically check this, for example by checking the JSON against a JSON Schema, or XML against an XML Schema Definition.

Note: this criterion was added to reflect the organizational and standardization background against which this evaluation is being performed.

Effort required: 3; Community input possible.
Benefit provided: 4

4.4. Level 4: All involved

This level pertains more to the maturity of the wider ecosystem built up around the standardized specifications. The aspects described go beyond the control of the individual data providers and thus must be supported through diverse stakeholder communities. Achieving this level facilitates both the provision and usage perspectives.

Relevant questions: Can support for the API be seen as a community effort?

4.4.1. SDK Availability

Are API SDK's available for one or more environments?

Software Development Kits (SDKs) simplify the interaction with the API on the development and client side. They contain libraries that reduce the amount of code that needs to be created to interact with the API and help reuse existing work and integrating the API in new contexts.

Effort required: 3; Community input possible.
Benefit provided: 2.

4.4.2. Code Examples

Are there examples of code in one or more commonly used programming languages?

Like SDKs, code examples simplify and clarify the interaction with an API. They show in detail how certain aspects of the API should be used and can often directly be executed to see the covered features live in action.

Effort required: 3; Community input possible.
Benefit provided: 4.

4.4.3. Community

Is there a growing community to consult if needed?

A big, active and friendly community can be, in addition to the documentation, an additional source of information. Questions, best-practices and issues can be discussed within a community to spread available knowledge and provide support.

Effort required: –; Community input required.
Benefit provided: 4.

4.4.4. Playground

Is there an API Playground for testing and getting familiar with the API?

Learning by doing and practically testing the usage of the API helps getting more insights. A playground helps with this. Such a playground can range from a public server that anyone can access, to a docker image that can be quickly deployed with standard settings, to a one-click-install installation package that can run on a desktop PC.

Effort required: 2; Community input possible.
Benefit provided: 3.

4.4.5. Linked Documentation

Is documentation linked to code examples and back again?

A documentation that links to code examples, example data and request (and back) helps the reader to better understand the API.

Effort required: 2; Community input possible.
Benefit provided: 3.

4.4.6. API Evolution

Can a developer, data provider or user provide feedback on issues with the data model or API functionality and are custom extensions to the API foreseen?

Note: this criterion was added to reflect the organizational and standardization background against which this evaluation is being performed.

APIs are rarely perfect and finished in their first incarnation. They also always have to strike a balance between the diverse requirements of many different use-cases on the one hand, and complexity on the other. Because of this, it is important that developers, providers and users have a way to provide feedback on issues, deficiencies and unclarities in the API. It is important that these issues are addressed in future versions of the API. Similarly, it is very helpful if the API has clear extension points so that the API can be extended for certain use cases that require functionality that is not in the API.

Effort required: 2; Community input required.
Benefit provided: 2.

4.5. Level 5: All develop

This level describes a well developed and mature ecosystem built up around the standardized specifications. The aspects described go far beyond the control of the individual data providers and thus must be supported through diverse stakeholder communities. Achieving this level facilitates both the provision and usage perspectives.

Relevant questions: Can (further) development of the API be seen as a community effort?

4.5.1. Code Visible

Is code visible/can be cloned?

Having access to the source code of a reference implementation allows a deeper understanding of the implementation. It offers the possibility to check if specific behaviour was intended or is a bug. Available open-source code with a suitable license offers the possibility to add own changes, so that there is no dependency on a third-party.

Effort required: 5; Community input required.
Benefit provided: 3.

4.5.2. Bug Tracker

Can Bugs, issues and suggestions be reported in a public place and is this dialogue public?

Though an API is not software and can thus not have “bugs” in the traditional sense, an API can still have inconsistencies, errors and unclear definitions. Often these issues are not noticed until the API is applied in specific use cases, or implemented by multiple people. Having a Bug Tracker publically available offers a place, where to report issues and to track discussions and solutions. Having such a system openly available allows input from a wider community, helping the system to evolve to support a wider user community. It also makes it easier to collaborate on extensions.

Effort required: 1; Community input possible.
Benefit provided: 4.

4.5.3. API License/reuse

Is the API's license known, are parts of the API covered by patent claims, and does it allow further development and re-use?

Parts of APIs can be covered by patent claims, making it impossible to implement the API without paying royalties. The license of the API is important and might be a blocker, if the API needs to be re-used or if further development of the API is required. Ideally, if a license is required this should be based on a standard licensing scheme such as CC BY.

Effort required: 2; Community input possible.
Benefit provided: 3.

4.5.4. Development Roadmap

Is the API's development roadmap known and is it visible for all?

A roadmap can help to understand the further development direction of the API and to know what to expect in the (near) future.

Effort required: 3; Community input possible.
Benefit provided: 2.

4.5.5. Linked Data Ready

Is the data provided structured in a Linked Data ready manner, i.e. JSON-LD?

Note: this criterion was added to reflect emerging technological advances to be expected within the stakeholder community.

Linked Data is a technology that looks very promising and that has been on the horizon for some time now, with parts and concepts of it finding their way into APIs and data models. While it is not yet practical to have an API that fully employs all Linked Data principles, it is possible to design the API and data models in a way to allows Linked Data adoption in the future.

Effort required: 3; Community input required.
Benefit provided: 3.

4.5.6. Test Framework available

Can conformance to the API be formally tested?

Note: this criterion was added to reflect the organizational and standardization background against which this evaluation is being performed.

Test Frameworks can be used to verify the implementation and deployment of the API. This helps interoperability by insuring the deployed services implement the API correctly.

Effort required: 4; Community input required.
Benefit provided: 3.

4.6. SQuaRE: ISO 25010 Model

Our main evaluation criteria is the “Five Level Open Data API evaluation model” described above. To validate this model, additional evaluation catalogs were investigated. In the context of software product quality the “ISO 25010 Standard on Systems and software engineering - Systems and software Quality Requirements and Evaluation (SQuaRE)” is well established. It defines properties for a software product (without naming a concrete implementation) which should be considered to have a high-quality product. Although those properties were developed for evaluating a software product, they can be used for evaluating an API, too. To prevent having an additional evaluation model in this project, the ISO standard is used to validate our already defined criteria and to check the completeness of the “Five Level model”.

As a result our evaluation criteria were mapped to the categories of the ISO standard (see Table 5 in Annex B - SQuaRE: ISO 25010 Model). In addition we double-checked that all categories of the ISO standard are mapped to at least one criteria of the Five Level model. The categories *Context coverage* and *Compatibility (technical)* of the ISO standard were considered irrelevant. *Suitability* was added to Level 2, as an additional criterion.

5. Evaluation Methods

For the evaluation of the degree of achievement of the evaluation criteria defined pertaining to each of the stakeholder perspectives, a set of evaluation types have been defined. These range from Heuristic Expert Evaluation over Peer Review of critical functionality to Interviews and Questionnaires performed in the framework of various events. Details on the different evaluation types as well as on events being organized in order to allow us to engage with the various stakeholder groups is provided in the following subsection 5.1 Evaluation Types.

The different evaluation types have been matched with both the stakeholder perspectives as well as the evaluation criteria, with the most suitable evaluation types assigned to each criterion paired with a stakeholder perspective. Detailed information on how the evaluation types will be applied towards the different stakeholder perspectives can be found in subsections 5.2 API Development and Deployment 5.3 API Use. Explicit alignment of evaluation types with the evaluation criteria and stakeholder perspectives can be found in Table 3: Evaluation types for the stakeholder perspectives develop and deploy and Table 4: Evaluation types for the stakeholder perspective use. Based on these alignment tables, the explicit content of the interview and questionnaire questions will be derived, whereby the questions defined for each evaluation criterion are described in Annex A - Extended evaluation criteria.

A great deal of insight pertaining to the development and deployment processes will be gleaned from our data providers. The data sources and provision technologies have been selected to not only provide insights into direct development and deployment requirements, but also to allow for comparison between competing technological solutions. For selected data providers, the same data source will be exposed through alternative technologies, allowing for direct comparisons pertaining to both provision and usage efforts. Based on these interviews, we will gain insights as to the effort required in reaching the successively more complex levels.

Events are being organized in order to allow us to directly engage with the wider user community. The various data sources provided by the development and deployment stakeholders linked to this project will be made available; information on the provision process will be provided together with various usage examples. The participants will then be given the opportunity to interact and experiment directly with the APIs and other services provided to gain a better understanding of the potential of these technologies as related to their daily work. During these events questionnaires will be provided to collect feedback from participants, individuals will be selected for more in depth interviews. More information on the events being organized is available from the following subsection Events within subsection 5.1.3 Stakeholder Evaluation.

5.1. Evaluation Types

5.1.1. Heuristic Expert Evaluation

The Heuristic Expert Evaluation is the least structured of the planned evaluation types. Under this methodology, we rely on the experience of the domain experts involved within the API4INSPIRE

Project, both the project team as well as data provider staff will be included. While some of this input will be collected during dedicated meetings, we expect many of these insights to be gained on-the-fly while performing the tasks required to develop, deploy and use the two APIs being evaluated. Such interactively identified insights will be collected as issues within the project GitHub at https://github.com/DataCoveEU/API4INSPIRE. These will be merged with the insights gained from the more formalized evaluation meetings, and will provide a well-founded basis for our subsequent analysis.

