SIMILARITY MEASUREMENT FOR ONTOLOGY-BASED PRODUCT CONFIGURATIONS

Niklas Baumbach[(1)], Diana Peters[(1)], Stefan Humbla[(2)], and Lars Wagner[(2)]

[(1)]*DLR Institute of Data Science*
Mälzerstraße 3, 07745 Jena, Germany
Email: niklas.baumbach@dlr.de
Email: diana.peters@dlr.de

[(3)]*Jena-Optronik GmbH*
Otto-Eppenstein-Straße 3, 07745 Jena, Germany
Email: Stefan.Humbla@jena-optronik.de
Email: Lars.Wagner@jena-optronik.de

ABSTRACT

Many technical devices are complex systems with a wide range of configuration options. Such options can be selected, but not independently and with restrictions and implications. The matching between customers requirements to existing configurations that a supplier can build with sensible effort and technical limitations is currently mostly done manually based on experience. In this paper, we present an approach to automate this matching, based on ontologies and (tailored) similarity measures.
We explain an implementation that works on a reduced parameter set for star trackers, based on the example of ASTRO APS® from Jena-Optronik GmbH. This system is able to automatically suggest the best fitting and buildable configuration for any requirement configuration and we point out possible ways to extend it in the future.

1. INTRODUCTION

In this paper, we present an approach to find similar product configurations based on ontologies and explain how this can lead to a new and more efficient way to exchange technical information along the supply chain.
Currently, suppliers usually provide the technical data of their products via PDF data sheets on their websites. Engineers who design a new spacecraft have to find the data sheets, compare the technical properties of the described product with their requirements, and then extract the values of the product they finally want to use. This is even more challenging as not all parameters are easily comparable (e.g. imperial vs metric system, different error terms, different functionalities).
Although many new configurations can be realized, the mutual aim is to re-use existing configurations whenever possible. This reduces non-recurring effort and cost and enables the re-use of existing qualification campaigns and data. Existing configurations are usually well proven and often already in use (space flight heritage), which is a strong argument for the usage of established configurations. However, it is hard to find the closest existing configuration for a given set of requirements when none of existing configurations completely meets all required parameters.
Often, clarification regarding buildability of certain configurations and their qualification status is necessary — e.g. via phone or e-mail. We discussed before why this process can and should be improved [1]. In the end, the engineer does not really want to know the technical capabilities of the supplier, but wants to know, if the supplier can meet the core requirements. Often a discussion of a specific configuration is necessary to identify which requirement has higher weight and which can be changed to fit existing configuration while maintaining the customer's overall system performance.
To automate the described matching process, we need the following:

1. a model of what the supplier is able to build,
2. a model of what the customer wants built,
3. an interface to get both together, and
4. something that decides whether both models match.

In this paper, we present a system that implements points 1 and 4 for a star tracker as an example case, but the idea can be easily adapted to other products. Points 1 and 2 are addressed by an ontology, point 3 by SPARQL[1] requests, and point 4 by a tailored similarity measure. The connection between customer and supplier is currently not yet implemented via an API, but the models are put together on supplier side. This will be added in a next version of

[1]This is a recursive acronym: SPARQL Protocol and RDF Query Language

the code. The system is able to automatically suggest the most fitting actually buildable configuration for any requirement configuration. We are aware that this system will not revolutionize the whole supply chain information exchange, but we think that it is a starting point.

