AN ONTOLOGY-BASED AUTOMATION FRAMEWORK FOR CONTINUOUS COMPLIANCE OF AIRBORNE SOFTWARE

Tim Schubert (presenting), Mohamad Ibrahim, Nora Breitmoser-Widdecke, Umut Durak

DASC 44th
Digital Avionics Systems Conference
Montreal, Canada - September 14-18, 2025

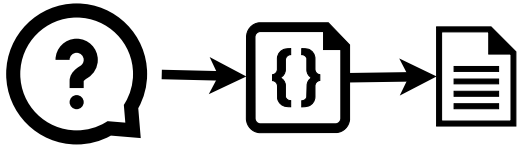
DLR

Why do we need continuous compliance?

1. Code and requirements change during development or for later SW revisions

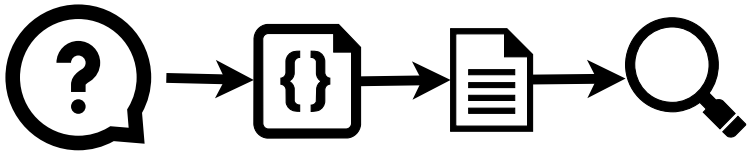
Why do we need continuous compliance?

1. Code and requirements change during development or for later SW revisions
2. Update of compliance documents and **review of compliance objectives**



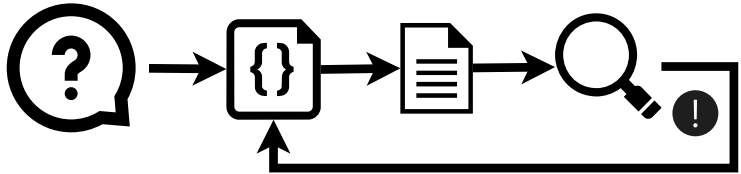
Why do we need continuous compliance?

1. Code and requirements change during development or for later SW revisions
2. Update of compliance documents and **review of compliance objectives**
3. Compliance violation is noticed *after* code changes have been made

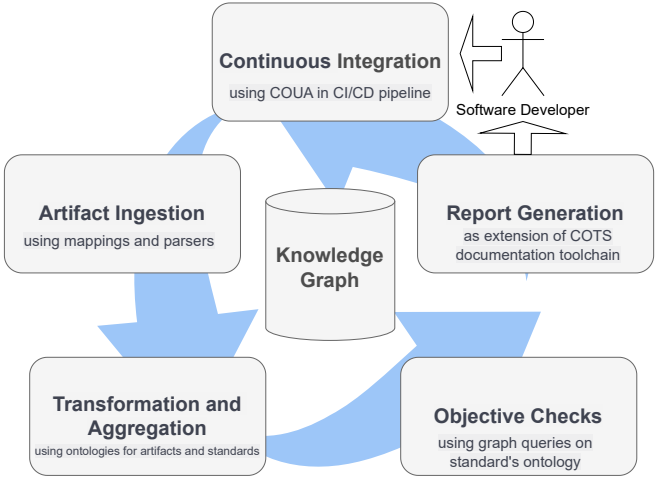


Why do we need continuous compliance?

1. Code and requirements change during development or for later SW revisions
2. Update of compliance documents and **review of compliance objectives**
3. Compliance violation is noticed *after* code changes have been made
4. **Late detection** of violations and **long and costly cycles**

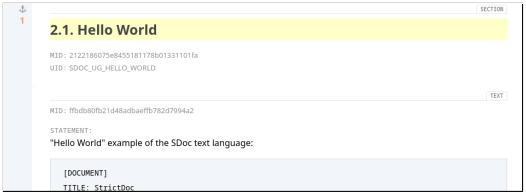


How can we apply CI to Continuous Compliance?