The Heuristic Expert Evaluation will be applied to all interaction types described above. For some criteria, this evaluation type will suffice to provide well-founded input. For other criteria, this evaluation step will help to highlight issues that must be followed up on by the other methods listed.

5.1.2. Peer Review

The API Usability Peer Review process proposed by Farooq will be a valuable tool for gaining focused insights on specific aspects of the API usability. While this methodology has been designed for the evaluation of the features of the emerging APIs, we believe that it can be effectively applied within the context of evaluating existing API Specifications. The original process foresaw the following roles:

1. Feature Owner: responsible for the specification and development of a specific functionality
2. Feature Area Manager: responsible for the wider functionality area, can put the individual functionality into context
3. Usability Engineer: responsible for evaluating usability, coordinates process
4. Reviewers: organizational peers who will provide feedback.

In the original process, once the Usability Engineer has organized a review, the Feature Owner presents the new functionality in order to allow rapid uptake by the Reviewers. The Reviewers provide feedback, the Usability Engineer collects and structures this feedback, the Feature Area Manager assures that all feedback and ensuing activities remain aligned with the requirements of the wider feature area.

As the API Usability Peer Review process was custom tailored to the evaluation of an API under development, we must first adapt this process to our current evaluation target pertaining to the deployment and usage of a standardized API. As we are not evaluating the explicit implementations but instead only evaluating the functionality according to the standard, we believe that we can merge the roles of Feature Owner and Feature Area Manager without negatively impacting the robustness of the methodology, whereby we will refer to this role as the Feature Manager. Thus, within the API Peer Review, the following roles will be required:

1. Feature Manager: responsible for the description of a specific functionality, as well as ensuring alignment with the wider context.
2. Usability Engineer: responsible for evaluating usability, coordinates process
3. Reviewers: organizational peers who will provide feedback.

Before the review is started, the Feature Manager and Usability Engineer will prepare basic documentation on the feature to be evaluated, if relevant examples. In addition, questions to be posed will be documented.

Once the review commences, the Feature Manager will have 20 minutes to present the functionality, explain how it is to be utilized, known strengths and weaknesses. Once the functionality has been presented, the prepared questions are posed to the reviewers. The Usability Engineer will collect all insights from the ensuing dialog between the Feature Manager and Reviewers. Towards the end of this block, there should also be a question foreseen allowing reviewers to provide input on strengths and weaknesses identified but not covered by the other questions (AOB, but for questions, AOQ). This question and answer session should last about 60 minutes. At the end, the Usability Engineer has 10 minutes to present the insights taken and finalized these with the reviewers.

After the review has been performed, the Feature Manager and Usability Engineer come together for a post mortem meeting, discuss the outcomes of the review in detail and formalize the output.

5.1.3. Stakeholder Evaluation

While we have a well-balanced team of developers and data providers both within our internal project team as well as our associated data providers, we must go beyond this select group in order to gain deeper insights into the requirements of the wider stakeholder community.

In contrast to our internal and associated teams the stakeholder community will mostly not be familiar with the APIs being presented nor the data model variants utilized. Before we proceed to interrogate this community via questionnaires or interviews, we must first provide them with all relevant information required to gain the necessary insights into the proposed technologies and data models. Thus this evaluation type will be undertaken within the framework of various events, during which the necessary information will be disseminated. The questionnaires will be made available to all participants, while we will select individuals for more in depth questioning in the form of an interview.

Events

Foremost pertaining to the usage aspects of the APIs under evaluation, various events will be organized to allow us to directly engage with potential stakeholders, disseminate information on the current status of these APIs and collect feedback based on their experiences. The participants in these events will form the pool of experts who will be further interrogated via questionnaires and interviews as described in the sections below.

One challenge faced in the organization of events is posed by the tight project duration. Many relevant events initially planned such as the regular hackathons organized by the German cities of Munich or Hamburg do not temporally convene with our requirements. Participation in events cannot be guaranteed as we must await the outcome of the submission process where we have applied.

Events to be conducted in the scope of this study are planned to include, whereby the later points are fallbacks for the case that the first two points prove not viable:

- INSPIRE/api APIthon at the INSPIRE Conference in Dubrovnik (12-15 May: WS submitted, further organizational aspects discussed with JRC) In addition to API usability aspects, in Dubrovnik we will provide information on our development and deployment activities, as well as allowing participants to provide their own existing WFS2 solutions via LD-Proxy
- FOSS4G Europe (13-17. July) API Hackathon will be submitted as a WS
- An alternative for gaining feedback on API usage would be a dedicated in-house event at Fraunhofer IOSB
- A similar alternative would be the organization of a Webinar. The organizers present the available APIs together with existing code examples, the participants are given the opportunity to experiment and ask questions
- While formal EGU (3-8 May) participation is no longer possible, we are considering leveraging the high concentration of relevant experts in Vienna the week before Dubrovnik through the organization of a side event or round table.

Questionnaire

Questionnaires will be created in order to gain insight into various evaluation criteria, whereby the questions to be posed will be constrained to those where we expect answers that fill well to the questionnaire format. For a full list of questions pertaining to the evaluation criteria, please see the Annex A - Extended evaluation criteria. Most questionnaire questions will be limited to the following response types to allow for wide scale analysis of the outcomes:

- Binary yes/no responses
- Multiple choice responses (both single response and multiple response versions)
- Word Clouds

We propose the usage of an online survey tool for the questionnaires, whereby the project team can provide such functionality via Mentimeter[20]. At present we foresee one questionnaire that also covers provision aspects; should such an approach lead to unnecessary questions (i.e. it is not possible to branch the questionnaire to only pose usage questions if the respondent does not provide data) separate questionnaires will be created for deployment and use.

Interview

While many of the evaluation criteria can be reduced to simple questions suited for a questionnaire as described above, other criteria will require more complex feedback. For this purpose, participants will be selected from our events based on their level of engagement and the more complex evaluation criteria discussed with them in the form of a structured interview, with the interview questions being derived from the questions pertaining to development, deployment and usage provided together with the full list of criteria in Annex A - Extended evaluation criteria.

[20] https://www.mentimeter.com/

5.2. API Development and Deployment

As we only have three API development teams, the evaluation types will be limited to expert heuristic evaluation and interview as described above. Creation of a questionnaire for such a small stakeholder group would not be efficient, but could be done if a similar evaluation is performed in a wider context. Should the need arise we may introduce the Peer Review evaluation type, but at present we believe that the evaluation types posed suffice. The evaluation aims to provide answers to the detailed criteria questions pertaining to development provided in the Annex to this document.

Pertaining to API deployment, the evaluation types expert heuristic evaluation and interview will be used similarly to the API development evaluation. In addition, questionnaires will be utilized in order to address a wider audience of interested data providers that are taking notice of our activities, be it through our events or through word-of-mouth. The evaluation aims to provide answers to the detailed criteria questions pertaining to deployment provided in the Annex to this document. The interview questions will take administrative and infrastructural aspects into account, illustrating how existing investments in infrastructure and applications can be leveraged. Table 3 shows which evaluation types are to be applied to which evaluation criteria:

Table 3: Evaluation types for the stakeholder perspectives develop and deploy

	Stakeholder Perspective:	**API Development**		**API Deployment**		
	Criteria:	**Heuristic Expert Evaluation**	**Interview**	**Heuristic Expert Evaluation**	**Questionnaire**	**Interview**
Level 1: All find						
	Single Entry Point	X		X		
	Documentation	X	X	X	X	X
	Example Requests	X	X	X	X	X
	Example Data	X	X	X	X	X
	Discoverability	X	X	X	X	
Level 2: All use						
	JSON or XML	X		NA	NA	NA
	Data License	X	-	X	X	
	Terms of Use	X	-	X	X	
	Embedded Metadata	X		X		X
	Authentication	X	X	X	X	X
*	**API Standardization**	X		X		X
*	**Suitability**			X	X	X
Level 3: All trust						
*	**Query and Analytics API**	X	X	X		
	Error Handling	X	X	X		-
*	**Performance and Cache**	X	X	X		X

	Background Support	X	-	X	X	
*	**Availability**	NA	NA	X	X	X
*	**API Data Validation**	X	X	X	X	
Level 4: All involved						
	SDK Availability	X	X	NA	NA	NA
	Code Examples	X	X	NA	NA	NA
	Community	X		X	X	
	Playground	X		X	X	
	Linked Documentation	X		X	X	
*	**API Evolution**	X		X		X
Level 5: All develop						
	Code Visible	X		X	X	X
	Bug Tracker	X		X	X	-
	API License/reuse	X		X	X	-
	Development Roadmap	X		X	X	X
*	**Linked Data Ready**	X		X		
*	**Test Framework available**	X	X	X	-	-

5.3. API Use

As API usability is the strongest argument for their uptake, we will investigate this aspect with a wide range of evaluation types. In order to gain deep insights into suspected weaknesses of the APIs, in addition to Heuristic Expert Evaluation we will apply the Peer Review process described above with participants from among our data providers, development teams and associated experts.

Various events as described above are planned that will allow us to present the APIs to a wider audience and collect feedback after they have had the opportunity to interact with the APIs and gain their own experience. Questionnaires will be used to address the wider audience, while interviews will be conducted with select event participants.