2. STATE OF THE ART

Many publications have dealt with various aspects of the problem how to represent product configurations through ontologies. One important aspect is the description and validation of constraints to ensure integrity, because some properties of a configuration might exclude or imply others. The first to address this aspect were Soininen et al. in 1998 [2]. With the standardization of semantic web technologies by the W3C[2] and the spread of Web Ontology Language (OWL), researchers used and extended these standards for configuration options, e.g., Yang et al. [3] and Ardito et al. [4]. Multiple works addressed the problems of defining integrity constraints in OWL that follow from the Open World Assumption [5][6][7]. Bergner et. al. presented an approach using the SPARQL Inferencing Notation (SPIN) created by the company TopQuadrant [8][9]. A year later the W3C released the Shapes Constraint Language (SHACL) [10], which is based on SPIN and was used in this work.
In order to find matching existing configurations for a given requirement configuration, a way to determine the similarity of configurations is needed. A method to calculate the similarity of two objects is called a similarity measure, i.e. a function with the range $[0; 1]$ where 1 means identity of the two input objects. Paul Jaccard formulated the first simple similarity measure for sets (nowadays called Jaccard-Index) [11]. More recent approaches are used in the field of data analysis and clustering and are often based on probabilistic approaches. One widely used approach to cluster mixed numerical and categorical data is the adapted K-means algorithm by Ahmad and Dey which uses the distribution of values among the data points to calculate the dissimilarity of objects [12]. A dissimilarity measure can be easily converted into a similarity measure [13]. This approach was developed further and fine-tuned for different problems, e.g. by [14], [15], and [16]. However, none of those approaches promise big improvements over the Ahmad and Dey approach for our use case.
The last class of similarity measures considered in this work are the semantic ones. Apart from distributional semantic measures, which are used in text analysis and are also based on probabilities, these include knowledge-base semantic measures. They use the taxonomy and relations of an ontology or another knowledge representation system to determine the similarity of classes, properties, and instances [17]. Bisson was amongst the first who worked on this problem [18]. Approaches for the similarity of classes and properties, e.g. [19], can not be used as similarity measures for instance data like explicit product configurations. A fitting semantic measure was described by Rydzynski, Felic, and Zirpins [20], but not used in this work in favor of a self-developed graph-based measure that includes suggestions from the domain experts of Jena-Optronik. This method is described in the following section.
Ehresmann et al. [21] introduced an approach to optimize the whole space system according to the mission goal — focusing there on an electric propulsion system. While there each component is optimized, our approach focuses on finding the best-fitting existing component.

[2]World Wide Web Consortium, https://www.w3.org/

3. IMPLEMENTATION

This section contains brief explanations of the example ontology, of the three implemented similarity measures, and of the setup which was used to compare them. We omit some details as the implementation, including a running example, can be found at [22].

3.1 Ontology

The implemented ontology is a simplified description of the domain of star trackers and contains the most important components and properties of the star tracker ASTRO APS® by Jena-Optronik for demonstration purposes. SHACL-Shapes are used to restrict invalid configuration options. The URI of the ontology is *http://ontology.dlr.de/spacecraft-parts/star-tracker#* and the associated prefix is *st*.
The class *st:StarTracker* describes the general concept star tracker and an instance of this class is a specific star tracker with concrete properties, i.e. a configuration. The set of ASTRO APS configurations is modeled by the class *st:AstroAps*, which is a subclass of *st:StarTracker*. Each important component of the star tracker is modeled by a dedicated class and can be linked to the configuration through a corresponding *owl:ObjectProperty*. These are the detector (*st:Detector*), the power converter (*st:PowerConverter*) and the communication protocol (*st:CommunicationProtocol*). The properties modeled here are:

- *st:hasUpdateRate* of type *xsd:double* defines how many updates a star tracker sends to the spacecraft
- *st:hasMaxUpdateRate* of type *xsd:double* defines the maximum update rate of a detector
- *st:hasNominalVoltage* of type *xsd:double* defines the nominal voltage of a power converter
- *st:isInitialOn* of type *xsd:boolean* defines if the power converter is turned on when a supply voltage is applied
- *st:isMilCompatible* of type *xsd:boolean* defines if the instance is compatible with MIL-STD-1553

- *st:hasLcl* of type *xsd:boolean* defines if the power converter has a protection against high current peaks (”latching current limiter”)

A communication protocol has no properties in this simplified description. Instances are distinguished by their labels.

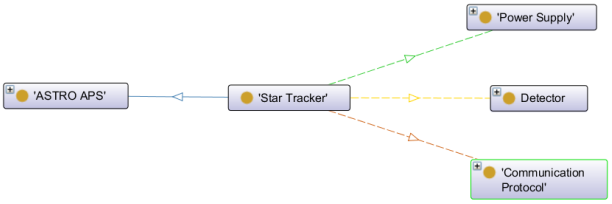

Figure 1: The ontology classes as graph

3.2 Similarity Measures

Three similarity measures with different complexities are considered in this paper. The simplest similarity measure is the set-based Jaccard-Index [11], which can be expressed by Eq. 1.

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|} \qquad (1)$$

In our case, the properties of two configurations are the items of sets A and B. The most important feature of the Jaccard-Index in this context is that it does not understand numerical properties. This means, that for example two configurations with the update rates 8 and 9 Hz (and all other properties equal) are as similar as two configurations with the update rates 1 and 100 Hz (again, all other properties equal).
The second similarity measure uses the dissimilarity from the *K-means algorithm for mixed numerical and categorical data* by Ahmad and Dey, which is based on a probabilistic approach. The distance between two categorical values is computed as a function of their overall distribution and co-occurrence with other properties (see [12] for an in-depth explanation). The distance between two numerical values is their difference weighted by a significance. The significance is calculated by first discretizing the numbers and then treating them as categorical values. The output of the distance measure is converted to a similarity measure by Eq. 2.

