The SAREF Pipeline and Portal-An Ontology Verification Framework

Maxime Lefrançois, David Gnabasik

The SAREF Pipeline and Portal — An Ontology Verification Framework

Maxime Lefrançois[0000−0001−9814−8991]
and David Gnabasik[0000−0002−6052−7782]

Mines Saint-Étienne, Univ Clermont Auvergne, INP Clermont Auvergne, CNRS,
UMR 6158 LIMOS, F - 42023 Saint-Étienne France
{maxime.lefrancois,david.gnabasik}@emse.fr

Abstract. The Smart Applications REFerence Ontology (SAREF) defines a modular set of versioned ontologies that enable semantic interoperability between different Internet of Things (IoT) vendor solutions across various IoT industries. The European Telecommunications Standards Institute Specialist Task Force (ETSI STF) 578 recently completed the "Specification of the SAREF Development Framework and Workflow and Development of the SAREF Community Portal for User Engagement". This project specifies the development pipeline and workflow needed to accelerate the development of SAREF and its extensions along with the development of software that automates the generation of ontology portal content from SAREF sources on the public ETSI Forge. This paper describes the SAREF Pipeline that provides an efficient and robust support infrastructure for the Continuous Integration and Delivery of semantic ontology development.

Keywords: SAREF · ETSI · SmartM2M · Ontology · Semantic · IoT · Continuous Delivery · Quality control · Pipeline.

1 Introduction

The Smart Applications REFerence Ontology (SAREF) defines a modular set of versioned ontologies that enable semantic interoperability between different Internet of Things (IoT) vendor solutions across various IoT industries [5,17]. SAREF was promoted by the European Commission in collaboration with the European Telecommunications Standards Institute (ETSI) *SmartM2M* technical committee to design a common data model to limit the fragmentation of the Internet of Things (IoT) by enabling interoperability between solutions from different vendors across various IoT industries, thus contributing to the development of the global digital marketplace and the European data spaces.

The value of SAREF is strongly correlated with the size of its community of users, and also to the agility of the SAREF developers to improve the SAREF ontologies and react to raised issues. As such, SAREF users' community and the industry actors need be attracted to SAREF with clear web documentation and a clear indication about how to provide their input and the kind of input that

they can provide. ETSI Specialist Task Force (STF) 578 recently completed the "Specification of the SAREF Development Framework and Workflow and Development of the SAREF Community Portal for User Engagement". The ultimate project goal is to enable SAREF users in various industries to contribute to and maintain SAREF without requiring specialized or advanced ontology engineering skills, so as to lower the level of support needed by ETSI members, and in particular SmartM2M members. Experts participating in STF 578 specified the general development framework for the SAREF ontology and its extensions, generally referred to as *SAREF projects*, in the ETSI TS 103 673 technical specification [13], and the sources of SAREF were migrated to the public ETSI Forge portal `https://saref.etsi.org/sources/`.

This paper provides an overview of this ETSI TS 103 673 technical specification, with an emphasis on the specification of requirements for *SAREF project repositories*: git repositories that contain the sources of a SAREF project. It then provides a functional and technical overview of the *SAREF Pipeline* software, that automates checking the compliance of SAREF project repositories against this specification, and automates the generation of the static documentation website. The SAREF pipeline is openly available under an open BSD-3-Clause license at `https://saref.etsi.org/sources/saref-pipeline/`. It has already been used to accelerate the development of version 3 of SAREF and the 12 published SAREF extensions, and to generate the contents of the public SAREF community portal available at `https://saref.etsi.org/`

The rest of this paper is organized as follows. Section 2 describes related work. Section 3 describes the modular architecture of the SAREF ontology core and its extensions. Section 4 provides an overview of ETSI TS 103 673, especially who and how SAREF ontology development proceeds, and which rules SAREF project repositories must comply to. Section 5 describes the SAREF Pipeline software and how it is configured for continuous integration and delivery. Section 6 qualifies the impact, reusability and availability of the SAREF Pipeline and discusses current work. Section 7 offers our conclusions.

2 Related Work

Automation in ontology engineering aims to accelerate the production of high quality ontology artifacts by automating manual or redundant tasks and preventing bad releases. For example linting tools[1] supporting the ontology editing process limit the difficulty of editing ontologies of good quality. Examples include Jena Eyeball,[2] and RDFLint[3]. The latter is integrated in the extension

[1] Linting, named after a UNIX pre-processor command for the C language, is an approach that consists of statically analyzing software source code to detect errors, bugs, or style mistakes.

[2] `https://jena.apache.org/documentation/archive/eyeball/eyeball-manual.html`

[3] `https://github.com/imas/rdflint`

RDF language support via rdflint[4] of Visual Studio Code, and allows to execute SPARQL queries, to validate SHACL constraints, or to validate that literals are well formed. Command line interfaces available in software such as Apache Jena[5] also help automating common tasks in ontology development.

The ROBOT tool [22] developed by the Open Biological and Biomedical Ontologies (OBO)[6] community provides ontology processing commands for a variety of bio-medical / disease research tasks such as commands for converting formats, filtering axioms, invoking a reasoner for logical validation and automatic classification (where all direct inferred *subClassOf* axioms are added to the ontology), creating and extracting import modules, verifying the correct execution of SPARQL unit test queries, and running reports. Although ROBOT itself is not a workflow manager, the various ROBOT commands are often combined into automated workflows using a separate task execution system such as GNU Make. The main release artifacts are an OWL file and an OBO file. Since ROBOT was designed to enforce many of the OBO Foundry conventions, ROBOT helps guarantee that released ontologies are free of certain types of logical errors and conform to standard quality control checks, thereby increasing the overall robustness and efficiency of the ontology development life cycle. ROBOT also employs the OWL API library. Whereas ROBOT facilitates the ontology engineering in the bio-medical research community, SAREF's engineering mandate is the unified interoperability of IoT sensor networks from multiple industries and use-cases.