In Table 4 we show which evaluation types are to be applied to which evaluation criteria:

Table 4: Evaluation types for the stakeholder perspective use

Stakeholder Perspective:	**API Use**			
Criteria:	**Heuristic Evaluation**	**Peer Review**	**Questionnaire**	**Interview**
Level 1: All find				
Single Entry Point	X			
Documentation	X		X	X
Example Requests	X		X	X
Example Data	X		X	X
Discoverability	X	X	X	
Level 2: All use				
JSON or XML	X			
Data License	X	X		

	Terms of Use	X			
	Embedded Metadata	X	X	X	
	Authentication	X	X	X	
*	**API Standardization**	X		X	
*	**Suitability**		X	X	X
Level 3: All trust					
*	**Query and Analytics API**	X	X	X	X
	Error Handling	X	X	X	
*	**Performance and Cache**	X	X	X	
	Background Support	X		X	
*	**Availability**	X		X	
*	**API Data Validation**	X		X	
Level 4: All involved					
	SDK Availability	X		X	
	Code Examples	X		X	
	Community	X		X	X
	Playground	X		X	
	Linked Documentation	X		X	
*	**API Evolution**	X		X	
Level 5: All develop					
	Code Visible	X			X
	Bug Tracker	X		X	X
	API License/reuse	X			X
	Development Roadmap	X			X
*	**Linked Data Ready**	NA	NA	NA	NA
*	**Test Framework available**	X			

6. Evaluation Process

In this section, we describe the individual steps of our evaluation process in detail. We begin with a description of how the evaluation methodology initially foreseen for this task has been further reflected against the project goals, the methodologies refined as required. We go on to sketch the consultation process, describing the different evaluation types being applied to the different stakeholder perspectives in order to gain insights pertaining to our evaluation goals. Once the consultation has been completed, the analysis of the outputs will commence, providing insights as to strengths and weaknesses of the new technologies under evaluation as well as potential measures. In a final section, we describe how the insights gained will be used to generate the necessary materials required to enable MS stakeholders to assess the implications of adopting these technologies as well as deploy and utilize them within their own organizations.

Figure 2 provides an overview of the Evaluation Process. The consultation step has been split by stakeholder perspective, as these different perspectives require quite different approaches to the evaluation process. All evaluation steps are described in greater detail in the sections below.

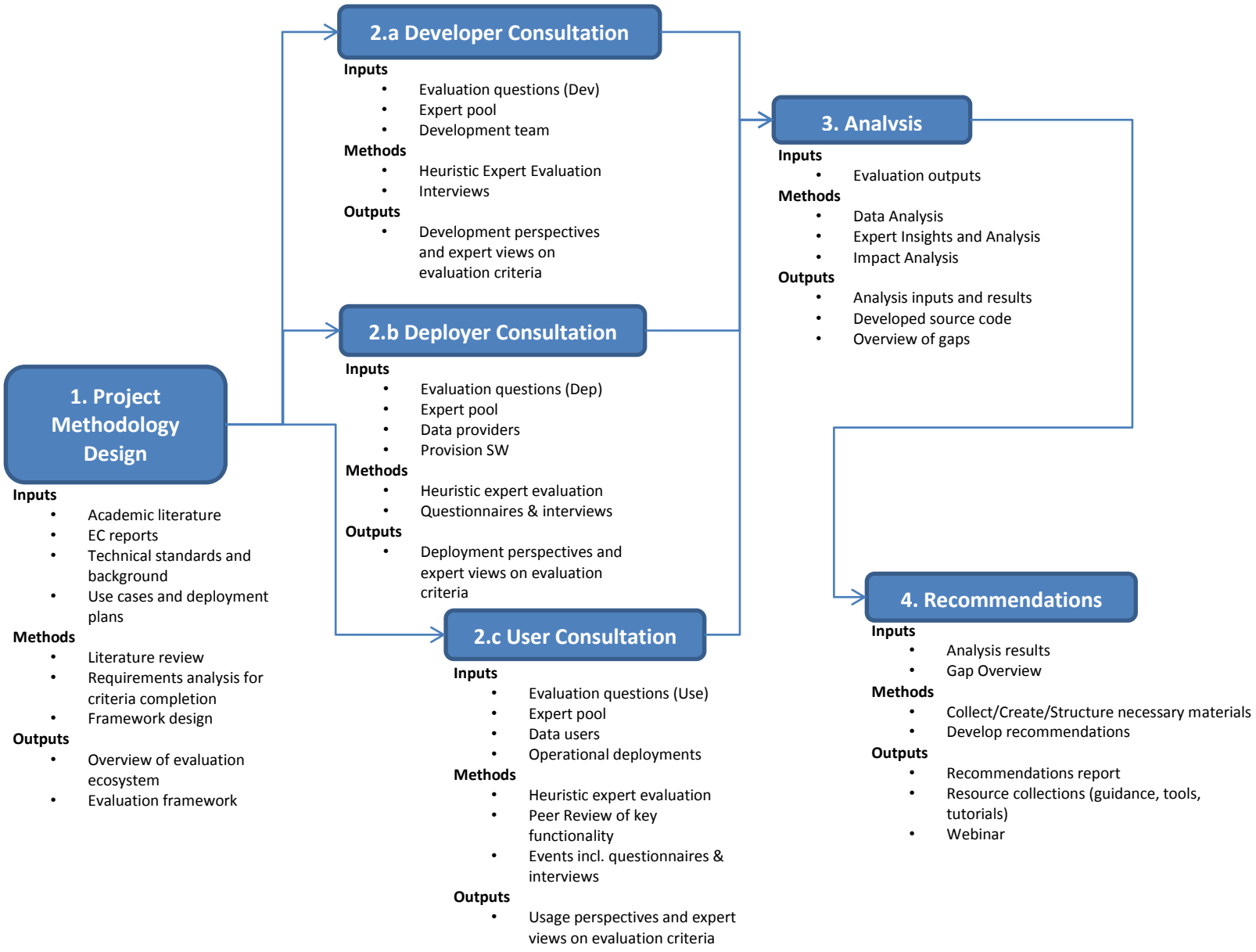


Figure 2: Overview of the evaluation process. The consultation step has been split by stakeholder perspective, as these different perspectives require quite different approaches to the evaluation process. All evaluation steps are described in greater detail in the sections below.

6.1. Project Methodology Design

As described in the sections above, the initial evaluation process foreseen was refined based on relevant literature and the experience of the project team. In order to gain deeper insights into the stakeholder community and their requirements, we first put together a stakeholder overview, whereby this information was based both on our long experience within the INSPIRE and wider governmental data sector complemented with insights from relevant reports. This list was extended through information on typical provision and usage patterns pertaining to the stakeholder group together with implications stemming from these requirements. Based on this list, we then derived the core stakeholder perspectives that serve to guide this evaluation.

Based on the stakeholder overview with provision and usage requirements, we then further refined both the provision and usage spectra, detailing the diverse provision and usage options available. In addition to deepening our understanding of stakeholder requirements, this analysis also provides a good overview of the existing landscape, allowing us to evaluate the impact of the new APIs and data models against known quantities and efforts.

In order to further guide our evaluation process, we also investigated and documented the degrees of freedom available for remediation of issues identified. This will help guide us in identifying potential solution space while giving us the wisdom to accept the things we cannot change. In line with our agile approach, we then revisited our goals and refined them in light of these insights.

The Five Level Open Data API evaluation model by Jarkko Moilanen was then refined based on the requirements identified. Further criteria were added, both from other evaluation schemes such as described in ISO 25010 as well as based on our understanding of the wider evaluation domain, whereby the main extension focus was to cover the standardization aspects missing in Moilanen's work. In some cases existing criteria were modified to better reflect our requirements. In addition, we defined more generalized criteria to indicate general achievement of a specific level.

In parallel to the work on the evaluation criteria, we also refined the evaluation types foreseen in the project concept. Some evaluation types were modified to better reflect the evaluation goals specific to this project, as most methodologies available are tailored for the evaluation of a concrete software implementation. Paired with this work was determining which evaluation type is most suited for gaining insights pertaining to a specific criterion applied to the perspective of a specific stakeholder perspective. This alignment is provided together with the description of the different evaluation types. In addition, events have been foreseen in order to attract a wider user community, and gain insights from their interactions with the APIs being made available by this project.

Once the evaluation criteria were aligned with the requirements of this project, we then derived questions pertaining to each level and individual criterion, tailored to reflect the insights of the stakeholder perspective being addressed. These questions will be further refined in accordance with the evaluation types deemed suitable for the evaluation of the specific criterion being addressed as different evaluation types allow for different types of interactions, and thus different complexities in both the

questions posed and the types of answers expected. All these additions and modifications to the evaluation criteria have been documented and provided within this document; full information on potential questions has been shifted to Annex A - Extended evaluation criteria for easier reference.

6.2. Consultations

While the consultations pertaining to API use will be grouped around our proposed events, the consultations with the developers and the data provider staff involved in deploying the new APIs will take place in an ongoing manner during the deployment process. More details on the deployment process will be provided in D2 Deployment strategies for standard based APIs. In addition to the more fine-grained criteria stemming from the 5-level evaluation model, the consultations will also include effort required in achieving these individual levels.

