# Using Ontologies in Failure Analysis

***Anna Safont-Andreu***
*Infineon AT,Villach, Austria*

***Christian Burmer***
*Infineon AG, Neubiberg, Germany*
*christian.burmer@infineon.com*

***Konstantin Schekotihin***
*Universität Klagenfurt, Austria*
*konstantin.schekotihin@aau.at*

## Abstract

Fault analysis is a complex task that requires electrical engineers to perform various analyses to detect and localize a physical defect. The analysis process is very knowledge-intensive and must be precisely documented to report the issue to customers as well as to ensure the best possible reuse of the acquired experience in similar future analyses. However, writing unambiguous documentation can be complicated for many reasons, such as selecting details and results to be presented in a report, or the naming of terms and their definition. To avoid some of these issues, FA engineers must agree on a clearly defined terminology specifying methods, physical faults and their electrical signatures, tools, and relations between them. Moreover, to allow FA software systems to use this terminology, it must be stored in a format that can be interpreted similarly by both engineers and software.

This paper presents an approach that solves these challenges by using an ontology describing FA-relevant terminology using a logic-based representation. The latter guarantees the same interpretation of the defined terms by engineers and software systems, which can use it to perform various tasks like text classification, information retrieval, or workflow verification.

## Introduction

Many FA labs have software systems, like databases or wikis, supporting the creation of reports describing findings of different fault analyses. However, these systems often store these reports as free texts and, therefore, they might be ambiguous and cannot be processed by the software automatically. To avoid these issues, one needs to provide standard definitions of terms used in the reports that all engineers agree upon. Moreover, these definitions must be stored in a format interpreted by engineers and software in the same way.

One possible solution to this challenge is to formalize the knowledge about the FA domain as an ontology. The latter is a form of a knowledge base specifically designed to store terminological definitions. The languages used to define ontologies are versatile and allow for the specification of various knowledge about the target domain. The most commonly used one is Web Ontology Language (OWL) from the World Wide Web Consortium (W3C).[1] This language is equipped with formal logic-based semantics, which ensures that the formulated statements are interpreted unambiguously. That is, any definition of a concept made in an FA ontology, e.g., a physical fault, is encoded as a logic axiom, which has only one correct interpretation.

In this paper, we present an approach to formalization of FA knowledge in an ontology and make the following contributions:

- We present by example basic notions of encoding FA knowledge as an OWL ontology;
- We suggest a general high-level ontology that provides definitions of basic concepts and relations used in the FA domain, which can be extended with specific definitions suitable for a particular lab; and
- Possible applications of the ontology are demonstrated on a simple use case of a physical fault due to a fused wire.

## Preliminaries on Ontologies

In this section, we provide a short overview of ontologies formulated using OWL language sufficient to encode knowledge about the FA domain. Thus, an ontology formulated using OWL language might include:

- *individuals* describing real-world entities, like sample integrated circuits of a job or tools available in a lab,
- *classes* defining parts of the world by summarizing properties of a collection of individuals, e.g., a class `OpticalMicroscope` that comprises all individual microscopes installed in the lab or a class `IncomingInspection` including all database records describing applications of this method, and
- *properties* determining relations between two individuals (and data), such as a property `magnification` relates an individual microscope with a number (data) indicating the

[1] https://www.w3.org/TR/owl2-overview/

magnification of an eyepiece, or a property `uses_tool` linking methods, like `IncomingInspection`, with tools, like `OpticalMicroscope`.

Statements in an ontology, called triples, in an ontology consist of three elements: subject, predicate, and object. For instance, one can state that an optical microscope with an identifier `optM12` was used in an incoming inspection `inc_3356` with a triple

```
inc_3356 uses_tool optM_12
```

A triple

```
optM_12 magnification 100
```

indicates a relation between an individual and data (literals). The difference between object-to-object and object-to-literal relations is rather technical and handled appropriately by most ontology editors, like Protégé.[2]

These triples, unlike database records, make no assumption that everything known is stored in the ontology. That is, an ontology including a triple shown above assumes that other analyses, optical microscopes, and relations between them might also exist but are simply not known at the moment. In the database context, the assumption is different – all records constitute complete knowledge and anything else does not exist. As a result, multiple ontologies can comprise knowledge about the target domain and use import functionality to construct complex ontologies. For example, one can split FA ontology into numerous parts: A high-level ontology including general statements about the FA domain, like the general `Fault`, `Process`, or `Equipment` definitions, and many low-level ontologies defining specific knowledge about subdomains, like `FusedWire` fault or `OpticalMicroscope` tool. Low-level ontologies can import the high-level one to ensure consistency of definitions among them. In addition, high-level ontologies can be used to ensure interoperability of definitions with other ontologies, e.g., used in FMEA or production.