$$s(i,j) = \frac{1}{1 + d(i,j)} \qquad (2)$$

3.2.1 Tailored Approach

The third function is, after fine-tuning for this specific use case, no longer a similarity measure as it is not symmetric. It receives two instances of an OWL class and determines how similar the second instance is to the first one (”reference instance”) by comparing the values of *owl:ObjectProperties* and *owl:DatatypeProperties*.
The function calls itself recursively with the property values (which are also instances) as arguments if the property is an *owl:ObjectProperty*. For categorical *owl:DatatypeProperties*, a binary measure is used which returns 1 if the two values are equal and 0 otherwise.
Eq. 3 is used for numerical properties. The handling of numeric properties is influenced by an *owl:AnnotationProperty*, *st:preferredDirection* in our ontology. This property is, beneath the asymmetry of Eq. 3, the second reasons why this approach is asymmetric: It encodes if a value should be as high as possible, as low as possible, or as close to the reference value as possible. The first two options give the property a direction.

$$\delta_p(v_1, v_2) = \begin{cases} 1 & if\ v_2 = v_1 \\ \frac{v_1 - v_2}{v_1 - min(p)} & if\ v_2 < v_1 \\ \frac{v_2 - v_1}{max(p) - v_1} & if\ v_2 > v_1 \end{cases} \qquad (3)$$

The similarity ratings of all properties are added up to a weighted sum, which represents how similar the second instance is to the first one. The weights of the different properties were defined by hand according to the experience at Jena-Optronik.

3.3 Test Setup

To compare the three approaches and to optimize the third one, tests were conducted as follows:
14 ASTRO APS configurations were provided by Jena-Optronik as existing configurations as well as retrictions and limitations to create new configurations. Based on a permutation of the parameter sets, 3600 query, or requirement, configurations were created and evaluated (see [22] for details). Each of the three similarity measures compared each requirement configuration against the 14 existing configurations and provided a similarity value for each comparison. In order to provide a comparison between the algorithms, five requirement configurations where selected, where the result order of the 5 best fitting existing configurations was different [3]. These cases were discussed with the experts from Jena-Optronik and results were compared with ratings of the algorithms.

[3] This means: the five highest rated buildable configurations of two similarity measures match in a maximum of two places

4. RESULTS

Fig. 2 shows in how many cases out of the 3600 query configurations the three algorithms agreed pair-wise with each other about the Top 3, Top 2, and Top 1 best fitting buildable configurations. All three algorithms agreed in 6.10% of all cases about the Top 3, in 15.0% about the Top 2, and in 43,6% about the Top 1. Therefore, in less than 50% of all cases all three algorithms agreed on the best matching buildable configuration. None of the three algorithms agreed about the Top 3 in 1.7% of all cases, about the Top in 2.14% of all cases, and about the Top 1 in 9.14% of all cases. The ratings of Jaccard and Ahmad Dey were more often identical than any of them and our tailored algorithm.

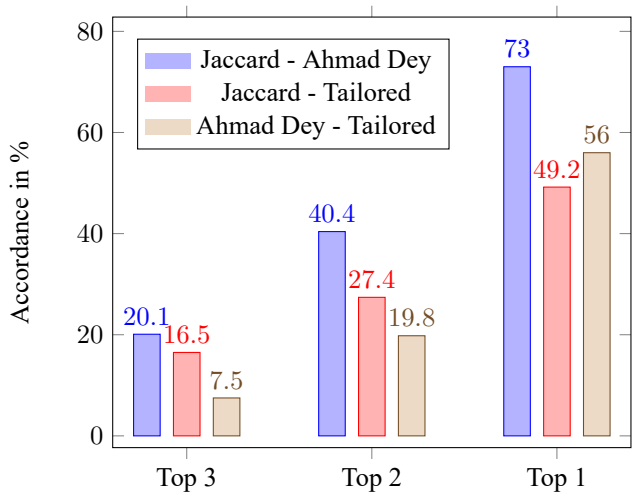

Figure 2: Accordance of the different similarity measures for the most similar 3, 2, and 1 configuration(s).

Tab. 1 shows the total runtimes of the algorithms for calculating the similarity of 3600 queries to the 14 reference configurations. Since all three implementations use the same data access, differences in the runtimes must come from the algorithms themselves. This point is further discussed in the next section.

Method	Runtime
Jaccard	81.9 s
Ahmad Dey	104.8 s
Tailored	117.9 s

Table 1: Total runtimes for the three implementations using the sample parameter set, 3600 queries and 14 reference configurations.

In order to compare results, five interesting queries from the set of 3600 were selected. Their parameters are shown in Tab. 2 and are discussed in the next section.