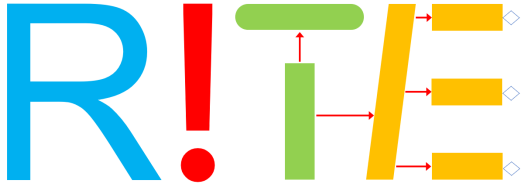


Related Work

- **StrictDoc** (document-based, tracing of source code and test data, query language)
- **RITE** / RACK (full IDE, knowledge graph, data ingestion and query languages, assurance cases, *see next presentation*)
- Various other commercial tools



Screenshot from StrictDoc documentation



We want something that can **check objectives in CI pipelines**

What are the rules for compliance?

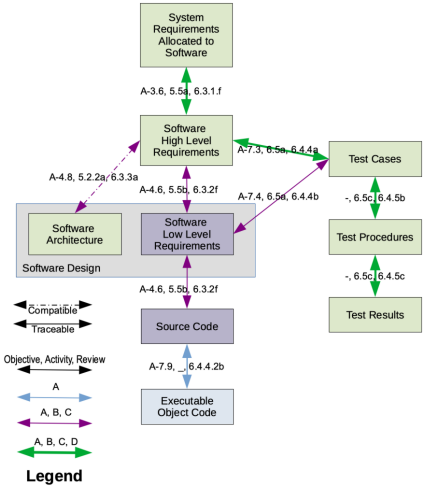
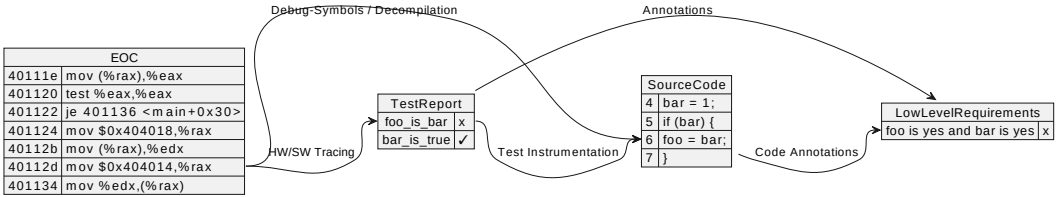


Figure by Steven H. VanderLeest - CC BY-SA 3.0

- Acceptable Means of Compliance (AMC) determine standards to be used for demonstrating compliance
- Standards (e.g. ED-12C / DO-178C) define releations between objectives, data-items, evidence, . . .
- But standards documents are not machine-readable
- Ontologies structure these semantic relations

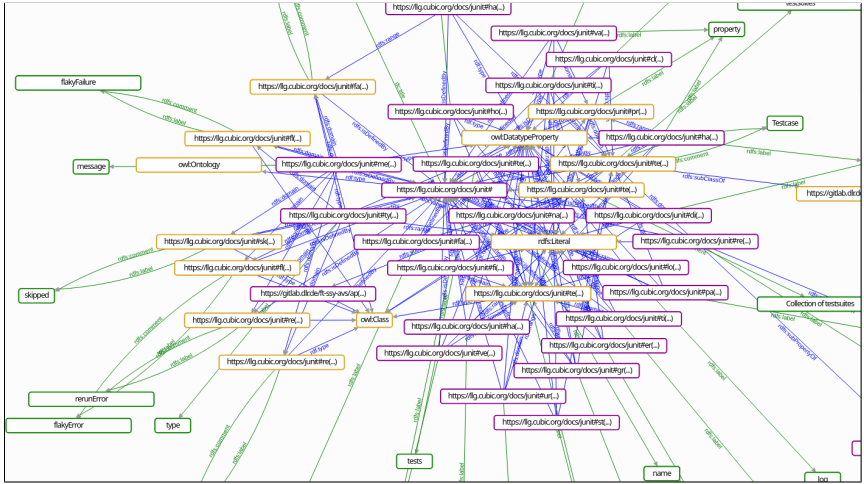
Example: How is tracing data established?