In addition to relations between individuals, ontologies can comprise definitions of classes, also known as *concepts* in the literature, and their hierarchies. One can use `Type` and `SubClassOf` properties to express that an individual belongs to a class and that a class is a subclass of another one, respectively.[3] For instance, a triple

```
optM_12 Type OpticalMicroscope
```

means that the individual belongs to a class of optical microscopes. The following triple

```
OpticalMicroscope SubClassOf Equipment
```

states that any individual of a class `OpticalMicroscope` also belongs to the class `Equipment`. Using these two triples, one can deduce that `optM_12` is also an individual of the class `Equipment`. Each class hierarchy starts with the most general class `owl:Thing`, which comprises all individuals and consequently all classes, and ends with `owl:Nothing`, which corresponds to an empty set. All subclasses of `owl:Nothing` are also empty and, therefore, are called *unsatisfiable*.

More complex class definitions can be done using Boolean connectives such as `and`, `or`, and `not`. Using triples to encode such class expressions might be complicated since the resulting documents are hard to read. Therefore, many OWL syntax definitions, like Manchester Syntax, allow for the formulation of complex axioms that are then mapped to triples on the background. For instance, one can define that two classes are disjoint[4], i.e., share no individuals, using the following axiom:

```
Fault SubClassOf not Equipment
```

The fact that a physical fault `FusedWire` also has the fault mechanism EIPD can be represented by conjunction:

```
FusedWire SubClassOf Physical and EIPD
```

We can easily represent all these examples on a Venn diagram, as shown in Figure 1.

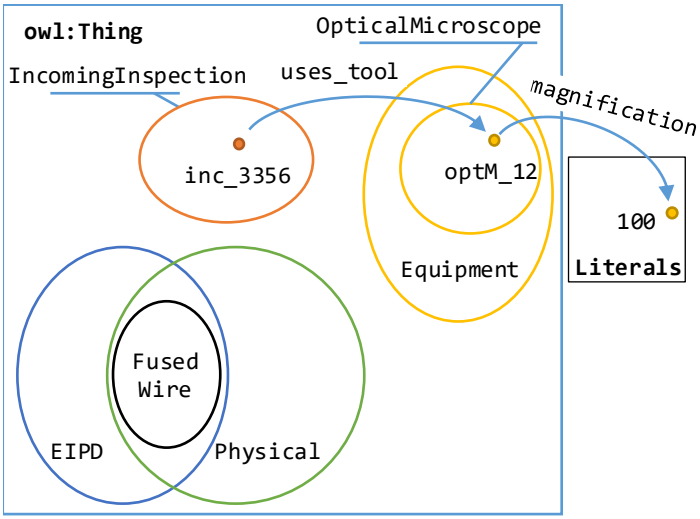


*Figure 1: Example ontology statements shown on a Venn diagram*

Similar to classes, *properties*, also known as *roles* in the literature, can be organized in hierarchies using the `SubPropertyOf` property. For instance, during analysis, we would like to differentiate between the precise and rough location of a fault. Nevertheless, both properties indicate the location of a fault. This fact can be expressed by axioms:

```
is_located_detail SubPropertyOf is_located_at
is_located_coarse SubPropertyOf is_located_at
```

Moreover, properties can be restricted in multiple ways to make the ontology definitions more precise. Properties `Domain` and `Range` allow one to define restrictions on the domain and range of a property. For instance, one can restrict the domain of the property `uses_tool` to the class of all failure analysis methods and the range to equipment with the two axioms:

[2] https://protege.stanford.edu/

[3] In this paper we use the Manchester Syntax of the OWL language (https://www.w3.org/2007/OWL/wiki/ManchesterSyntax). This syntax was developed to simplify the textual representation of ontology axioms, since the default XML-based representation is often wordy and hard to understand. This syntax is also used in the most popular ontology editor Protégé.