All gaps pertaining to documentation and other essential resources identified during the development, deployment and use processes will be collected on the project GitHub as issues marked with the label "documentation missing", providing a dynamically up to date overview of all such gaps encountered. All required materials will be collected and collated on the GitHub Wiki, whereby the contents of this wiki can later be transferred to a target site of the customer’s choice.

6.2.1. Developer Consultation

Pertaining to developer consultation, we have an interesting resource pool to interrogate. On the one side, we have the developers of two of the leading API provision systems GeoServer and FROST within our project team. These colleagues from Fraunhofer IOSB and GeoSolutions will provide well-founded information on their experiences in implementing professional solutions for the provision of these APIs.

In order to complement these insights with fresh perspectives, we have also triggered the development of a simplified green-fields implementation of the OGC API – Features tailored to the provision of simple features corresponding to the SF-0 feature standard. This implementation is being done by a team of two highly motivated students currently undergoing vocational training in Information Technology at the Höhere Technische Lehranstalt Spengergasse, a school for higher technical education located in Vienna. The initial implementation is being done on datasets provided by Austro Control, both as SQLite and PostGIS data sources.

As the number of involved developers is naturally limited, the evaluation types applied will be far more interactive, consisting of ongoing Heuristic Expert Evaluation paired with more structured interviews aligned with the evaluation criteria. In addition, the developers will be requested to provide insights into the efforts required for implementation, both in absolute terms as well as in relation to known efforts pertaining to existing technologies.

6.2.2. Deployer Consultation

The six institutions integrated into this project as data providers will provide us with insights pertaining to the process of deploying existing solutions for the provision of the APIs under evaluation. As all of these organizations are presently involved in the INSPIRE data provision process, they can provide well-founded input on the deployment effort of both the new APIs as well as the simplified data models in comparison to their already accrued efforts pertaining to existing technologies.

Similar to the developer consultation, as the number of involved parties is necessarily limited, the evaluation types applied will be far more interactive. Most responses to the evaluation criteria will be obtained via ongoing Heuristic Expert Evaluation paired with more structured Interviews aligned with the evaluation criteria. However, we also plan on engaging further potential data providers in the course of our events. Questionnaires are foreseen in order to gain access to this additional information, which will be integrated to the insights gleaned from our internal teams.

6.2.3. User Consultation

Gaining access to the wider user community is one of the challenges of this project. For this purpose, we are organizing various events where the participants will be able to engage with operational examples of the various constellations of APIs and data models being provided by our data providers. Simple tools and scripts for access and use of the available data will be presented, giving the participants the opportunity to interact and experiment with these APIs in a hands-on manner. In addition, project staff will be available for interested users to provide support in understanding and using these resources.

As we see the greatest potential for reduction of required effort pertaining to the usability of the emerging APIs and simplified data models, we will apply the Peer Review evaluation type to the methodologies utilized to evaluate selected functionality. In the user consultation we will focus more strongly on the questionnaires as a method of gaining insights from a larger stakeholder group while identifying the more engaged participants and pulling them aside for more in depth interviews. These insights will be rounded off by our team of experts providing insights based on their experience.

6.3. Analysis

Once the APIs have been deployed and evaluated, the analysis of the evaluation outputs will commence. Before we describe the analysis in detail, we would like to revisit the various dimensions that define the information space created by this evaluation:

- Stakeholder Perspectives: development, deployment and usage aspects are included
    - Provision alternatives: green-fields development vs. configuration of existing software. Full details of this dimension are provided in section 3.1.2 Provision Spectrum.

    - Usage options: accessing APIs via custom software or integrating into existing tools. Full details of this dimension are provided in section 3.1.3 Usage Spectrum.
- APIs: OGC API – Features and SensorThings API, comparison to previous generation OWS
- Data Models: Complex features as defined in the INSPIRE data specifications as well as simplification options
- Available documents and other community resources

Requirements and opportunities related to these dimensions have been collected in accordance with the evaluation criteria, providing not only basic data such as the extent to which the criterion is fulfilled by a specific API or data model but also information providing deeper insights into the ramifications of this criterion and highlight suitable measures for potential future support activities and resources.

Efforts in provision and use have been evaluated in relation to

- the attainment of the evaluation levels
- the APIs in comparison to OWS
- simple vs complex data models

These inputs must be reflected against the degrees of freedom identified in order to guide recommendations, quantify effort and define the contents of the supporting materials (technical guidance, software tools, tutorials, etc.) to be provided.

The core feedback unit will pertain to a specific evaluation criterion as well as a specific stakeholder perspective; e.g. the criterion Embedded Metadata pertaining to the Deployment Perspective. Depending on the evaluation types utilized, different types of feedback will be merged to provide consistent information including fulfillment of criterion, effort information (effort required for provision or effort incurred due to its lack during use) and recommendations for supporting material on this feedback unit.

Once all information has been collated per feedback unit, we will merge this information along the analysis axes in order to generate overview information pertaining to the different aspects being analyzed. In the examples given in Figure 3: Analysis overview of level of achievement, we compare OGC API – Features with SensorThings API for the Stakeholder Perspective User as well as provision and use aspects pertaining to OGC API – Features. The colors, as described in the key, denote the following values:

- Yes: clear agreement that the criterion is fully met.
- Partly: either only certain aspects of the criterion have been fulfilled.
- No: clear agreement that the criterion has not been met.
- Unsure: the stakeholders interrogated were not able to provide a satisfactory answer to the criterion, or there is strong divergence in the feedback to this criterion.
- No Data: the criterion was deemed not applicable to the evaluation dimension.

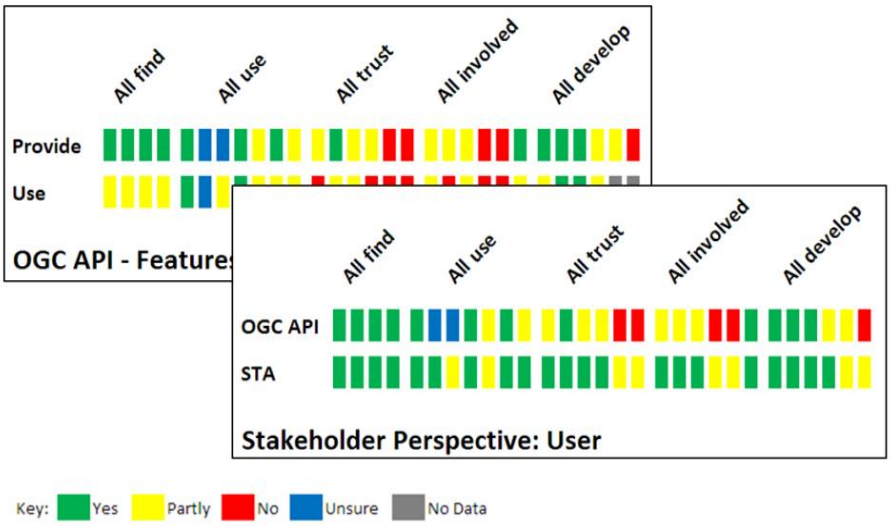


Figure 3: Analysis overview of level of achievement

Effort information collected will first be merged per stakeholder perspective and then evaluated against alternatives along the dimensions described pertaining to effort evaluation above. In addition, effort will be compared between provision and usage stakeholder perspectives, whereby we expect to find a Sweet-Spot somewhere in the middle of the effort spectrum, not too much effort required for provision while supporting the most important usability concerns (See Figure 4: balance between deployment and use effort, please note that the current diagram is a heuristic estimation serving only illustrative purposes. Updated figures will be provided in “D5 Conclusions and recommendations for MS authorities”). As mentioned above, effort will only be qualitatively estimated on a scale of 1 to 5, with one being the least effort, and 5 being the highest. Effort information will also be differentiated into effort relating to initial uptake and effort relating to day to day use.