Data items can have relations to each other that we need to analyze to check compliance objectives



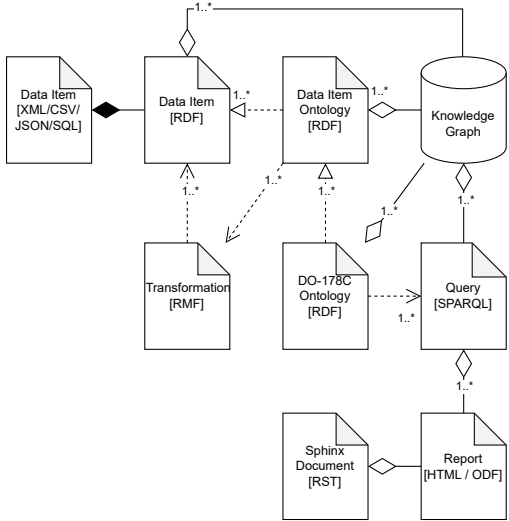
What are data item ontologies?

Data item ontologies define semantics of ingested data and usage as data items



Screenshot from JUnit ontology

What could an automated compliance framework look like?



How to combine artifacts and objectives to check compliance?

SPARQL queries match patterns on knowledge graph

Source Code is traceable to low-level requirements. — *DO-178C Objective 6.3.4.e*

```
# The following statement becomes true if the query yields no results
ask {
  filter not exists {
    # Classes of artifact ontolgies that are LL requirements
    ?rcl rdfs:subClassOf+ do178c:LowLevelRequirement .
    # Instances of these classes
    ?r a ?rcl.
    # Filter out all LL requirements that were actually traced
    minus {
      # Properties of artifact ontolgies that are requirementIds
      ?rp2 rdfs:subPropertyOf+ do178c:requirementId.
      ?r ?rp2 ?rid.
      # Trace data in artifacts
      ?tc rdfs:subClassOf+ do178c:TraceData.
      ?rp1 rdfs:subPropertyOf+ do178c:requirementId.
      # Match up traces and requirements
      ?e a ?tc ;
      ?rp1 ?rid.
    }
    # Exclude requirements which were marked as not yet implemented
    minus { ?r coua:notImplemented "true"xsd:boolean }
  }}
```

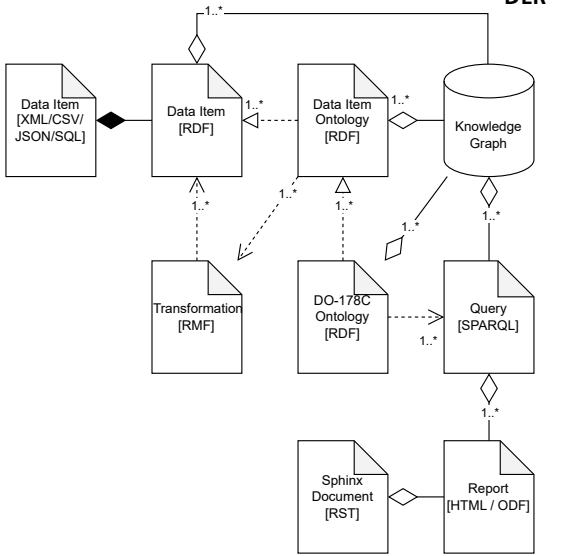
Command line interface

```
$ coua check
INFO | 2025-07-08 11:13:52,768 | https://gitlab.dlr.de/ft-ssy-avs/ap/coua/ontologies/do178c#Obj_A_2_1: YES
INFO | 2025-07-08 11:13:52,768 | https://gitlab.dlr.de/ft-ssy-avs/ap/coua/ontologies/do178c#Obj_A_2_4: YES
INFO | 2025-07-08 11:13:52,768 | https://gitlab.dlr.de/ft-ssy-avs/ap/coua/ontologies/do178c#Obj_A_3_6: YES
INFO | 2025-07-08 11:13:52,768 | https://gitlab.dlr.de/ft-ssy-avs/ap/coua/ontologies/do178c#Obj_A_4_6: YES
INFO | 2025-07-08 11:13:52,768 | https://gitlab.dlr.de/ft-ssy-avs/ap/coua/ontologies/do178c#Obj_A_5_1: YES
INFO | 2025-07-08 11:13:52,768 | https://gitlab.dlr.de/ft-ssy-avs/ap/coua/ontologies/do178c#Obj_A_5_5: YES
INFO | 2025-07-08 11:13:52,769 | https://gitlab.dlr.de/ft-ssy-avs/ap/coua/schema#CheckAskUcsCovered: YES
```