[4] In OWL this relation can also be expressed using the `DisjointWith` keyword: `Fault DisjointWith Equipment`.

```
uses_tool Domain FailureAnalysisMethod
uses_tool Range Equipment
```

Properties might have modifiers that define their characteristics, such as inverse properties, functionality or transitivity of the relation, etc. For instance, one can define that if a method uses a tool, then a tool is also used by this method and vice verse:

```
uses_tool InverseOf used_by_method
```

Properties can further be restricted in class expressions using existential – `some` – and universal – `only` – quantifiers. Existential restrictions require each first argument of a property to allow for the existence of at least one individual of a specific type as a second argument. For instance, one can require that for every individual fused wire fault, there exists at least one individual location in package wire bonding where this fault occurs:

```
FusedWire SubClassOf is_located_detail
                        some LocPkgWireBonding
```

Since ontologies assume that there might be some unknown knowledge about the domain (open world assumption), this statement is interpreted differently from the databases domain. That is, an ontology reasoner will only check for a possibility for such an individual to exist without requiring an ontology to comprise it because, e.g., it might be added in the future.

The universal quantification is much more restrictive. For any pair of individuals in an ontology related over a property, universal restriction implies that any individual appearing as a second argument of a property is of a given type. For instance, one can define that whenever a fused wire individual is localized with anything, then this individual is an element of the package localization tool class:

```
FusedWire SubClassOf is_localized_with
                      only PackageLocalization
```

Therefore, universal restrictions must be used very carefully since, also, in cases when a fused wire is detected using some different tool, it will be added to the class of package localization tools.

Besides the modeling possibilities discussed above, many other features of the OWL language can be used to encode detailed knowledge about the target domain. For instance, one can specify the equality and difference of two individuals using `SameAs` and `DifferentFrom`, respectively. The Boolean connectives can be used in property restrictions to define complex class expressions. For instance, one can specify that a die crack can be located roughly in die- or interface-related locations:

```
FusedWire SubClassOf is_located_rough
            some (DieRelated or InterfaceRelated)
```

Readers are referred to, e.g., [1], [2], or [3], for an in-depth discussion of these and other OWL features applied in ontologies.

## Ontologies in Failure Analysis

Ontology applications in the FA domain provide various benefits. The development of FA ontology, suggested in this paper, was driven by practical use cases identified in our FA Labs. During this process, we identified the following advantages of ontology application in the FA domain:

- *Standardization of terms and their interpretation by FA engineers*. The process of knowledge encoding initiates a discussion between FA experts regarding the definition of failure classes, their causes and relations to locations, tools, methods, technologies. Already this discussion among experts contributes to the standardization of terms used in FA reports and enhances their overall consistency.
- *Creation of a machine-readable thesaurus that can be used to integrate various FA information systems*. There are many possible ways to store the standardization process results, like text documents or databases. Among them, ontologies based on the OWL language offer a reach set of features that ease the encoding of knowledge about standards, as described in the previous section. The ontological definitions are machine-readable and can be used as an integration platform for various information systems used in FA, e.g.:
  - databases, where classes can be used as identifiers,
  - wikis, which pages can directly refer to the ontology, or
  - Office documents, which properties can be used to store references to the classes and properties.

  An information retrieval system, such as a search engine, can use all these embeddings of the ontology definitions to find and group information from all these systems in one central access point.
- *Provide seed knowledge for machine learning systems*. An essential feature of ontologies is that they allow for the annotation of class and property definitions. The annotations are XML[5] snippets that can comprise additional data like keywords used for tools, faults, or methods, different explanations and observations, references to additional documents describing best practices or instructions. Machine learning algorithms can use all this structured data to train models to solve various problems, including predicting possible failures and their electrical signatures for textual representation or recommendation of analyses that might result in faster failure localization.

*Define restrictions for FA jobs*. Support FA systems, such as databases tracking jobs or analysis workflow systems, can use ontology definitions to generate routines for automatic testing of requirements to data that engineers enter. For example, whenever a fused wire is detected, its rough location, according to the ontology, is expected in die- or interface-related locations. If the location belongs to a different group, a testing system might raise an issue, which requires either changing the location or extending the ontology. Other restrictions can be used to control the analysis workflows in a lab by defining restrictions on sequences of operations, e.g., by specifying prerequisites and successors of each operation. If an engineer registers the execution of a method on a sample, this action can be validated with respect to its prerequisites and the results communicated to the