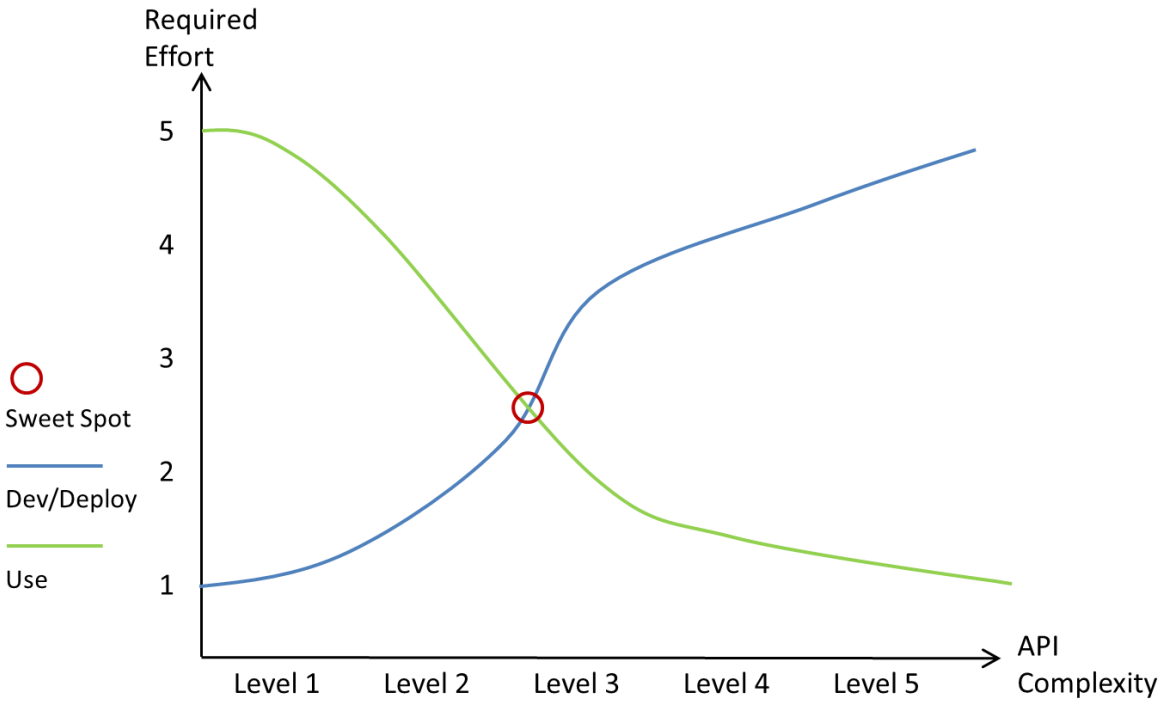


Figure 4: balance between deployment and use effort

Information on information gaps and other resource deficiencies collected through various aspects of the evaluation process will be collected and structured. This will help in the formulation of measures to guide the recommendation and training process.

6.4. Recommendations

Based on the evaluation analysis, the outputs will be summarized and made available in various forms in order to meet the information requirements of the stakeholder groups involved.

For administrative stakeholders concerned with providing the necessary resources for deployment and use of APIs, information on costs and benefits to be expected will be provided, allowing them to evaluate the potential of adoption of these technologies within their own organizations. Note that effort (costs) will not be explicitly quantified, but instead provided on a graduated scale in relation to known efforts. This is in line with the fact that an evaluation is always a comparison against something else, whereby this foil being evaluated against may be abstract criteria or existing systems such as previous generation OWS. This approach allows administrators to apply these relative results to the level of expertise available within their organizations based on experience in the provision of existing systems.

For data providers, the necessary materials required to empower Member State authorities to deploy their data via the new APIs, as well as to understand the benefits and pitfalls of data simplification options will be provided. These materials will include a rich collection of reference links, tutorials and technical guidance documents, providing a central access point for all required information. The various

pieces of information collected within the GitHub Wiki in order to support the development and deployment teams will form the core of this resource. Where possible, we will endeavor to collect existing resources; where gaps have been identified, resources will be created either through the project team or triggered via other initiatives.

For data users, different levels of information will be required in order to address the different types of stakeholders. Based on feedback from events, information will be regularly integrated into the GitHub Wiki providing a core for the final resource collection. As described above pertaining to data providers, existing resources will be complemented as required.

A Webinar will be held in order to widely disseminate the outcomes of this project, whereby we may foresee some thematic splitting in order not to overload the Webinar format. Separate Webinars pertaining to provision of a specific API may be beneficial to empower Member State authorities to deploy and utilize the OGC APIs. Training material will be provided as required.

7. Annex A - Extended evaluation criteria

This annex lists the evaluation criteria, with questions illustrating how the criteria apply to the three different stakeholder perspectives.

7.1. Level 1: All find

This level corresponds to very basic non-standardized data provision, data is available together with simple examples. This level focuses on in-house use of APIs. Requires minimal effort in provision but is difficult to impossible to use.

Relevant questions: Is the API mainly designed for in-house use? Did the API happen by accident? Did the API evolve without relevant oversight or adherence to standards? Was the API designed based on a large set of realistic use-cases?

7.1.1. Single Entry Point

Is all information available from a single source (the portal), either directly or through links?

Finding all data one needs is a key requirement for any task. Ideally, there is a single place (the portal) through which all required information can be reached. For development this pertains to the API description, for deployment the data model definition, and for API use the data exposed through the API and the metadata required for understanding the data.

Effort required: 1.

Benefit provided: 4

Dev: Does the API documentation contain all required information for implementing the API? Does it contain clear references to all other APIs or standards that are used as part of the API?

Deploy: Does the API documentation contain all required information for modeling the use-case data? Can metadata be published through the API or does it have to be supplied through other means? Does the API have a single entry point, or do different end-points need to be documented elsewhere?

Use: Is all the data provided by the API available through a single entry point (e.g. starting page?). Is all metadata available through the API, or does this have to be downloaded from a separate location?

7.1.2. Documentation

Is updated documentation available?

Documentation is the main source to get information about the API. It should answer all the questions arising for all interested in the API, be it the software developer implementing the server, the publisher mapping his data to the data model, or the user requesting data. The documentation should be clear and precise, so that no “trial and error” is needed.

Effort required: 2.

Benefit provided: 5

Dev: Is the API Standard clearly documented (specified), to allow an unambiguous implementation?

Deploy: Does the documentation clearly explain how to map use case data to the data model of the API?

Use: Is the API clearly documented and understandable? Does the documentation describe how the API can be used? Does the documentation offer the information needed to realize the intended use case? Are the usage constraints clearly described?

7.1.3. Example Requests

Are there examples of API requests as part of the documentation?

The interaction point with an API are requests, created by a client-(application), sent to the API implementing server, processed and the result is sent back. The request contains all information, which describes the users information needs. Example requests help with understanding the documentation from all perspectives.

Effort required: 2; Community input possible.

Benefit provided: 5

Dev: Are there example requests available which show a potential usage of the API? Is it clear what incoming requests to expect, based on the examples?

Deploy: Do the example requests illustrate how the data modeling influences the use of the data? Are there example requests available which show a potential usage of the API? Are all aspects covered by the sample requests?

Use: Are example requests available demonstrating the functionality of the API? Are different use cases covered by the sample requests? Can the samples be used as a basis for one's own use-case specific requests?

7.1.4. Example Data

Are there examples of the data returned by API requests?

After processing the received request, the server returns the requested data. It's advantageous if the example data matches the example requests, and is described in detail.

Effort required: 2; Community input possible.

Benefit provided: 5

Dev: Is there example data available? Is it possible to use the example data to test the API implementation? Is the example data available in a format that is expected as response from the server?

Deploy: Is there example data available which can be loaded into the implementation? Is this example data covering all aspects of the use-case? Is there an easy alignment of the example data and the data that should be exposed by the API? Are there example responses available which can be used as a reference to validate the results provided by the deployed server?

Use: Are there example responses available that show possible results from the server? Is example data available which can be easily imported into an existing implementation, to test the use of the API?

7.1.5. Discoverability

Is it possible to discover deployed instances of the API based on the resources provided?

What means are available for discovering deployed instances of the API? Are dedicated services such as the OGC Catalogue Service Web (CSW) required for discovery? Can deployed instances of the API be discovered via normal search engines following W3C Data on the Web Best Practices (DWBP)[21] and Spatial Data on the Web Best Practices (SDWBP)[22]?

Note: this criterion was added to reflect the organizational and standardization background against which this evaluation is being performed.

Effort required: 2.

Benefit provided: 5.

Dev: Are mechanisms for making deployed instances of the API discoverable available? Does making the API discoverable by search engines greatly increase the implementation effort?

[21] https://www.w3.org/TR/dwbp/
[22] https://www.w3.org/TR/sdw-bp/

Deploy: Must discovery criteria be explicitly specified, or is it automatically extracted from the data resources? Must additional discovery services be deployed, or is discovery via search engines integral to the API?

Use: Is it easy to discover deployed instances of the APIs providing relevant data? Can deployed instances of the API be discovered via search engines or must dedicated services/APIs be utilized for discovery?

7.2. Level 2: All use

This level corresponds to basic standardized data provision. Data is available in a standardized format, basic standards based access functionality is provided. This level focuses on providing data to external users. Medium effort in provision, use is possible, but not optimal.

Relevant questions: Has the API been reviewed by external users? Has the API been designed for providing data to external users?

7.2.1. JSON or XML

Does the API support the use of JSON and/or XML?

Data needs to be represented in a serialized form to be transmitted. For web-based APIs, JSON and XML are established as de-facto-standard, since they're human readable and many implementations exists for various programming languages, which allows easy integration and reuse. These aspects are relevant for server development and API use, but not for the deployment.

Effort required: 1.

Benefit provided: 4

Dev: Should the response of the API provide data in JSON or XML? Does the incoming request containing JSON/XML data? Is the JSON/XML-representation specified in a way that allows the easy implementation?

Deploy: Not relevant.

Use: Is the provided data available in JSON/XML? Is JSON/XML used to create requests? Are both representations available? Is it easy to reuse existing tools/libraries to parse the representation?

7.2.2. Data License

Are data license details given through the API?

The main value of an API is provided through the data that is published through the API. Therefore, it's important that the license of the data is also available through the API itself. The data license is usually a legal document, and in most cases this document will be linked to from within certain API responses. Ideally, the data license is based on a standard licensing scheme such as CC BY.

Effort required: 1.

Benefit provided: 3

Dev: Is it clear how data license information should be made available through the API?

Deploy: Is there a way to make the data license available through the API? Is it possible to add all required licensing information?