Command line interface allows to the user or the CI to check the current version of the software against supported compliance objectives

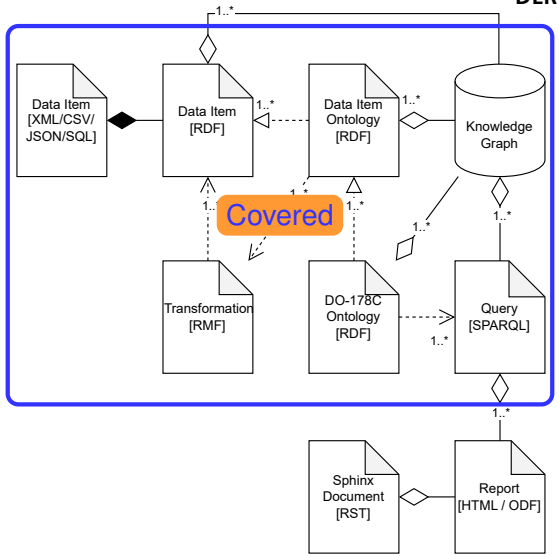
Data ingestion, abstraction, compliance objective checks

- Many **data items** can be obtained from machine-readable artifacts
- **Data item ontologies** define semantics of ingested data and usage as data items
- **Standards ontologies** define the usage of and releations between data items
- **SPARQL queries** check compliance objectives using patterns on knowledge graph



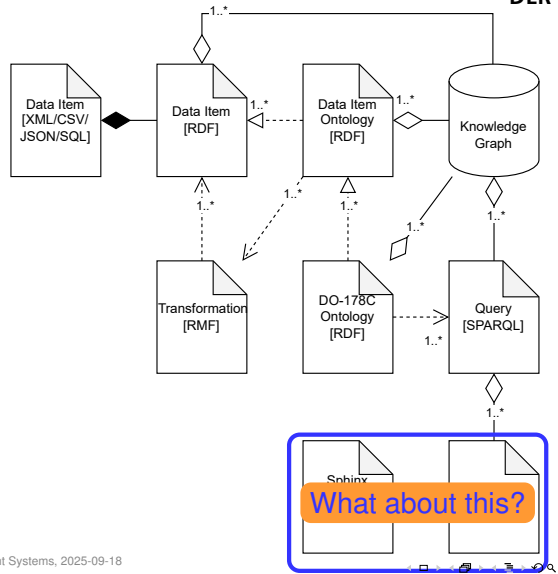
Data ingestion, abstraction, compliance objective checks

- Many **data items** can be obtained from machine-readable artifacts
- **Data item ontologies** define semantics of ingested data and usage as data items
- **Standards ontologies** define the usage of and releations between data items
- **SPARQL queries** check compliance objectives using patterns on knowledge graph



Data ingestion, abstraction, compliance objective checks

- Many **data items** can be obtained from machine-readable artifacts
- **Data item ontologies** define semantics of ingested data and usage as data items
- **Standards ontologies** define the usage of and releations between data items
- **SPARQL queries** check compliance objectives using patterns on knowledge graph



Document generation

```
Check results
-------------

This is the table showing the results for
checking each objective.

.. coua:check_list::
```

- Custom directives as part of extension for Sphinx
- Can be included anywhere in documentation
- Various formatting options (tables, paragraphs)
- Includes references between different data items

Check results

Check	Status	Description
Coua UC check	passed	All Use-Cases are covered by at least one requirement.
DO-178C A-2-1	passed	High-level requirements are developed.
DO-178C A-2-4	passed	Low-level requirements are developed.
DO-178C A-3-6	passed	High-level requirements are tracable to system requirements.
DO-178C A-4-6	passed	Low-level requirements are tracable to high-level requirements.
DO-178C A-5-1	passed	Source Code complies with low-level requirements.
DO-178C A-5-5	passed	Source Code is tracable to low-level requirements.