Authors in [4] identified data standards and reporting formats that use version control and summarize common practices in earth and environmental sciences. Just as Agile methods aim to improve collaborations between software project customers and developers, DevOps methods improve collaborations between developers and IT operations professionals. Jenkins, Travis CI, Circle CI, Gitlab CI/CD, Github Actions, are all frameworks that allow to specify task pipelines that will be executed automatically when, for example, a commit is pushed to the server. Before the democratization of these frameworks, a few preliminary approaches were proposed in the ontology engineering community using Github applications[7]. For example VoCol [20] or OnToology [2]. *Ontology Development Kit* (ODK) [27] uses Travis CI to run workflows with ROBOT. CI/CD pipelines are reported for the publication of different ontologies, such as the Financial Industry Business Ontology (FIBO) in [1], the International Data Spaces Information Model (IDSA) in [3], and the CASE Cyber Ontology[8]. Specific GitHub actions are available on the GitHub marketplace for running RD-

[4] `https://marketplace.visualstudio.com/items?itemName=takemikami.vscode-rdflint`
[5] `https://jena.apache.org/documentation/tools/index.html`
[6] `https://obofoundry.org/`
[7] `https://docs.github.com/en/developers/apps`
[8] `https://github.com/marketplace/actions/case-ontology-validator`

FLint[9], validating RDF syntaxes[10,11], or validating RDF files against SHACL shapes[12] or ShEx [31].

3 SAREF - A Modular and Versioned Suite of Ontologies

As illustrated in Figure 1, the SAREF suite of ontologies is composed of ontologies that define generic patterns such as SAREF4SYST [12], a core ontology SAREF Core [14] illustrated in Figure 2, and different extensions developed for distinct vertical domains: SAREF4ENER for energy [6], SAREF4ENVI for environment [8], SAREF4BLDG for smart buildings [8], SAREF4CITY for smart cities, SAREF4INMA for industry and manufacturing, SAREF4AGRI for agriculture and food, SAREF4AUTO for automotive [9], SAREF4EHAW for e-health and ageing well [10], SAREF4WEAR for wearables [11], SAREF4WATR for water management [7], and SAREF4LIFT for smart lifts [15].

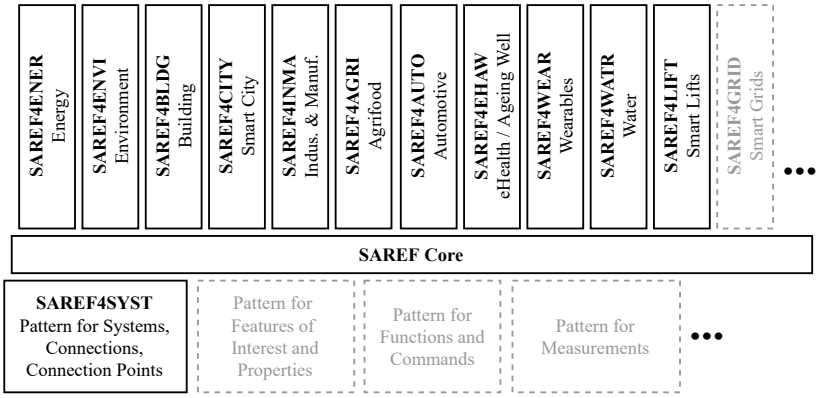

Fig. 1: The SAREF suite of ontologies with its different modules. Dashed blocks are under development in ongoing projects.

SAREF Projects are formally endorsed by ETSI as Technical Specification documents, and therefore adopt the *semantic versioning* approach as specified by the ETSI version numbering system for documents.[13] Each SAREF project version is tagged with three numbers: a *major*, a *technical*, and a *non-technical*. A major increment indicates a break in backward compatibility. A technical increment indicates technical changes. A non-technical increment indicates an editorial change.

[9] https://github.com/marketplace/actions/setup-rdflint
[10] https://github.com/marketplace/actions/rdf-syntax-check
[11] https://github.com/marketplace/actions/validate-rdf-with-jena
[12] https://github.com/marketplace/actions/validate-shacl
[13] https://portal.etsi.org/Services/editHelp/How-to-start/Document-procedures-and-types/Version-numbering-system

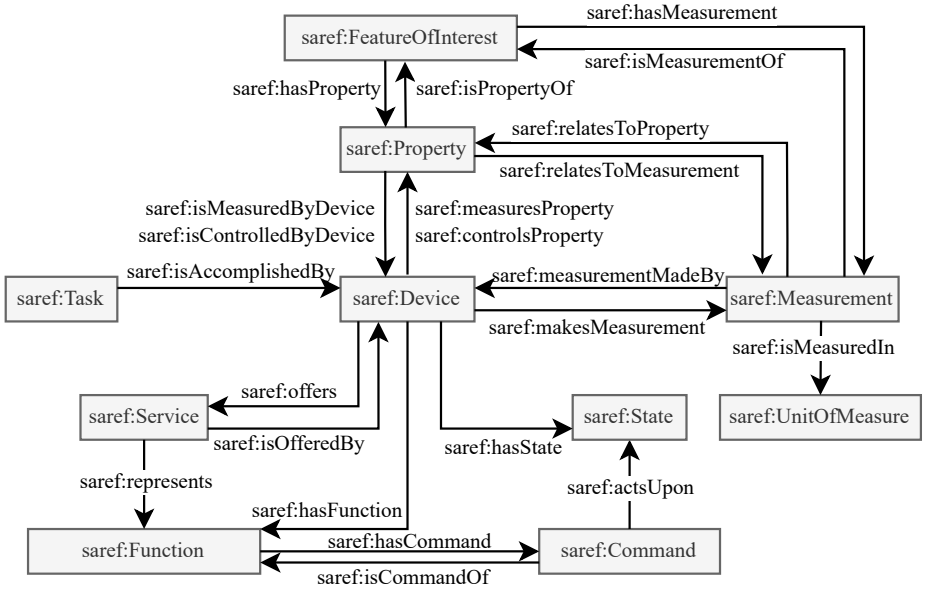

Fig. 2: An overview of the SAREF Core ontology V3.1.1 (adapted from [14])

4 The SAREF Development Framework and Workflow

4.1 The SAREF Development Workflow

In general, the development of SAREF ontologies follows the Linked Open Terms (LOT) methodology [29], which adopts a V-model approach with conditional jumps to previous development stages. The following roles are defined in [13, Clause 5]:

Steering Board member: steers the development of SAREF extensions, community participation, and underlying infrastructure.
Technical Board member: maintains the SAREF public forge and portal.
Project Leader: performs SAREF project management tasks.
Ontology Developer: uses their in-depth understanding of ontology development to interactively modify the ontology during the development cycle. Ontology developers create and modify the different development artefacts, provide new requirements to the ontology, validate whether they are satisfied or not when implemented, and make decisions regarding what contributions are included in the ontology.
Contributor: proposes contributions to the ontology given their domain-specific knowledge.
Ontology User: is interested in any of the SAREF projects or in proposing a new project.