Use: Is the data license available through the API? Is it clear where to find the data license in the API? Is the data license easy to find and understand? Is a standard license (such as CC BY) being used?

7.2.3. Terms of Use

Are Terms of Use clear and easily accessible?

In addition to data license, the terms of use should be available. A data provider should be able to specify how and with which constraints an API service can be used, and it should be clear to users of the API where to find this information. The terms of use are usually described in a legal document, and in most cases this document will be linked to from within certain API responses.

Effort required: 1.

Benefit provided: 3

Dev: Is it clear how the terms of use should be made available through the API? Does the API describe technical mechanisms to enforce the terms of use?

Deploy: Can the Terms of Use be configured in the server? Is there a technical mechanism to enforce the Terms of Use (e.g. rate limiting)? Is it possible to enforce the terms of use with external mechanisms not defined by the API?

Use: Are the terms of use accessible through the API? Is it clear under which constraints the API can be used? Is it clear when technical mechanisms that enforce the terms of use take effect?

7.2.4. Embedded Metadata

Does returning data include metadata?

Getting data is often not sufficient. Additional metadata is needed to use and interpret the data correctly. E.g. what is represented by the data? What are the units? The API must allow the developer to enable the provider to provide the metadata required by the user to interpret the data.

Effort required: 3 – 4.

Benefit provided: 4

Dev: Is it clear how metadata should be included in the API? How flexible is the mechanism for providing metadata?

Deploy: Is it possible to integrate all relevant metadata into the API and its data model? Is it clear how this should be integrated?

Use: Is the relevant metadata available or is some information missing? Is it possible to automatically consume/process the provided metadata?

7.2.5. Authentication

Does the API support authentication/authorization?

Not all data should be available publically. To limit access to authorized users, authentication/authorization (auth*) mechanisms exist. To ease integration, the auth* methods should be based on existing, well-known protocols (e.g. Oauth[23] or OpenID-Connect[24]). After authenticating, the authorization mechanisms define exactly which data the user is allowed to see.

Effort required: 2 – 4.

Benefit provided: 3

Dev: Does the API specify how to deal with authentication / authorization? Does the API specify security mechanisms? Does the API limit the used security mechanisms to the specified ones? Is it clear how authorization should be applied to the data model?

Deploy: Is it possible to enable/configure an auth* method on the server? Is it possible to configure multiple authentication methods? Is it possible to integrate with existing identity management systems?

[23] https://en.wikipedia.org/wiki/OAuth
[24] https://en.wikipedia.org/wiki/OpenID_Connect

Can all security aspects be handled by the auth* method? Is it possible to configure authorization methods? Is it clear how authorization methods interact with the data model?

Use: If using the API needs some kind of authentication, is it based on existing methods that can be reused? Is there a clear way to discover which authentication method should be used? Is it possible to seamlessly manage/use the provided authentication method while accessing the API?

7.2.6. API Standardization

Is the API itself standardised, and is this specification openly available?

Note: this criterion was added to reflect the organizational and standardization background against which this evaluation is being performed.

For an API to become widespread, it is helpful if the API is formally adopted as a standard by a well-respected, international standards body. Furthermore, an API standard rarely stands alone. Most standards refer to other standards for specific aspect. For example, in most standards, whenever a date or time is used, the encoding is done according the ISO 8601 standard. Referencing existing standards usually reduces the effort required for implementing the standard and makes it easier for clients to use the standard, since they can use common libraries that implement these referenced standards. It also reduces the effort of mapping data models, since standardised building blocks are likely to already be in use.

Effort required: 5; Community input required.

Benefit provided: 5

Dev: Is the API itself standardised? Does the API re-use existing standards that are implemented/supported by mature libraries? Are the standards that the API references readily available?

Deploy: Is the API itself standardised? Does the API data model re-use existing data model standards? Are the standards that the API references readily available? Is it possible to map the existing data model to the API? Is this easily possible?

Usage: Is the API itself standardised? Does the provided API correspond to the API standard and the defined data model? Does the API re-use existing standards that are implemented/supported by mature libraries? Are the standards that the API references readily available?

7.2.7. Suitability

Is the API suitable for the intended use?

Note: this criterion was added based on the criteria available from within the ISO 25101 Standard.

For an API to be deployed and used, it has to be suitable for the intended use case. From the deployment perspective, this means the API has to be implemented in server software that fits in the deployment landscape of the data provider and that can be connected to, or loaded with, the data that the data provider intends to publish. From the user perspective, this means that the user must be able to request, in a suitably efficient manner, the data that he needs.

Effort required: 3.

Benefit provided: 4

Dev: Not relevant.

Deploy: Is it possible to deploy the implementation to the existing infrastructure? Can the existing infrastructure be reused? Is it possible to use existing data sources and expose them, using the API implementation?

Use: Having a given use-case is it possible to realise all involved aspects with the given API? Is the overhead involved with the use of the API acceptable?

7.3. Level 3: All trust

This level corresponds to mature standardized data provision. Encompassing data is available in complex standardized format, powerful and mature standards based access functionality is provided. Fairly large effort in provision, use is well supported.

Relevant questions: Has the API been designed for use in a diverse set of use cases, in a diverse set of environments? Can the API support the complexity of data models required for real-world use cases? Has the API been designed for use with large data sets?

7.3.1. Query and Analytics API

Does the API include Querying and Analytics?

Note: this criterion was modified from the original to include query functionality together with analytics

APIs are often used to get data from a service. Usually only a subset of the available information is of interest. Therefore, the API should contain querying and analytic capabilities. First, this includes a suitably powerful filter mechanism to limit the response to those parts of the data that the user is interested in. Second, this includes a mechanism in the API to do basic analytical calculations, e.g. some aggregation functions.

Effort required: 4.

Benefit provided: 5

Dev: Is the filter and analytic mechanism of the API clearly defined and easy understandable? Is it easy to implement this functionality? Are libraries available for development of these filters?

Deploy: Is the available data source able to handle the filter/analytic requests coming from the API server implementation? Or is there a need to fully import and transform the data into the server to have the full analytics API available?

Usage: Does the API offer a possibility to filter for the needed data? Is it possible to map the use-case specific request to the supported filter mechanism? Does the API offer the needed aggregation functionality or is some post-processing needed? Are those mechanisms easy to understand?

7.3.2. Error Handling

Is Error Handling in place and documented?

While using an API errors might occur. Either there's an issue with the server itself, or the data sent by the user isn't correct or doesn't match the available data. To allow a user to handle these errors, the API should specify how errors are reported back to the user. At the same time, the error message should not expose sensitive internal information about the server deployment.

Effort required: 2.

Benefit provided: 4

Dev: Are there standard error codes and error responses defined? Is standard behavior defined for when errors are encountered? Is the error behaviour aligned to existing best practices?

Deploy: Is it possible to create an inconsistent data model that will cause errors for the client? Are there good ways to find errors in the data modeling? Is it clear which error messages may expose internal information?

Use: Is there a well defined error behaviour? Are error messages and error codes defined in the API? Is it clear how to handle specific errors? If there is an error in the usage, is some information provided, how to fix adapt and fix the usage?

7.3.3. Performance and Cache

Can the API offer sufficient performance and does it support caching?

Note: this criterion was modified from the original to include performance functionality together with caching

APIs may have inherent performance bottlenecks that become apparent when deploying and using the API with large data sets or a large number of users. To increase performance and efficiency, caching mechanisms may be used.

Effort required: 2 – 4.

Benefit provided: 3

Dev: Is caching of queries part of the API standard? Is the caching mechanism clearly specified? Is the caching mechanism defined in a way that existing implementations (or even existing infrastructure) can be reused? Are there inherent performance bottlenecks in the API? Is it possible to create a server that scales horizontally (by adding more back-end servers)?

Deploy: Is there a way to use existing caching infrastructure, external to the server software? Are there limits to the amount of data served by a single server instance? Can horizontal scaling be used to increase the number of clients served?

Use: Is the caching mechanism clearly described? Is it clearly defined if and when a user might get outdated data? Does the standard offer the possibility to establish a client-side caching?

7.3.4. Background Support

Is the development and maintenance of the API supported by a big, stable entity or company?

Deciding to use a specific API requires investments on all sides (dev, deploy, use). Thus, for the future development of the API itself and the implementation is important, to be sure that the API will still be relevant in the future. A big entity or company in the background increases this probability.

Effort required: 1 – 5; Community input possible.

Benefit provided: 4

Dev: Is the API standard supported by a big entity or company?

Deploy: Is the entity developing the API well-known, and is it thus likely that there are multiple server implementations of the API? Are there multiple, independent server implementations?

Use: Is there big entity or a company in the background which supports the standard?

7.3.5. Availability

Is it easy to integrate into existing workflows and toolsets?

Many users and domain experts have well established workflows that use existing tools and (desktop) software packages. Switching to a different data source or API may break these workflows, which would greatly reduce the incentive for users to switch to the new API. These tools can include those used by providers to import data from external or primary sources into the local data infrastructure, or those used by data users to visualise or otherwise use the data.

Note: this criterion was modified from the original to include availability of relevant tooling.

Effort required: 4; Community input required.

Benefit provided: 4

Dev: Not relevant.