The following objectives require an independent review:

"DO-178C A-5-1", "DO-178C A-7-7"

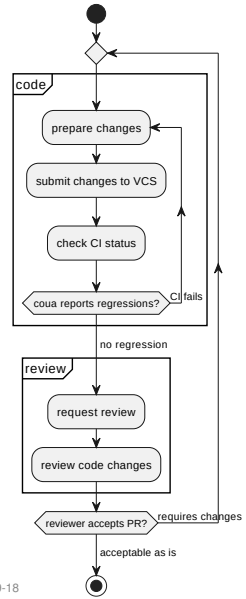
Screenshot of `https://dlr-ft.github.io/coua/main/`

How can this be integrated into CI pipelines?

Example in GitHub actions

```yaml
cert:
  runs-on: ubuntu-latest
  needs: [test]
  steps:
    - uses: actions/checkout@v4
    # Perform compliance checks
    - run: coua check
    # Build docs including check results
    - run: cd doc && make doc
    - name: Report
      uses: actions/upload-artifact@v4
      with:
        name: report
        path: doc/build/html
```

The command line interface is called from within a GitHub action



Demo of COUA

Results & Future Work

- COUA checks compliance objectives in Continuous Integration (CI) pipelines
- Approach reduces risks from late detection of compliance violations
- Future Work
  - Ontology's value and property constraints
  - One-click integration into GitHub Actions
  - Support more DO-178C objectives and COTS data formats

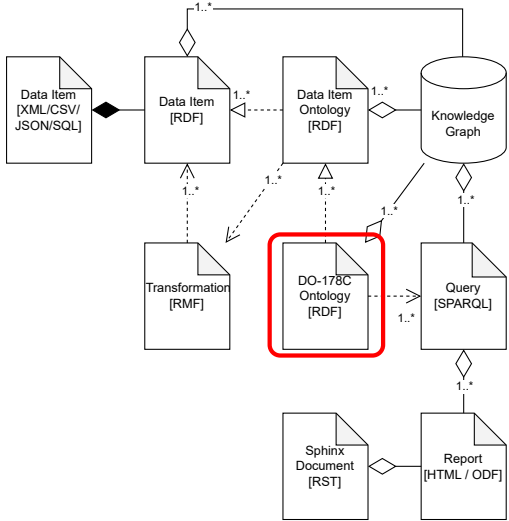
Image by Merrittimages - CC BY-SA 4.0

Gefördert durch:

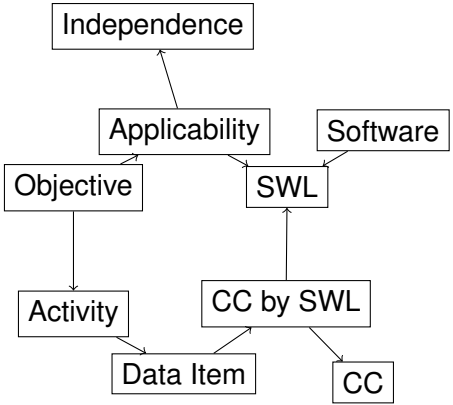
Bundesministerium
für Wirtschaft
und Energie

aufgrund eines Beschlusses
des Deutschen Bundestages

DO-178C ontology



How can DO-178C objectives be modeled as an ontology?



Relations between Objectives, Activities, Applicability, Independence, Software, Software Levels (SWL), Data Items, and Control Categories (CC) in DO-178C.

```
[...]
do178c:Software a owl:Class;
  rdfs:isDefinedBy <https://[...]/do178c#>;
  rdfs:label "DO-178C Software".

do178c:hasSoftwareLevel a owl:ObjectProperty;
  rdfs:isDefinedBy <https://[...]/do178c#>;
  rdfs:label "DO-178C Software Level of software";
  rdfs:range do178c:SoftwareLevel;
  rdfs:domain do178c:Software.

do178c:Objective a owl:Class;
  rdfs:label "Certification Objective";
  rdfs:isDefinedBy <https://[...]/do178c#>.

do178c:SoftwareLevel a owl:Class;
  rdfs:label "Software Design Assurance Level";
  rdfs:isDefinedBy <https://[...]/do178c#>.
[...]
```

Standards ontologies define the usage of and relations between data items

What are software data items?