Different workflows are established:

Workflow 1: new SAREF project versions [13, Clause 6.1].
Workflow 2: project version development [13, Clause 7.1].
Workflow 3: project release (publication) [13, Clause 8.1].

Workflow 1 applies to new versions of the SAREF Core, new versions of existing SAREF extensions, or initial versions (V1.1.1) of new SAREF extensions. For example, Figure 3 illustrates the workflow for the development of different project versions from the SAREF community of users. The project version development workflow is founded upon the issues recorded in the corresponding SAREF project issue tracker on the public ETSI Forge portal. The issue tracker not only presents a single point of development interaction, but also tracks development activity and discussions. Any update to a SAREF project version is made through a change request that is posted as an issue in the corresponding git repository of the public forge, where it is assigned an issue number. Issues include change requests related to new ontology requirements, defects, or improvements in the ontology specification, in the ontology tests and examples, or in the ontology documentation. Any contributor can create a new change request or review and discuss existing change requests. Steering Board members then review these change requests. Ontology developers also review change requests, propose, and review implementations of accepted change requests. The project leader ensures that the change requests are approved by SmartM2M and that the implementations of the change requests satisfy the requested change.

4.2 SAREF Project Repositories on the ETSI Forge

SAREF Projects may correspond to SAREF Core or a SAREF Extension. Each SAREF Extension is assigned an identifier that is based on a four letter code. SAREF Projects are hosted in a git repository on the public ETSI Forge `https://saref.etsi.org/sources/`. *Release branches* are used instead of tags to identify releases, thus allowing continuous evolution of the documentation or examples after the ontology version is published. SAREF project repositories therefore have four different types of branches:

- `issue-x` branches to work on an issue,
- `develop-vx.y.z` branches to work on a version,
- `prerelease-vx.y.z` branches to work on the final validation of the ontology,
- `release-vx.y.z` branches for published versions.

Protection rules are defined to prevent ontology developers from directly pushing their changes to `development-vx.y.z` branches or from directly accepting merge requests in `prerelease-vx.y.z` branches. In addition, the `saref-portal` repository[14] contains the static resources of the SAREF public documentation portal, and a file `.saref-repositories.yml` that references each of the SAREF projects whose documentation needs to be generated on the portal.

[14] `https://saref.etsi.org/sources/saref-portal`

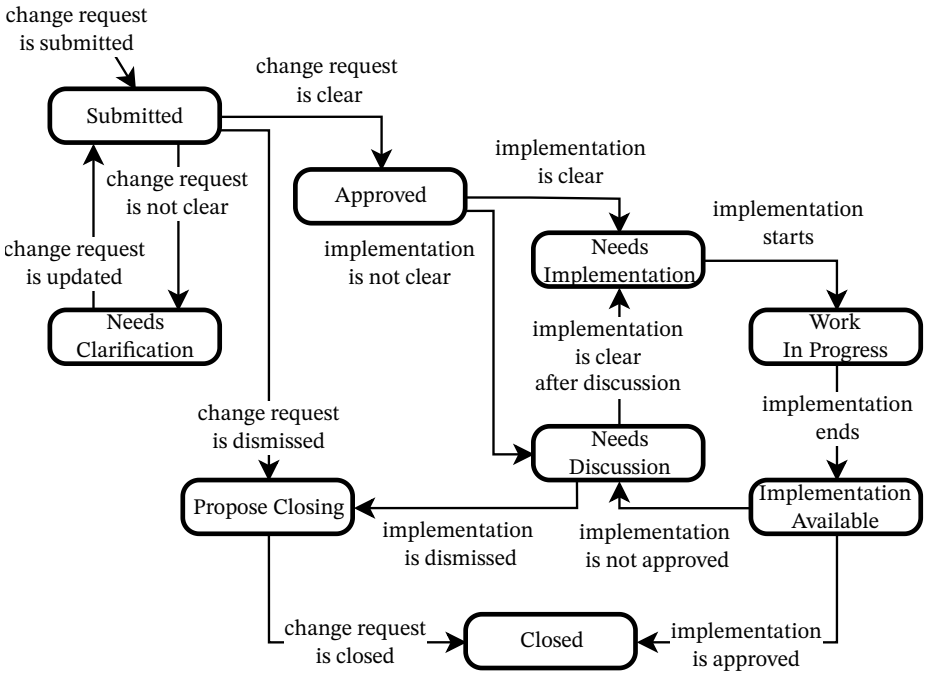

Fig. 3: The SAREF project version development workflow (adapted from [13])

4.3 SAREF Project Version Specification and Documentation

Clause 9 in the ETSI TS 103 673 technical specification [13, Clause 9] outlines the rules to which each SAREF project repository must comply. All these rules, summarized below, are automatically checked by the SAREF Pipeline.

Clause 9.2. The SAREF project version directory shall[15] contain a `README.md`, a `LICENSE` file, and folders `requirements`, `ontology`, `tests`, `examples`, `documentation`.

Clauses 9.3.1. and 9.3.2. The `requirements` directory should contain a file `requirements.csv`. If present, this file shall be a UTF-8 encoded CSV file with specific delimiter, quote character, and specific header row.