Upd. rate	Nom Volt.	LCL	init. on	MIL	Protocol	Det.
10	53	✓	✓	✓	SCDI	H2
8	53	✗	✗	✓	HDLC	H2
8	33	✓	✓	✓	UART	S1
10	88	✗	✗	✓	SCDI	S1
8	28	✗	✗	✗	HDLC	S1

Table 2: Query configurations used for evaluation that show differences in results between the algorithms (update rate in Hz, nominal voltage in Volts, LCL equipped, device is initial on, MIL-STD-1553 compatible, communication protocol, and detector). Detectors are either STAR1000 (S1) or HAS2 (H2).

5. DISCUSSION

Evaluation of non-agreement configurations: We explain one query configuration, where the three algorithms disagreed about the Top 5 matching buildable configurations (definition in section 3), in detail. The query configuration of this example has the following properties:

- Update Rate: 10Hz
- Nominal Voltage: 53V
- LCL equipped
- is initially on
- compatible to MIL-STD-1553
- Communication Protocol SCDI
- Detector HAS2

Tab. 3 shows the most three most similar rated configurations for each of the three algorithms using the example query. First result value is the number of the rated existing configuration, values in brackets are the similarity values.

Method	1st	2nd	3rd
Jaccard	7 (0.38)	8 (0.38)	9 (0.38)
Ahmad Dey	12 (0.58)	9 (0.569)	7 (0.567)
Tailored	7 (0.94)	8 (0.642)	14 (0.64)

Table 3: Top 3 most similar configurations for example query according to the different similarity measures. Green highlight is the preferred solution based on expertise while orange is a less suited configuration.

A human expert would choose configuration 7 as best matching for the example query configuration. The nominal voltage is the only property that is not a perfect match. In the existing configuration it is higher than in the query configuration, which is usually no issue due to wide operational voltage ranges. Configurations 8, 9, and 12 were also suggested by the algorithms, but they have the wrong

communication protocol and the wrong voltage range. Such deviations are usually unacceptable to the customer. Both the tailored and the Jaccard algorithm rate buildable configuration 7 as the best matching, but the Jaccard algorithm rates buildable configurations 8 and 9 exactly as good as 7, which is incorrect. The Ahmad and Dey based algorithm rates buildable configurations 12 and 9 before 7, which is also incorrect.

For all query configurations that were discussed with experts, the tailored algorithm voted for the same configuration as the human experts. This was not surprising as the algorithm was tailored using expert rating. However, it was not certain that the same weighting of parameters for the algorithm would be working perfect for all queries. The tailored algorithm performed quite well and showed the best results.

The main problem of the Jaccard algorithm we encountered, was that it gives different buildable configurations the exactly same rating. Therefore, there is no clear difference between configurations and the solution contains a wide range of solutions. The Ahmad and Dey based algorithm gives more distinct ratings, but often rates unsuitable configurations the highest.

Runtimes: We did not measure the runtimes for data access and actual similarity evaluation separately, though this might be an interesting point for future work. We see, however, that the runtimes do not differ by magnitudes between the different algorithms. We also see that the runtimes become longer with increasing complexity of the algorithm, as expected

6. CONCLUSION AND OUTLOOK

We implemented a system to find the best matching buildable configuration of a star tracker to a given requirement configuration — for a simplified and reduced parameter set. The algorithm can be applied to more complex systems (e.g. as described in ECSS-E-ST-60-20C) and is not limited to the demonstration system. Modeling all parameters is done easily, more complex is the task of transferring knowledge about dependencies and weighting of parameters to achieve robust behavior and results as shown. As mentioned before, the runtimes for the algorithm(s) can be optimized. The data access should become faster if data is kept in RAM instead of requested from the ontology every time. The runtime of the algorithm itself should become faster by parallelization as individual query configurations are not related.

Another point to be pursued further is the practical implementation in the software ecosystems of an actual supplier and an actual customer. It would be interesting to expand the system beyond technical aspects and to cover, e.g., potential delivery dates. Another direction for future work is the adaption to other satellite components and sub-systems, e.g., batteries or solar panels. The interesting problems, however, will arise where data will be exchanged that is considered intellectual property of a company. We are certain that the solutions to those are not purely technical.

Overall, we hope to show with this system that there are other ways of data exchange than the currently established ones. Likely, even more complex systems will not cover all relevant technical data along the whole supply chain as current data sheets also only cover a subset of the overall data and rarely any dependencies. However, we would like to see how far this approach can be stretched and which new problems arise at its borders.