[5] https://www.w3.org/TR/xml/

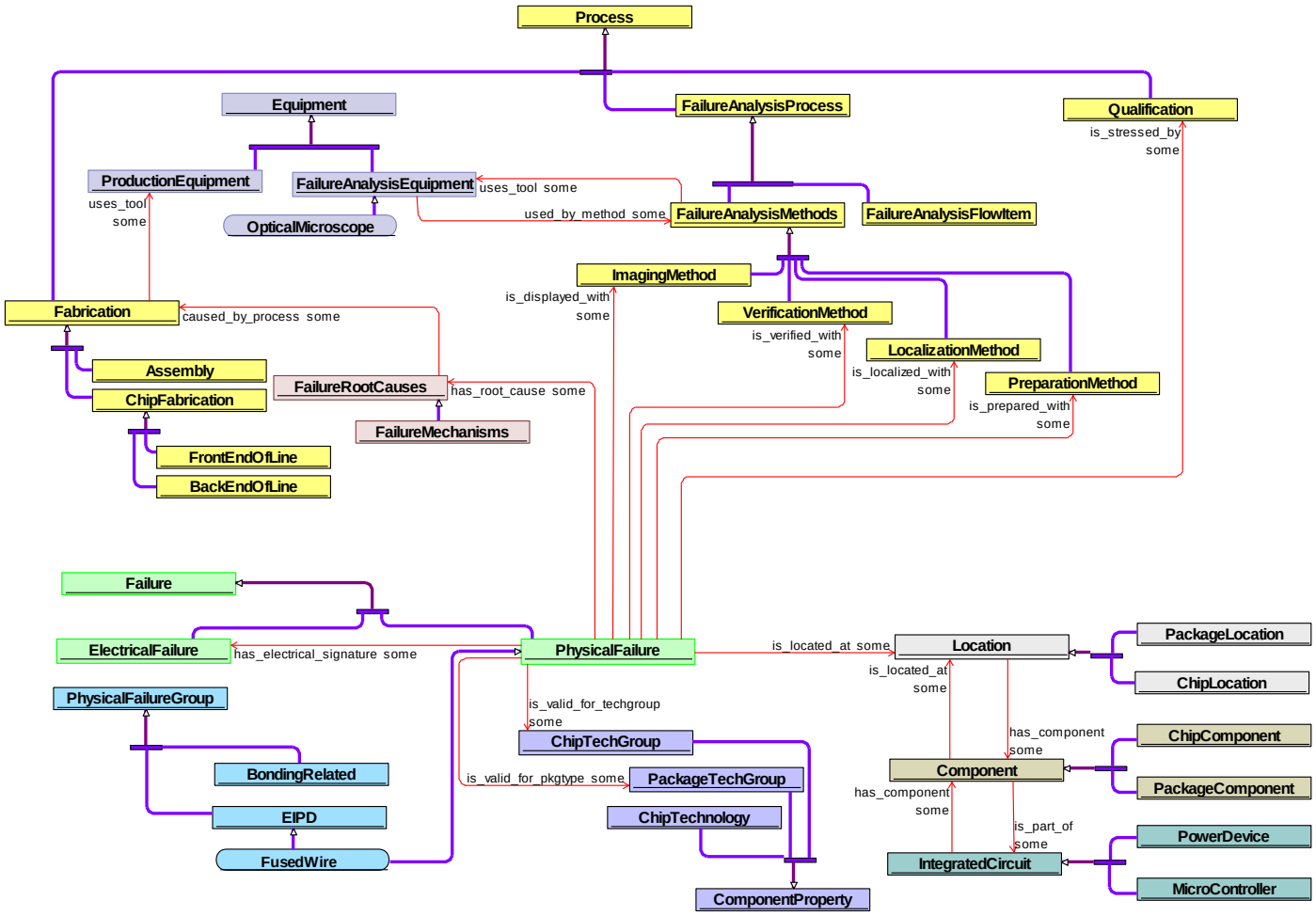


*Figure 2: General structure of the FA ontology*

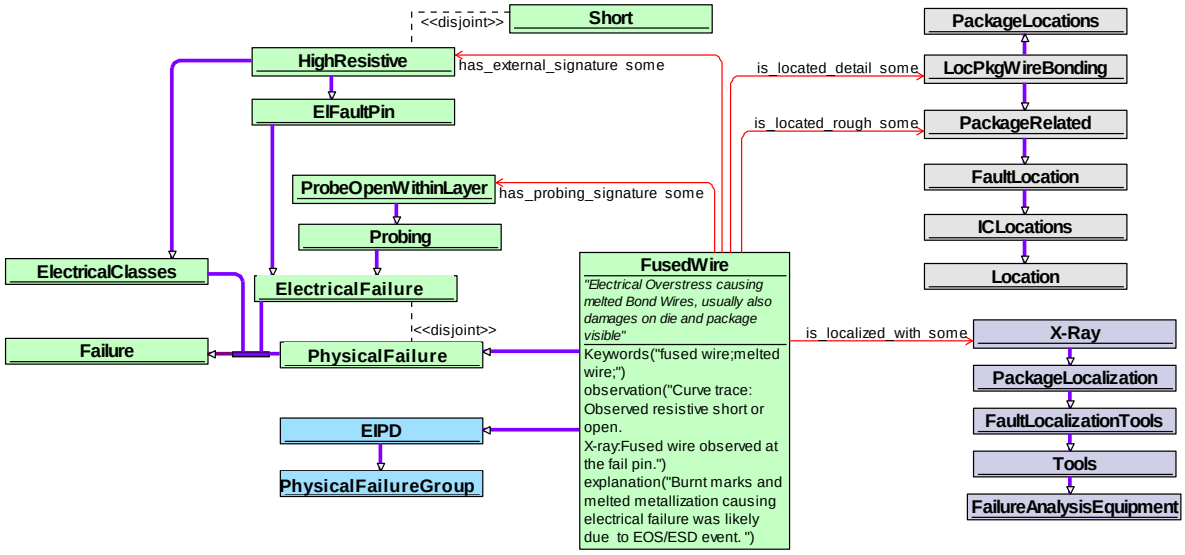


*Figure 3: Properties of physical defect: FusedWire*

expert. Similarly, by analyzing possible successors, one can inform an engineer that the execution of this method will make the application of other methods impossible.

This paper suggests a high-level FA ontology, presented in Figure 2, that defines a general class hierarchy capturing the previously discussed points.[6] The main subtree, shown in green, provides central definitions of `Failure`, `PhysicalFailure`, and `ElectricalFailure`. These classes represent the root of the failure and comprise all possible failures as subclasses. Consider an example presented in Figure 3. `FusedWire` is a subclass of `PhysicalFailure` with an electrical signature `ProbeOpenWithinLayer` since the property `has_probing_signature` is defined in the ontology as a subproperty of `has_electrical_signature`. The disjoint property, defined for the classes `PhysicalFailure` and `ElectricalFailure`, indicates that these two classes and any pair of their subclasses have no intersections. Moreover, the `FusedWire` failure is an electrically induced physical damage (`EIPD`) from `PhysicalFailureGroup`. The latter class groups all physical failures according to their mechanisms. The `FusedWire` failure `is_localized_with` `X-Ray`, which is a subclass of the `PackageLocalization` methods. Finally, the example shows that the expected location of the failure we consider is on the package wire bonding. Roughly, this location is categorized as `PackageRelated`.