Deploy: Are there tools available that help with the data mapping? Are there tools available that help with the data import into the server?

Use: Are there plugins available for the client software that is popular in the domain of the API.

7.3.6. API Data Validation

Can the data returned by the API be validated?

For interoperability it is important that the data returned by an API conforms to the data model defined by the API. It is beneficial if there is a way to automatically check this, for example by checking the JSON against a JSON Schema, or XML against an XML Schema Definition.

Note: this criterion was added to reflect the organizational and standardization background against which this evaluation is being performed.

Effort required: 3; Community input possible.

Benefit provided: 4

Dev: Can a developer verify that the implementation returns correctly modelled data?

Deploy: Is there a way to verify that the data mapping is done correctly by validating the returned data?

Use: Can a user easily verify that the data being provided by the APIconform to a given definition?

7.4. Level 4: All involved

This level pertains more to the maturity of the wider ecosystem built up around the standardized specifications. The aspects described go beyond the control of the individual data providers and thus must be supported through diverse stakeholder communities. Achieving this level facilitates both the provision and usage perspectives.

Relevant questions: Can support for the API be seen as a community effort?

7.4.1. SDK Availability

Are API SDK's available for one or more environments?

Software Development Kits (SDKs) simplify the interaction with the API on the development and client side. They contain libraries that reduce the amount of code that needs to be created to interact with the API and help reuse existing work and integrating the API in new contexts.

Effort required: 3; Community input possible.

Benefit provided: 2

Dev: Is there an SDK available which implements the API? Are all aspects covered (e.g. parsing the request, parsing filters, …)? Is the SDK usable regarding license and technical implementation?

Deploy: Not relevant.

Use: Is there and SDK available for using the API? Are all aspects covered (e.g. parsing the request, parsing filters, …)? Is the SDK usable regarding license and technical implementation?

7.4.2. Code Examples

Are there examples of code in one or more commonly used programming languages?

Like SDKs, code examples simplify and clarify the interaction with an API. They show in detail how certain aspects of the API should be used and can often directly be executed to see the covered features live in action.

Effort required: 3; Community input possible.

Benefit provided: 4

Dev: Are there code example available, showing how to to implement specific aspects of the API? Are those examples in one or more programming languages? Is it possible to transfer those examples to other programming languages?

Deploy: Not relevant.

Use: Are there code example available, showing how to use specific aspects of the API? Are those examples in one or more programming languages? Is it possible to transfer those examples to other programming languages?

7.4.3. Community

Is there a growing community to consult if needed?

A big, active and friendly community can be, in addition to the documentation, an additional source of information. Questions, best-practices and issues can be discussed within a community to spread available knowledge and provide support.

Effort required: –; Community input required.

Benefit provided: 4

Dev: Is there a community available which can help solving API specific issues while implementing the standard? Are the provided communication channels adequate to consult the community? Is the community friendly and open for new members?

Deploy: Is there a community available which can help solve issues with the data modeling and mapping? Are the provided communication channels adequate to consult the community? Is the community friendly and open for new members?

Use: Is there a community available which can help solving API specific issues while using the standard? Are the provided communication channels adequate to consult the community? Is the community friendly and open for new members?

7.4.4. Playground

Is there an API Playground for testing and getting familiar with the API?

Learning by doing and practically testing the usage of the API helps getting more insights. A playground helps with this. Such a playground can range from a public server that anyone can access, to a docker image that can be quickly deployed with standard settings, to a one-click-install installation package that can run on a desktop PC.

Effort required: 2; Community input possible.

Benefit provided: 3

Dev: Is there a playground available which can serve as an example/reference implementation which can be used to match the own implementation?

Deploy: Is there a playground available showing a sample implementation of the API? Is it possible to simply adapt the content of the playground to your use-case?

Use: Is there a playground service, implementing the API standard available? Does this playground contain sample data, allowing to test all aspects of the API? Is it possible to simply adapt the content of the playground to your use-case?

7.4.5. Linked Documentation

Is documentation linked to code examples and back again?

A documentation that links to code examples, example data and request (and back) helps the reader to better understand the API.

Effort required: 2; Community input possible.

Benefit provided: 3

Dev: Does the documentation link to the examples? Is there also a backward link from the examples to the documentation?

Deploy: Does the API documentation link to data modeling examples, and do those examples link back to the API documentation?

Use: Is there a documentation available which links the documentation to the samples? Is there also a backward link from the examples to the documentation?

7.4.6. API Evolution

Can a developer, data provider or user provide feedback on issues with the data model or API functionality and are custom extensions to the API foreseen?

Note: this criterion was added to reflect the organizational and standardization background against which this evaluation is being performed.

APIs are rarely perfect and finished in their first incarnation. They also always have to strike a balance between the diverse requirements of many different use-cases on the one hand, and complexity on the

other. Because of this, it is important that developers, providers and users have a way to provide feedback on issues, deficiencies and unclarities in the API. It is important that these issues are addressed in future versions of the API. Similarly, it is very helpful if the API has clear extension points so that the API can be extended for certain use cases that require functionality that is not in the API.

Effort required: 2; Community input required.

Benefit provided: 2

Dev: Can issues with the API be reported? Are these issues takes into account in the development of the next version of the API? Are standard extension points foreseen by the data model and the API?

Deploy: Can issues with the data model of the API be reported? Can the data model be extended to better fit a use case?

Use: Can issues with the API be reported? Is it clear which API and data model extensions are active on a given server instance?

7.5. Level 5: All develop

This level describes a well developed and mature ecosystem built up around the standardized specifications. The aspects described go far beyond the control of the individual data providers and thus must be supported through diverse stakeholder communities. Achieving this level facilitates both the provision and usage perspectives.

Relevant questions: Can (further) development of the API be seen as a community effort?

7.5.1. Code Visible

Is code visible/can be cloned?

Having access to the source code of a reference implementation allows a deeper understanding of the implementation. It offers the possibility to check if specific behaviour was intended or is a bug. Available open-source code with a suitable license offers the possibility to add own changes, so that there is no dependency on a third-party.

Effort required: 5; Community input required.

Benefit provided: 3

Dev: Is the source code of a reference implementation of the API available? Is the source code well structured, so that issues that arise during the implementation can be checked against this reference implementation?

Deploy: Are there open-source server implementations of the API? Is the source code well structured and aligned to best practices in software development?

Use: Are there open-source client libraries available? Is the source code well structure, so that issues that arise during the usage can be checked in other implementations?

7.5.2. Bug Tracker

Can Bugs, issues and suggestions be reported in a public place and is this dialogue public?

Though an API is not software and can thus not have “bugs” in the traditional sense, an API can still have inconsistencies, errors and unclear definitions. Often these issues are not noticed until the API is applied in specific use cases, or implemented by multiple people. Having a Bug Tracker publically available offers a place, where to report issues and to track discussions and solutions. Having such a system openly available allows input from a wider community, helping the system to evolve to support a wider user community. It also makes it easier to collaborate on extensions.

Effort required: 1; Community input possible.

Benefit provided: 4

Dev: Is there a Bug Tracker available to report inconsistencies or errors in the API definition and discuss and design future extensions?

Deploy: Is there a Bug Tracker available to report inconsistencies or omissions in the data model that the API defines? Is there a place to discuss data model extensions?

Use: Is there a Bug Tracker available to report inconsistencies or errors in the API definition? Can feature request and extension proposals be discussed there?

7.5.3. API License/reuse

Is the API's license known, are parts of the API covered by patent claims, and does it allow further development and re-use?

Parts of APIs can be covered by patent claims, making it impossible to implement the API without paying royalties. The license of the API is important and might be a blocker, if the API needs to be re-used or if further development of the API is required. Ideally, if a license is required this should be based on a standard licensing scheme such as CC BY.

Effort required: 2; Community input possible.

Benefit provided: 3

Dev: Can the API be implemented without using patented technologies? Is the license of the API known? Is the license of the API based on a standard and open licensing scheme such as CC BY? Is it possible/allowed to re-use the API? Is it permissable to further develop the API?

Deploy: Is the license of the API known? Is it possible/allowed to re-use the API? Is it possible/allowed to further develop the API?

Use: Is the license of the API known? Is it possible/allowed to re-use the API? Is it possible/allowed to further develop the API?

7.5.4. Development Roadmap

Is the API's development roadmap known and is it visible for all?

A roadmap can help to understand the further development direction of the API and to know what to expect in the (near) future.

Effort required: 3; Community input possible.

Benefit provided: 2

Dev: Is there a development roadmap of the API available? Is it clear what to expect in the following months/years?

Deploy: Is there a development roadmap for the server implementation available? Is there a clear direction for the future?

Use: Is there a development roadmap of the API available? Is there a development/deployment roadmap of the data provider available?

7.5.5. Linked Data Ready

Is the data provided structured in a Linked Data ready manner, i.e. JSON-LD?

Note: this criterion was added to reflect emerging technological advances to be expected within the stakeholder community.

Linked Data is a technology that looks very promising and that has been on the horizon for some time now, with parts and concepts of it finding their way into APIs and data models. While it is not yet practical to have an API that fully employs all Linked Data principles, it is possible to design the API and data models in a way to allows Linked Data adoption in the future.

Effort required: 3; Community input required.