Clause 9.4.1. The `ontology` directory shall contain the ontology document of the project version `saref.ttl` for SAREF Core, `saref4abcd.ttl` for SAREF extension SAREF4ABCD. This file shall contain the sources of a consistent OWL2 DL ontology in the Turtle 1.1 format.

Clause 9.4.2. If the document contains a base or prefix declarations, they shall conform to the ones given in this clause.

Clause 9.4.3.1. specifies which ontology series IRI and which ontology version IRI shall be used, along with version-related metadata such as owl:versionInfo,

[15] ETSI draft editing rules disallow the use of the MUST keyword. The highest level of specification enforcement is SHALL.

vann:preferredNamespacePrefix, vann:preferredNamespaceUri, and owl:priorVersion if applicable.

Clause 9.4.3.2. specifies which ontology metadata shall, should, or may be defined, and which value and/or datatype they shall have if they are defined. In general the use of Dublin Core terms is enforced. Ontology imports are often a source of versioning problems. This clause stipulates that supporting ontologies are imported by their version IRI and not by the ontology series identifier. For example, SAREF4LIFT V1.1.1 imports SAREF Core V3.1.1, SAREF4SYST V1.1.2, and SAREF4BLDG V1.1.2. This rule effectively avoids imported ontology versioning issues.

Clause 9.4.3.3. defines who can be considered a creator or a contributor to the SAREF project version, and how they may be described in the ontology. In general persons are described using IRIs or blank nodes, which then shall be instances of schema:Person and further described using a schema:givenName and schema:familyName. Affiliations shall be described as instances of schema:Organization.

Clause 9.4.4.1. defines which namespace shall be used for terms defined in the ontology document, and which naming convention shall be used for classes and properties.

Clause 9.4.4.2. enforces the use of rdfs:label and rdfs:comment metadata on terms, which should at least have one rdf:langString datatype with the `en` language tag.

Clause 9.4.5. specifies that the ontology shall satisfy the OWL2 DL profile, with the exception that unknown datatypes may be used. It shall be consistent, it should not present ontology development pitfalls as per the the OntOlogy Pitfall Scanner! (OOPS!) [30], and every declared class should be satisfiable.

Clause 9.5.1. and 9.5.2. The `tests` directory should contain a file `tests.csv`. If present, this file shall be a UTF-8 encoded CSV file with specific delimiter, quote character, and given header row.

Clause 9.6.1. The `examples` directory should contain example documents that illustrate how the ontology can be used in practice. Every example document shall be a consistent OWL2 DL ontology in the Turtle 1.1 format. Main classes and properties should be illustrated with at least one example.

Clause 9.6.2. Specifies allowed prefixes and base declarations in examples.

Clause 9.6.3. The example document shall be declared as a dctype:Dataset. It shall be asserted to conform to (dct:conformsTo) the SAREF project version IRI. It additionally may conform to other SAREF project specific versions, or some ontology published by international Standard Development Organizations. Additional metadata elements shall be used.

Clause 9.6.4. The RDF graph in the example document, when augmented with an ontology declaration that imports all the ontologies the example conforms to, shall satisfy the OWL2 DL profile, with the exception that unknown datatypes may be used, and shall be consistent.

Clause 9.7.1. The `documentation` directory should contain documentation sources to provide human-readable documentation for the ontology and how it can be

used in practice. These documentation sources are for `creators`, `contributors`, `abstract`, `description`, `examples`, `references`, `acknowledgements`. They shall be a HTML snippet and have the extension .html or be a markdown snippet and have extension .md.

Clause 9.7.2. Diagrams should be included in a directory `documentation/diagrams`, and adopt graphical notations from [18].

5 Quality Control and Requirements Verification with the SAREF Pipeline

The SAREF Pipeline software verifies the conformance to the specification summarized in Section 4.3, and generates the HTML documentation of SAREF. It can be run on a SAREF project repository, or on the `saref-portal` repository for verifying all the SAREF projects and generating the complete documentation of SAREF to be deployed on the public SAREF community portal. This section provides technical details about this software, how the checks are performed, and how it is used.

5.1 User Interface, Execution Modes, Error Reporting

The SAREF Pipeline operates in either a graphical mode (Figure 4) or a command line interface (Figure 5) depending upon the ontology developer's preference. Choosing an *execution mode* determines the thoroughness and coverage of the pipeline tests and usually depends on the development stage:

develop some metadata such as version numbers are not checked;
release thorough check of the repository which must be clean (contain no tracked and modified files);
portal pre-release run on the `saref-portal` repository to operate a strict check and generate the documentation for all pre-release and release branches of the source repositories;
portal release same as above, but only considers release branches.

The ontology engineer may also choose to skip some tasks such as the analysis of examples, the generation of the HTML documentation for terms, or the generation of the HTML documentation altogether.

The SAREF Pipeline provides a detailed but easy-to-read global view of all the identified warnings and errors. Violation of *shall* clauses trigger errors, while violation of *should* clauses trigger warnings. The logs are formatted in HTML in the GUI and markdown in the CLI, which is convenient to create nicely-formatted issues rapidly to collaboratively deal with problems.[16] In addition, a log file in the JUnit report format is generated, which can be used by GitLab to provide an overview of the issues in Merge Requests.[17]

[16] example: `https://labs.etsi.org/rep/saref/saref4agri/-/issues/1`
[17] Test reports in Merge Request on GitLab: `https://docs.gitlab.com/ee/ci/testing/unit_test_reports.html`