In addition to the classes discussed in the example, the core ontology allows for the definition of integrated circuit components, where a failure might occur, and provide its root causes. On the one hand, this information might be used to provide statistics about failures for FA or development teams. On the other, it allows information systems to provide more detailed data about the device at hand. A similar idea is behind the definition of component properties, such as chip and package technology groups.

Another major part of the ontology, indicated in yellow, considers the failure analysis and fabrication processes. The localization part comprises general classes denoting various methods and their relation to the localization of a particular failure. This information allows for the creation of failure localization workflows, because it provides all necessary data about tools required to localize a failure. Moreover, such knowledge enables automatic control of the reporting consistency by checking whether the tools used according to the databases are suitable for the reported failure. The fabrication part provides an essential link between the FA domain and the production.

## Use Cases

Since the ontology is able to host all meta-knowledge about the FA- process, the use cases in Failure Analysis are manifold. In the following, we present three applications pursued in our FA Labs.

### Host failure catalog for FA report classification

The first application uses text classification to categorize failure analysis reports into different classes like electrical signature, physical root cause, or analysis methods [4]. The classes used to label the data are derived from the `Failure` entity tree, see Figure 2. Then, a learning algorithm trains a model to predict failure classes using available history jobs in FA databases. This application allows for a statistical evaluation of the failure signatures, as shown in Figure 4. Moreover, properties like `valid_for_technology` or `is_located` can be used to quickly narrow down a long list of different failure modes to the one that applies to the actual job.