- Requirements
- Verification records (test reports, reviews, ...)
- Verfication cases and procedures (test cases, ...)
- Documents like Plan for Software Aspects of Certification (PSAC), Software Quality Assurange Plan (SQAP), Software Development (SDP), Source Code Management System (SCM) plan, ...
- Source Code, Executable Object Code
- Trace Data

How can data be ingested in a declarative way?

Test-runner output in JUnit XML format

```
<?xml version="1.0" encoding="UTF-8"?>
<testsuites name="nextest-run" tests="5" failures="1" errors="0" uuid="[...]" time="[...]">
    <testsuite name="coua" tests="4" disabled="0" errors="0" failures="1">
        <testcase name="test_req01" classname="coua" timestamp="[...]" time="0.004">
            <failure type="test failure">Ingesting example artifact failed</failure>
            <system-out>
[...]
```

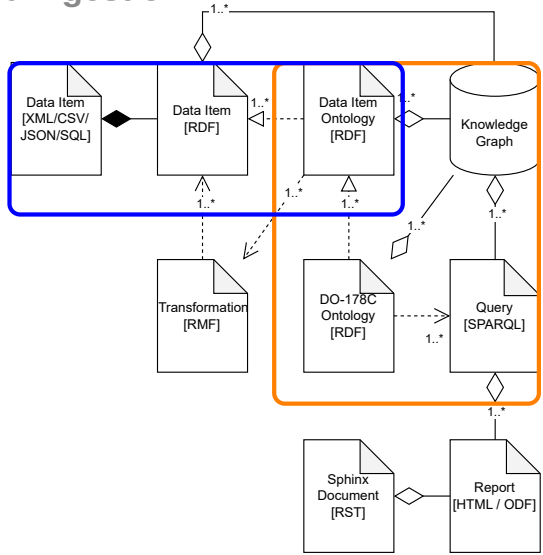
Declarative mapping specification / trivial program

```
<#TriplesMap> a rr:TriplesMap ;
  rml:logicalSource [
    rml:source "junit.xml" ;
    rml:refereceFormulation ql:XPath ;
    rml:iterator "/testsuites/testsuite/testcase"
  ] ;
  rr:subjectMap [
    rr:template "https://llg.cubic.org/docs/junit#{@name}"
  ] ;
  rr:predicateObjectMap [
    rr:predicate rdf:type; rr:objectMap [
      rr:constant coua:TestCase]].
```

Triples for knowledge graph

```
<junit:test_req01> a coua:TestCase;
    coua:failureType "test failure";
    coua:testFailureDescription
      "Ingesting example artifact failed".
```

Data retrieval and ingestion



How does COUA know which data items and standards to use?

coua.toml:

```toml
[project]
name = "Coua"

[checks]
suites = ["do178c", "coua", "cobertura"]
disabled = []

[do178c]
software = "https://dlr.de/ft/ssy/coua"
software_level = "A"

[artifacts."requirements.xml"]
type = "rdf"

[artifacts.Junit.morph]
file_path = "junit.xml"

[artifacts."coverage.xml"]
type = "cobertura"

[artifacts."traces.json"]
type = "malkoha"
```

objectives-for-software.rq

```
select ?Objective ?swl {
  # SW and its SWL are declared configuration file
  ?sw a do178c:Software ;
  do178c:hasSoftwareLevel ?swl .
  ?appl do178c:toSoftwareLevel ?swl .
  # Check if the objective is applicable to the SWL
  ?Objective a do178c:Objective ;
  do178c:hasApplicability ?appl .
}
```

The **manifest** file defines which data items / artifacts exist and which standards the software should comply with

Imprint

Topic:	**An Ontology-based Automation Framework for Continuous Compliance of Airborne Software** `https://elib.dlr.de/215567/`
Date:	2025-09-18
Author:	Tim Schubert (presenting), Mohamad Ibrahim, Nora Breitmoser-Widdecke, Umut Durak
Institute:	Institute of Flight Systems
Credits:	