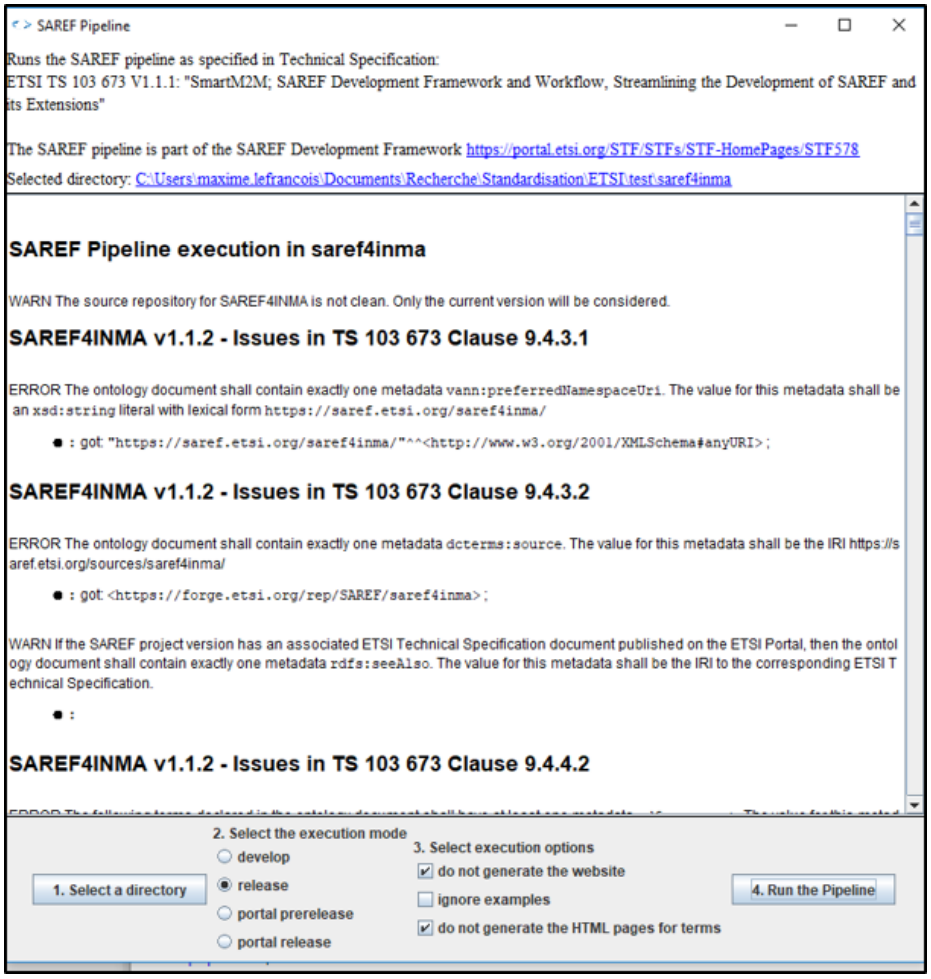

Fig. 4: Graphical User Interface (GUI) of the SAREF Pipeline `https://saref.etsi.org/sources/saref-pipeline/`

```
saref-pipeline/target$ java -jar saref-pipeline.jar
usage: java -jar saref-pipeline.jar <mode> [<options>] [<target>]

Runs the SAREF pipeline as specified in Technical Specification:
  ETSI TS 103 673 V1.1.1: "SmartM2M; SAREF Development Framework and
            Workflow, Streamlining the Development of SAREF and its
            Extensions"

<mode> can take the following values:
  develop   Run the SAREF pipeline in relax mode in the target SAREF
            project
  release   Run the SAREF pipeline in strict mode in the target SAREF
            project
  prerelease-portal    Operate a strict check and generate the portal for
            pre-release or release branches of each source in the
            configuration file `.saref-repositories.yml`
  release-portal    Operate a strict check generate the portal for release
            branches of each source in the configuration file
            `.saref-repositories.yml`
  clean     Remove all files generated by the previous executions
  help      Displays this message

<target> points to the target directory. By default, target is the current
            directory.

<options> can take the following values:
 -e,--no-examples   Do not check examples
 -s,--no-site       Do not generate the static portal
 -t,--no-terms      Do not generate the static portal for terms
```

Fig. 5: Command Line Interface (CLI) of the SAREF Pipeline `https://saref.etsi.org/sources/saref-pipeline/`

5.2 SAREF Projects Dependency Management using Git

Since interdependent SAREF projects are often developed in parallel, dependencies to other SAREF projects may be declared in a `.saref-repositories.yml` file at the root of the repository. The SAREF Pipeline parses this file and clones and/or pulls the latest changes of those repositories in a temporary folder `sources`. It prompts the user for credentials for non-public repositories.

If the ontology document or if an example imports one of these SAREF ontologies in a specific version, then this folder is checked out at the corresponding branch (release, prerelease, or develop branch) before the ontology file is loaded.

Other checks use git operations. For example in Clause 9.4.3.1 the SAREF Pipeline computes the ontology version IRI from the version number in the branch name (not possible if on a `issue-x` branch), and the desired value for owl:priorVersion based on the list of `release-x.y.z` branches.

5.3 Testing Requirement Satisfaction with Themis

Requirements (`requirements.csv` file) and tests (`tests.csv` file) are sent to the Themis service [16] which automatically generates OWL Axioms based on Lexico-Syntactic patterns analysis of the tests, and checks that the ontology contains the generated axioms.

5.4 Checks based on SHACL

SHACL shapes [23] are used for verifying clauses 9.4.3.1 (ontology IRI and ontology version IRI), 9.4.3.2 (ontology metadata), 9.4.3.3 (creators and contributors), 9.4.4.2 (ontology terms metadata), and 9.6.3. (example declaration). The general process is as follows:

Step 1: The SAREF Pipeline loads the RDF graph containing SHACL shape stubs[18] for the clause being verified;
Step 2: The RDF graph containing the SHACL shapes is updated with information that is computed on-the-fly;
Step 3: The RDF graph of the ontology or the example being tested is evaluated against the SHACL shapes;
Step 4: The conformance result (a RDF Graph) is queried for warnings and errors, which are logged.

Updates in Step 2 include:

- adding localized violation messages (all clauses);
- computing the expected ontology series IRI, objects for the owl:versionIRI, owl:versionInfo and owl:priorVersion annotations, and preferred prefix and namespace metadata (Clause 9.4.3.1);
- removing some metadata checks when running in **develop** (relaxed) mode (Clause 9.4.3.2);
- expliciting the target nodes for shapes that verify metadata for terms: only those terms that have the namespace of the ontology are required to have a label and a comment in that ontology (Clause 9.4.4.2);
- computing to which ontology the example shall conform to (Clause 9.6.3).