REFERENCES

[1] Peters, D., Fischer, P.M., Schäfer, P.M., Opasjumruskit, K. & Gerndt, A. (2019). Digital Availability of Product Information for Collaborative Engineering of Spacecraft. In *International Conference on Cooperative Design, Visualization and Engineering*. Springer, pp 74–83.

[2] Soininen, T., Tiihonen, J., Männistö, T. & Sulonen, R. (1998). Towards a general ontology of configuration. *Artificial intelligence for engineering design analysis and manufacturing* **12**, 357–372. doi: 10.1017/S0890060498124083.

[3] Yang, D., Miao, R., Wu, H. & Zhou, Y. (2009). Product configuration knowledge modeling using ontology web language. *Expert Systems with Applications* **36**(3, Part 1), 4399 – 4411. ISSN 0957-4174. doi:https://doi.org/10.1016/j.eswa.2008.05.026.

[4] Ardito, C., Barricelli, B., Buono, P., Costabile, M., Lanzilotti, R., Piccinno, A. & Valtolina, S. (2011). An Ontology-Based Approach to Product Customization. In *2017 2nd International Conference on Knowledge Engineering and Applications (ICKEA)*. pp 92–106. doi:10.1007/978-3-642-21530-8_9.

[5] Motik, B., Horrocks, I. & Sattler, U. (2009). Bridging the gap between OWL and relational databases. *Journal of Web Semantics* **7**(2), 74 – 89. ISSN 1570-8268. doi:https://doi.org/10.1016/j.websem.2009.02.001.

[6] Tao, J. (2010). Adding Integrity Constraints to the Semantic Web for Instance Data Evaluation. In *International Semantic Web Conference*.

[7] Tao, J. (2012). *Integrity Constraints for the Semantic Web: An OWL 2 Dl Extension*. Ph.D. thesis, USA.

[8] Bergner, S., Bartelt, C., Bergner, K. & Rausch, A. (2016). Methodology for an Ontology-Driven Product Configuration Process.

[9] TopQuadrant. SPIN (SPARQL Inferencing Notation).

[10] W3C RDF Data Shapes Working Group. Shapes Constraint Language (SHACL).

[11] Jaccard, P. (1902). Lois de distribution florale dans la zone alpine. *Bulletin de la Société vaudoise des sciences naturelles* **38**, 69–130. doi:10.5169/seals-266762.

[12] Ahmad, A. & Dey, L. (2007). A k-mean clustering algorithm for mixed numeric and categorical data. *Data & Knowledge Engineering* **63**(2), 503 – 527. ISSN 0169-023X. doi:https://doi.org/10.1016/j.datak.2007.03.016.

[13] Härtle, W. & Simar, L. (2003). *Applied Multivariate Statistical Analysis*, Springer Verlag, p 381. ISBN 3-540-03079-4.

[14] Kuttiyannan, D.T. (2009). Improved K-Modes for Categorical Clustering Using Weighted Dissimilarity Measure. *Computational Intelligence - CI* **5**.

[15] ming Cheung, Y. & Jia, H. (2013). Categorical-and-numerical-attribute data clustering based on a unified similarity metric without knowing cluster number. *Pattern Recognition* **46**(8), 2228 – 2238. ISSN 0031-3203. doi:https://doi.org/10.1016/j.patcog.2013.01.027.

[16] Ali, D., Ghoneim, A. & Saleh, M. (2017). Data Clustering Method based on Mixed Similarity Measures. pp 192–199. doi:10.5220/0006245601920199.

[17] Harispe, S., Ranwez, S., Janaqi, S. & Montmain, J. (2015). *Semantic Similarity from Natural Language and Ontology Analysis*, volume 8. doi:10.2200/S00639ED1V01Y201504HLT027.

[18] Bisson, G. (1995). Why and how to define a similarity measure for object based representation systems. *Towards Very Large Knowledge Bases* pp 236–246.

[19] Lofi, C. (2015). Measuring Semantic Similarity and Relatedness with Distributional and Knowledge-based Approaches. *Journal of Information Processing* **10**, 493–501.

[20] Rydzynski, G., Felic, A. & Zirpins, C. (2017). An ontology-based hierarchical clustering approach for decision support in mass customization environments. In *2017 2nd International Conference on Knowledge Engineering and Applications (ICKEA)*. pp 199–203. ISSN null. doi:10.1109/ICKEA.2017.8169929.

[21] Ehresmann, M., Skalden, J., Herdrich, G. & Fasoulas, S. (2018). Automated System Analysis and Design for Electric Propulsion Systems. In *Space Propulsion 2018*.

[22] Baumbach, N. (2020). Product Configuration Similarity Measures. doi:10.5281/zenodo.3930663.