Benefit provided: 3

Dev: Can Linked-Data principles be applied to the API responses without violating the API specification?

Deploy: Can Linked-Data principles be used in the data model?

Use: Not relevant.

7.5.6. Test Framework available

Can conformance to the API be formally tested?

Note: this criterion was added to reflect the organizational and standardization background against which this evaluation is being performed.

Test Frameworks can be used to verify the implementation and deployment of the API. This helps interoperability by insuring the deployed services implement the API correctly.

Effort required: 4; Community input required.

Benefit provided: 3

Dev: Is there a test framework available to automatically verify the compliance of a server implementation with the API? Is the test framework publicly available so that the test can be run easily?

Deploy: Is there a test framework available to automatically verify that the server is deployed compliant with the API? Is the test framework publicly available so that it can be run easily?

Use: Is there a test framework available to test if the usage is compliant with the API (e.g. by providing a validator for requests)? Alternatively, is there some sort of formal certification mechanism for showing compliance being met as an outcome of a validation performed by the data provider.

8. Annex B - SQuaRE: ISO 25010 Model

For evaluating the quality of a software product the ISO 25010 Standard on Systems and software engineering - Systems and software Quality Requirements and Evaluation (SQuaRE) is well established. The quality is defined as “the degree to which the system satisfies the stated and implied needs of its various stakeholders, and thus provides value.” (from ISO 25010). However this indends to be applied to software components, it can be reused to evaluate APIs as well. SQuaRE does not define specific properties, concrete concepts or implementations. It defines a set of high level characteristics software systems (and also APIs) should fulfill. We’ll use these characteristics and set them in the context of the API evaluation while our main evaluation metric is the “Five Level Open Data API evaluation model”. In contrast to the abstract ISO standard, the five level model provides specific properties against which the APIs can be evaluated. Since SQuaRE defines high level characteristics, without specific implementations, this offers the possibility to take those characteristics and map those to the properties, defined in the Five Level model. On the one hand this offers to understand the bigger context of the property. On the other hand we can prove that the list of properties covers all quality characteristics.

The ISO 25010 splits the quality characteristics into two groups, which we apply to our three different perspectives (API development, deployment, use). Those two groups are:

- **Quality in use model:** *These characteristics relate to the interaction with a system.*

  We evaluate the interaction with the API in the *API use* perspective.

- **Product quality model:** *These characteristics relate to the impact the system/software has on stakeholders.*

  The product quality model relates to the implementation of the API standard and is considered in the *API deploy* perspective.

Classifying the *API development* perspective in context is not obvious. It clear that the result of the API development should keep the product quality model in mind. Anyhow, the characteristics of the quality in use model can be applied, since similar there are aspects on the API provider (development) and consumer (use) side.

In this report we omit a detailed explanation of the characteristics. Our focus is to set the SQuaRE characteristics in the context of the five level model, to prove the completeness of our evaluation criteria. For a complete overview and definition of the characteristics, we refer to the corresponding ISO document.

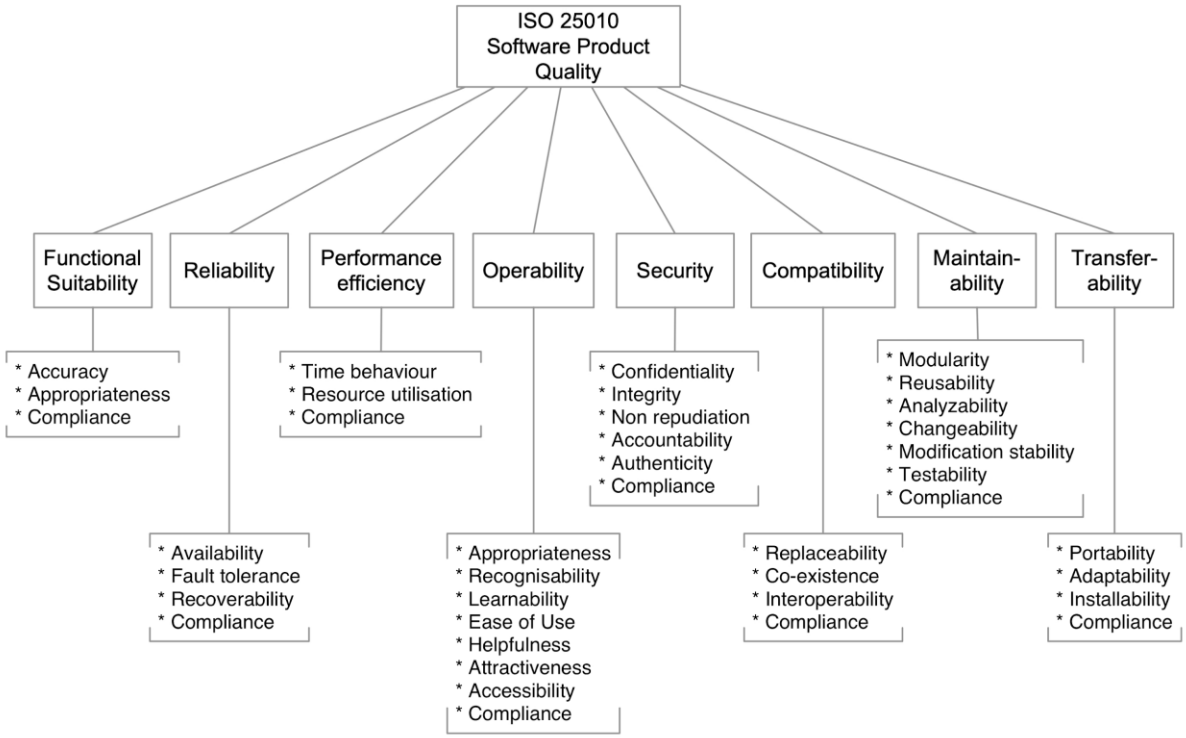
The **quality in use model** is split into five characteristics, with partly additional sub-characteristics:

1. Effectiveness
2. Efficiency

3. Satisfaction
    a. Usefulness
    b. Trust
    c. Pleasure
    d. Comport
4. Freedom from risk
    a. Economic risk mitigation
    b. Health and safety risk mitigation
    c. Environmental risk mitigation
5. Context coverage
    a. Context completeness
    b. Flexibility

Some of those characteristics (Effectiveness, Efficiency, Satisfaction) can be directly applied to APIs, whereas it's not that obvious for *Freedom from risk* and *Context coverage*. Anyhow *Freedom from risk* is important when implementing/using an API in a slightly altered version: "Is there a legal risk in using the API? Is there a technical risk in using the API, e.g. the used concepts are outdated? Is there a risk that further developments are discontinued?". The context coverage aims at the targeted use case: "Can the use-case be implemented using the API? Is the API flexible enough?"

The **product quality model** is summarized in the following picture:



Taken from: https://jaxenter.de/req4arcs-qualitaet-faellt-nicht-vom-himmel-86493

A crucial point in the API Deployment perspective is the implementation serving the API. Since this is a software application, the product quality model can be directly applied.

	ISO	
Criteria	**Product quality model (API Deployment)**	**Quality in Use (API Usage)**
Level 1: All find		
Single Entry Point	Usability	Effectiveness, Efficiency
Documentation	Usability	Effectiveness, Efficiency
Example Requests	Usability	Effectiveness, Efficiency
Example Data	Usability	Effectiveness, Efficiency
Level 2: All use		
JSON or XML	Usability, Maintainability, Portability	Efficiency
Data License	Compatibility (legal)	Freedom from risk
Terms of Use	Compatibility (legal)	Freedom from risk
Embedded Metadata	Usability, Maintainability, Portability	Effectiveness, Efficiency
Authentication	Security	Freedom from risk
* **API Standardization**	Usability, Maintainability, Portability	Freedom from risk
* **Suitability**	Compatibility (*technical)	Context coverage
Level 3: All trust		
* **Query and Analytics API**	Functional Suitability, Performance efficiency	Effectiveness, Efficiency
Error Handling	Usability, Reliability	Freedom from risk
* **Performance and Cache**	Performance efficiency	Effectiveness, Efficiency
Background Support	Reliability	Freedom from risk
* **Availability**	Reliability	Freedom from risk
* **API Validation**	Reliability	Freedom from risk
Level 4: All involved		
SDK Availability	Functional Suitability, Usability	Satisfaction, Efficientcy
Code Examples	Usability	Satisfaction, Efficientcy
Community	Usability	Satisfaction, Efficientcy
Playground	Usability	Satisfaction, Efficientcy
Linked Documentation	Usability	Satisfaction, Efficientcy
* **API Evolution**	Usability	Freedom from risk
Level 5: All develop		
Code Visible, Portability	Reliability, Maintainability	Freedom from risk
Bug Tracker	Usability, Reliability	Freedom from risk
API License/reuse	Compatibility (legal), Reliability	Freedom from risk
Development Roadmap	Reliability, Portability	Freedom from risk
* **Linked Data Ready**	Usability, Maintainability, Portability	Freedom from risk
* **Test Framework available**	Usability, Reliability, Maintainability	Freedom from risk

Table 5: shows how the criteria in the Five Level evaluation model relate to the characteristics defined by ISO 25010. A comparison shows that by adding the “Suitability”-criteria, all aspects of the ISO standard are covered by our Five Level model as well.