5.5 Checks using OWL API and Hermit

Checking Clauses 9.4.5 and 9.6.4 involves the use of the OWL API library [21]. Violations to the OWL2 DL profile are logged, except those related to undefined datatypes[19]. Then the consistency of the ontology or example is verified using Hermit [19], and optionally a maximum of 10 inconsistency explanations are logged. Finally, the coherence of each class in the signature except owl:Nothing is checked, and optionally a maximum of 10 incoherence explanations are logged.

As the SAREF Pipeline generally relies on Apache Jena and Git for dependency management, a specific `OntologyParser` realization class was developed such that OWL API can consume Jena RDF Models, and potentially checkout repositories in specific branches while resolving imports.

Before checking examples, the dctype:Dataset and dct:conformsTo declarations are replaced by owl:Ontology and owl:imports, respectively. This way, OWL

[18] Stubs for SHACL shapes are available at `https://saref.etsi.org/sources/saref-pipeline/-/tree/master/src/main/resources/shapes`
[19] Violations of type `UseOfDefinedDatatypeInDatatypeRestriction`, `UseOfDefinedDatatypeInLiteral`, `UseOfUndeclaredDatatype`, and `UseOfUnknownDatatype`

API can resolve imports and check the consistency of the example plus the ontologies it conforms to. In addition, all the classes in the signature of the example and the ontologies it conforms to are checked for coherence.

5.6 Documentation generation

The last phase in the SAREF Pipeline execution is the generation of the documentation in a folder `site`, that is ultimately served on the SAREF Portal `https://saref.etsi.org/` using an Apache 2 web server. Documentation is generated for ontology versions, examples, and terms, and an `.htaccess` document is generated to implement content negotiation and redirection from every ontology IRI to its most recent version IRI.

For ontology versions and examples, the source Turtle file is first copied to an appropriate directory. Then different other RDF serializations are generated in RDF/XML, N-triples, N3, and JSON-LD. Finally, an HTML documentation is generated. See for example `https://saref.etsi.org/core/v3.1.1/`. For each SAREF term `t`, an RDF Graph is created with the result of a SPARQL DESCRIBE `t` query evaluated on the last version of the ontology that defines this term. This graph is augmented with rdfs:isDefinedBy metadata that point to each ontology version IRI that defines this term, and custom ex:usedBy metadata that point to each ontology version IRI that uses this term. For terms also different RDF serializations are generated and an HTML documentation is generated. See for example `https://saref.etsi.org/core/Command`

The HTML documentation generation process is inspired by LODE [28], but developed as a set of 61 named TEMPLATE, SELECT and FUNCTION SPARQL-Generate queries,[20] which are executed using version 2.0 of the SPARQL-Generate engine [24,26].[21] This transformation is executed on the RDF Graph of the ontology augmented with the RDF Graph of imported ontology versions, explicit entity declarations, and annotations with literal values (especially labels and comments) for all the entities that are used but not declared in this ontology. This allow the SPARQL-Generate transformations to generate class and property hierarchies, and add tooltips for all terms with the appropriate label and comment. It also incorporates the documentation snippets from the `documentation` folder, optionally converting markdown snippets to HTML if required.

The documentation generation tool does not yet generate multilingual documentation, although initial efforts towards internationalization has been done.[22]

[20] Available in the `main/resources/documentation` folder of the `saref-pipeline`

[21] SPARQL-Generate 2.0 with GENERATE, TEMPLATE, SELECT, and FUNCTION queries has no dedicated publication yet. More information can be obtained from the website and presentation at `https://ci.mines-stetienne.fr/sparql-generate/`, and `https://www.slideshare.net/maximelefrancois86/overview-of-the-sparqlgenerate-language-and-latest-developments`

[22] See `https://labs.etsi.org/rep/saref/saref-pipeline/-/blob/master/src/main/resources/documentation/en.properties`

5.7 Continuous Integration and Delivery

The SAREF Pipeline is configured to run using the *Continuous Integration and Continuous Delivery* (CI/CD) features of the ETSI Forge, which is an instance of GitLab. Each SAREF project repository is configured such that the SAREF Pipeline is run in different modes and with different options depending on the type of branch where a commit is pushed:

issue and develop branches run `java -jar saref-pipeline.jar develop -s` (relaxed mode without generating the static portal), then publish an HTML report file to the snapshot area of the SAREF portal (deleted after one day) at `https://saref.etsi.org/snapshot/$CI_PIPELINE_ID/report.html`, even if the pipeline identified errors;

pre-release branches run `java -jar saref-pipeline.jar release -t` (do not generate the static portal for terms) then publish the generated report and partial static portal to the snapshot area of the SAREF portal at `https://saref.etsi.org/snapshot/$CI_PIPELINE_ID/report.html`, even if the pipeline identified errors;

release branches run `java -jar saref-pipeline.jar release -t` and, if successful, trigger the CI/CD pipeline on the `master` branch of `saref-portal`

The CI/CD process on the `master` branch of the `saref-portal` project is a three stage process:

Step 1: run `java -jar saref-pipeline.jar release-portal`,

Step 2: publish the generated report and static portal to the staging area of the SAREF portal (deleted after one week) at `https://saref.etsi.org/staging/$CI_PIPELINE_ID/`,

Step 3: if the portal in the staging area seems fine, this step can be manually triggered to publish the generated static portal to `https://saref.etsi.org/`.