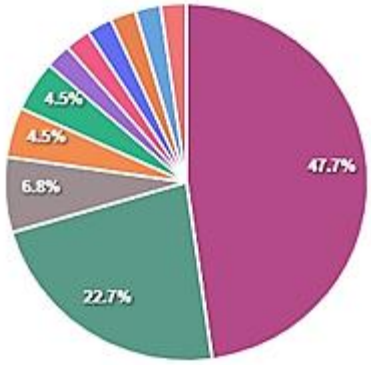


*Figure 4: Physical Failure distribution of a product for jobs where the electrical signature is `Pin Short`. Different colors indicate different physical failures.*

### Tag documents and images, enrich search queries

Search engines can use ontologies to enhance queries with domain-specific information while searching over FA reports, as Google engine does it for the Web.[7] For instance, given a query with the term `dislocation`, a search engine might automatically extend it with the meta-knowledge stored in the ontology annotations to look also for documents comprising the term `crystal defect`. The obtained search results can further be refined and grouped according to the class hierarchy of terms and expressions used in FA. Furthermore, the engine would know that the term `dislocation` is not meant in a medical but in a crystallographic context whenever document collections over a mixed domain are considered.

Finally, one can extend existing FA information systems to annotate images, results of single analyses, or the whole reports automatically during parsing.

### Derive knowledge about methods and flow

Workflow systems might exploit information about analysis tasks by using properties like `has_prerequisite` or `has_successor` to check if an engineer executes an analysis task is in accordance with a previously defined analysis flow or if some required tasks are missing. Links to suitable equipment based

[6] The ontology can be accessed in the WebProtégé system [5] at http://web-protege-aics.aau.at Guest account: visitor/visitor

[7] https://blog.google/products/search/about-knowledge-graph-and-knowledge-panels/ (retrieved 07.04.2021)

on job meta-information, like package or technology, could also be derived based on information stored in the ontology.

## Conclusions and Outlook

The presented work suggests an approach to the development and application of ontologies in the FA domain. The proposed high-level ontology reflects our experience in creating intelligent assistants for FA engineers, as indicated by the use cases described above. We share the ontology with the community to facilitate the discussion on the concepts that can widely be accepted as a foundation for various FA terminology standardization efforts in different organizations. In our future work, we will focus on implementing further extensions of the presented ontology by identifying new use cases in which the knowledge-based approach can provide significant simplifications and quality improvements of FA routines. For instance, one of the most interesting extensions is the prediction and recommendation of the next task in the failure analysis flow allowing an engineer to localize a failure more efficiently. In this case, the descriptions of tasks and their properties specified in the ontology must be associated with database logs of task sequences leading to fault localization. Then, machine learning methods can be used to analyze the state of the current job and suggest the most probable extension of the analysis flow with respect to the historical data. Another interesting problem that can be done using the knowledge stored in the ontology is the mining and analysis of causal relations between different concepts in the failure analysis reports. In this scenario, specific machine learning methods for named entity recognition must be applied to identify ontology concepts corresponding, e.g., to tools, failures, or their electrical signatures in the texts. Next, the extracted combinations of concepts can be organized in a knowledge graph by instantiating relations between them available in the ontology. Whenever there is evidence of a strong relation between concepts, which does not exist in the ontology, the algorithm can generate a recommendation for ontology extension. Moreover, the ontology can be used to establish links between FA and other related domains such as failure mode and effects analysis (FMEA) using (semi-)automated ontology alignment methods. The latter allows for the comparison of two ontologies and the identification of patterns common to both of them. These patterns are then converted into matches, which are ontology axioms establishing missing links between the ontologies. For instance, if an alignment method finds that two concepts are used similarly in both ontologies, it can generate an axiom stating that both concepts are equivalent. Such matching might enable interoperability between FA and FMEA systems as described in the paper for FA information systems.

## Acknowledgments

This work was partially supported by the EFRE, REACT-EU and Carinthian Economic Promotion Fund (Project ONTIS, Contract No. KWF-3520|34826|50900).

## References

[1] D. Allemang und J. A. Hendler, Semantic Web for the Working Ontologist - Effective Modeling in RDFS and OWL, Morgan Kaufmann, 2011.

[2] F. Baader, I. Horrocks und U. Sattler, „Description Logics,“ in *Handbook of Knowledge Representation*, Elsevier, 2008, pp. 135-179.

[3] P. Hitzler, M. Krötzsch und S. Rudolph, Foundations of Semantic Web Technologies, Chapman and Hall/CRC Press, 2010.

[4] F. F. Platter, Recommendation System for Semiconductor Failure Analysis (Master thesis), Klagenfurt: University Klagenfurt, 2019.

[5] T. Tudorache, C. Nyulas, N. F. Noy und M. A. Musen, „WebProtégé: A Collaborative Ontology Editor and Knowledge Acquisition Tool for the Web,“ *Semantic Web*, Bd. 4, Nr. 1, pp. 89-99, 2013.