6 Discussion and Current Work

Compared to the existing approaches listed in Section 2, the SAREF Pipeline is developed for a modular and versioned suite of ontologies, each having its sources in a separate git repository. It is usable both as GUI, CLI, and in CI/CD pipelines, and integrates many different semantic web technologies such as Apache Jena, OWL API, the Hermit reasoner, OOPS!, SHACL, Themis, SPARQL-Generate. It produces detailed error reporting, and generates a public documentation portal for the whole suite of SAREF ontologies and the terms they define. The SAREF Pipeline is publicly available under the open-source ETSI Forge license (BSD-3-Clause LICENSE) at the stable URL `https://saref.etsi.org/sources/saref-pipeline`. It is further registered on Zenodo with DOI 10.5281/zenodo.7913535 and has therefore an associated canonical citation. Its documentation, in terms of the list of checks it operates, is part

of the ETSI TS 103 673 technical specification which benefits from the ETSI standardization principles in terms of openness, maintenance, availability, and stability.[23] The SAREF Pipeline already had an impact in raising the quality level of all the SAREF ontologies, and publishing the public SAREF community portal `https://saref.etsi.org/`. Although originally tailored for the SAREF ontology development framework, it is currently used by projects outside ETSI to prepare new candidate extensions to SAREF. For example the newly created ETSI work item for SAREF4ENER V1.2.1, is based on contributions from the H2020 InterConnect project.[24]

Extensions to ETSI TS 103 673 and the SAREF Pipeline are planned in the context of the ongoing ETSI STF 653[25], which aims at consolidating the suite of SAREF ontologies into a more homogeneous and predictable structure, using operationalized ontology patterns. The initial observation that motivated STF 653 is that, even within the SAREF developer community, modeling and naming choices are sometimes varied and conflicting. The primary technique to resolve modeling discrepancies was to use *namespaces* to label the concepts and terms in each module. Choosing a distinct namespace for each module makes it easy to identify from which module a term originates. However, this approach poses several problems. First, it is sometimes difficult as a user of these ontologies to remember what the namespace is for each term. We have, for example, a variety of subclasses of saref:Property spread across the namespaces of the different extensions, depending on when they were first defined: saref:Temperature, saref:Humidity, saref:Power, s4ener:PowerMax, s4ener:PowerStandardDeviation, s4inma:Size, saref:Light, s4envi:LightProperty. Second, experience shows that it is sometimes necessary to move a term from one module to another. For example SAREF4CITY V1.1.1 introduced the concept of s4city:FeatureOfInterest, and it was decided during the development of SAREF Core V3.1.1 that this concept would be moved into the core ontology. So it is now identified by saref:FeatureOfInterest, and the SAREF4CITY implementations had to be modified. This problem would not have arisen if an approach based on a unique namespace had been adopted. Also, some SAREF developers decided to extend saref:Property not using classes, but using instances. Example include s4envi:Frequency, s4wear:SoundLevel, s4wear:BatteryRemainingTime, s4watr:Conductivity, s4wear:Temperature. Ongoing work in STF 653 therefore aims at defining ontology patterns to guide the development of SAREF extensions, and operationalizing them in the SAREF Pipeline.

[23] `https://www.etsi.org/standards/standards-making`
[24] `https://portal.etsi.org/webapp/WorkProgram/Report_WorkItem.asp?WKI_ID=68491`
[25] `https://portal.etsi.org/xtfs/#/xTF/653`

7 Conclusion

The SAREF Pipeline integrates the SAREF development and documentation generation pipeline, as part of the SAREF Development Framework. It automatically checks the conformance of SAREF project repositories to the technical specification ETSI TS 103 673. It has been applied on all the SAREF ontologies, and was used to generate the public SAREF community portal `https://saref.etsi.org/`. It is made available as open source under an open license to the semantic web community for the development of new SAREF extension, and could be adapted for other ontology development projects. ETSI TS 103 673 and the SAREF pipeline speed up the evolution of the current and future extensions of SAREF, and help reducing the costs of developing these extensions.

To summarize its main benefits, the SAREF Pipeline provides an efficient and robust support infrastructure for the continuous integration and continuous delivery of the modular and versioned suite of SAREF ontologies, which cover multiple application domains of the IoT. In particular, the SAREF Pipeline:

- Operates on a modular and versioned suite of ontologies;
- Operationalizes conformance checking against ETSI TS 103 673;
- Combines several semantic web technologies to automate checks;
- Is used by ontology developers as a GUI, and by CI/CD pipelines as CLI;
- Produces detailed error reporting, and generates documentation;
- Has been used to verify all SAREF ontologies, and is used to shape new SAREF extensions.

Acknowledgements

The development of the SAREF Pipeline has been funded by ETSI STF 578. The authors thank the other experts involved in this project, who contributed to the definition of the SAREF development workflow detailed in Section 4.1, and to online web services OOPS! and Themis that are called by the SAREF Pipeline. Ongoing development is funded by ETSI STF 653.

Resource Availability Statement: Source code of the SAREF Pipeline is available from the ETSI Forge.[26] It is also registered on Zenodo with DOI 10.5281/zenodo.7913534.[27] The canonical citation is [25].

[26] `https://saref.etsi.org/sources/saref-pipeline`
[27] `https://doi.org/10.5281/zenodo.7913534`

References

1. Allemang, D., Garbacz, P., Gradzki, P., Kendall, E., Trypuz, R.: An infrastructure for collaborative ontology development, lessons learned from developing the financial industry business ontology (FIBO). In: Formal Ontology in Information Systems. IOS Press (2022)
2. Alobaid, A., Garijo, D., Poveda-Villalón, M., Santana-Perez, I., Fernández-Izquierdo, A., Corcho, O.: Automating ontology engineering support activities with ontoology. Journal of Web Semantics **57**, 100472 (2019)
3. Bader, S., Pullmann, J., Mader, C., Tramp, S., Quix, C., Müller, A.W., Akyürek, H., Böckmann, M., Imbusch, B.T., Lipp, J., et al.: The international data spaces information model–an ontology for sovereign exchange of digital content. In: The Semantic Web–ISWC 2020: 19th International Semantic Web Conference, Athens, Greece, November 2–6, 2020, Proceedings, Part II. pp. 176–192. Springer (2020)
4. Crystal-Ornelas, R., Varadharajan, C., Bond-Lamberty, B., Boye, K., Burrus, M., Cholia, S., Crow, M., Damerow, J., Devarakonda, R., Ely, K.S., et al.: A guide to using github for developing and versioning data standards and reporting formats. Earth and Space Science **8**(8) (2021)
5. Daniele, L., den Hartog, F., Roes, J.: Created in Close Interaction with the Industry: The Smart Appliances REFerence (SAREF) Ontology. In: Proceedings of the Formal Ontologies Meet Industry workshop (FOMI 2015), Vol. Lecture Notes in Business Information Processing. vol. 225, pp. 100–112. Springer, Cham (2015)
6. ETSI: SmartM2M; Extension to SAREF; Part 1: Energy Domain. ETSI Technical Specification 103 410-1 V1.1.2. (05 2020)
7. ETSI: SmartM2M; Extension to SAREF; Part 10: Water Domain. ETSI Technical Specification 103 410-10 V1.1.1. (07 2020)
8. ETSI: SmartM2M; Extension to SAREF; Part 2: Environment Domain. ETSI Technical Specification 103 410-2 V1.1.2. (05 2020)
9. ETSI: SmartM2M; Extension to SAREF; Part 7: Automotive Domain. ETSI Technical Specification 103 410-7 V1.1.1. (07 2020)
10. ETSI: SmartM2M; Extension to SAREF; Part 8: eHealth/Ageing-well Domain. ETSI Technical Specification 103 410-8 V1.1.1. (07 2020)
11. ETSI: SmartM2M; Extension to SAREF; Part 9: Wearables Domain. ETSI Technical Specification 103 410-9 V1.1.1. (07 2020)
12. ETSI: SmartM2M; SAREF consolidation with new reference ontology patterns, based on the experience from the SEAS project. ETSI Technical Specification 103 548 V1.1.2. (06 2020)
13. ETSI: SmartM2M; SAREF Development Framework and Workflow, Streamlining the Development of SAREF and its Extensions. ETSI Technical Specification 103 673 V1.1.1. (2020), `https://www.etsi.org/deliver/etsi_ts/103600_103699/103673/01.01.01_60/ts_103673v010101p.pdf`
14. ETSI: SmartM2M; Smart Applications; Reference Ontology and oneM2M Mapping. ETSI Technical Specification 103 264 V3.1.1. (02 2020)
15. ETSI: SmartM2M; Extension to SAREF; Part 11: Lift Domain. ETSI Technical Specification 103 410-11 V1.1.1. (07 2021)
16. Fernández-Izquierdo, A., García-Castro, R.: Themis: a tool for validating ontologies through requirements. In: Software Engineering and Knowledge Engineering. pp. 573–753 (2019)
17. García-Castro, R., Lefrançois, M., Poveda-Villalón, M., Daniele, L.: The ETSI SAREF Ontology for Smart Applications: A Long Path of

Development and Evolution, pp. 183–215. Wiley-IEEE Press (2023). https://doi.org/10.1002/9781119899457.ch7
18. Garijo, D., Poveda-Villalón, M.: Best practices for implementing fair vocabularies and ontologies on the web. Applications and practices in ontology design, extraction, and reasoning **49**, 39 (2020)
19. Glimm, B., Horrocks, I., Motik, B., Stoilos, G., Wang, Z.: Hermit: an owl 2 reasoner. Journal of Automated Reasoning **53**, 245–269 (2014)
20. Halilaj, L., Petersen, N., Grangel-González, I., Lange, C., Auer, S., Coskun, G., Lohmann, S.: Vocol: An integrated environment to support version-controlled vocabulary development. In: European Knowledge Acquisition Workshop. pp. 303–319. Springer (2016)
21. Horridge, M., Bechhofer, S.: The owl api: A java api for owl ontologies. Semantic web **2**(1), 11–21 (2011)
22. Jackson, R.C., Balhoff, J.P., Douglass, E., Harris, N.L., Mungall, C.J., Overton, J.A.: Robot: a tool for automating ontology workflows. BMC bioinformatics **20**(1), 1–10 (2019)
23. Knublauch, H., Kontokostas, D.: Shapes Constraint Language (SHACL). W3C Recommendation, W3C (Jul 20 2017)
24. Lefrançois, M., Zimmermann, A., Bakerally, N.: A SPARQL extension for generating RDF from heterogeneous formats. In: European Semantic Web Conference. pp. 35–50. Springer (2017)
25. Lefrançois, M.: Saref pipeline (May 2023). https://doi.org/10.5281/zenodo.7913535, `https://doi.org/10.5281/zenodo.7913535`
26. Lefrançois, M., Noorani, B., el mehdi, K., Alqawasmeh, O., Alqawasmeh, O., Zimmermann, A., Ariff, A., Kolchin, M., Ceriani, M.: sparql-generate/sparql-generate: 2.0.12 (Oct 2022). https://doi.org/10.5281/zenodo.7141122, `https://doi.org/10.5281/zenodo.7141122`
27. Matentzoglu, N., Mungall, C., Goutte-Gattat, D.: Ontology development kit (Jul 2021). https://doi.org/10.5281/zenodo.6257507, `https://doi.org/10.5281/zenodo.6257507`, if you use this software, please cite it as below.
28. Peroni, S., Shotton, D., Vitali, F.: The live owl documentation environment: a tool for the automatic generation of ontology documentation. In: Knowledge Engineering and Knowledge Management: 18th International Conference, EKAW 2012, Galway City, Ireland, October 8-12, 2012. Proceedings 18. pp. 398–412. Springer (2012)
29. Poveda-Villalón, M., Fernández-Izquierdo, A., Fernández-López, M., García-Castro, R.: LOT: An industrial oriented ontology engineering framework. Engineering Applications of Artificial Intelligence **111** (2022)
30. Poveda-Villalón, M., Gómez-Pérez, A., Suárez-Figueroa, M.C.: OOPS! (ontology pitfall scanner!): An on-line tool for ontology evaluation. International Journal on Semantic Web and Information Systems (IJSWIS) **10**(2), 7–34 (2014)
31. Publio, G.C., Gayo, J.E.L., Colunga, G.F., Menendéz, P.: Ontolo-ci: Continuous data validation with shex. In: Proceedings of Poster and Demo Track and Workshop Track of the 18th International Conference on Semantic Systems co-located with 18th International Conference on Semantic Systems (SEMANTiCS 2022) (2022)