Applying Semantic Correlation Rules in Technical Communication for Generating Logic-Driven Cross-References in Static Publications

Julian Staehle† and Wolfgang Ziegler‡

†Karlsruhe University of Applied Sciences, Faculty of Information Management and Media, 76133 Karlsruhe, Germany

Abstract. Authors in the field of technical communication are responsible for producing and enriching content with cross-references to create high-quality static publications. However, manual cross-reference generation can lead to inefficiencies, inconsistencies, and errors. This research explores the automation of cross-reference generation in the publication process by utilizing semantic correlation rules (SCR) to logically link content. The study tested a novel approach that separates cross-references from content and enriches them during publication. The method involves defining use cases, extracting structure, and enriching cross-references based on metadata. This is implemented using Python and semantic technology. The study highlights benefits such as enhanced consistency and functionality. Future work involves refining the methodology and exploring extensions to SCR, which offers broader applications beyond cross-reference generation.

1 Introduction

Technical Communication presents the challenge of creating complex publications with numerous cross-references. These references serve as orientation within the publication and link to external information products. In static publications, authors traditionally create these references manually and directly in the content. This could result in inefficiencies and inconsistencies, leading to potential errors.

The study explores the feasibility of generating cross-references automatically during the publication process, rather than creating them manually for each publication. The project already has a theoretically possible approach that requires testing and evaluation. Previous publications on Semantic Correlation Rules (SCR) have introduced the concept and technology of SCR as a logical layer between content production and content distribution [1, 2]. SCR allows for logical linking of content without the need for physical binding.

* Julian Staehle: staehlejulian@gmail.com
** Wolfgang Ziegler: wolfgang.ziegler@h-ka.de

© The Authors, published by EDP Sciences. This is an open access article distributed under the terms of the Creative Commons Attribution License 4.0 (https://creativecommons.org/licenses/by/4.0/).
2 Traditional Linking

Documents created with text processing programs or Desktop Publishing Tools (DTP) often include cross-references to aid orientation. Ideally, these cross-references should contain a hyperlink with the destination of a jump marker, which is typically located within the same document. It is best practice for authors to use the integrated function of the software to create cross-references, as this supports their creation, maintenance, and formatting.

![Cross-reference in Microsoft Word.](3)

Authors are responsible for the use and placement of references and links, even with the support of tools and interfaces. Creating cross-references and placing them manually in content can be challenging. Missing, misplaced, or inoperable cross-references can be the result. It is important to verify the presence, placement, and functionality of cross-references to ensure the quality of the content.

3 Linking in CCMS

In component content management systems (CCMS), content is created modularly and combined to publish documents. Cross-references are created between modules, and some systems offer additional support for authors, such as 'Abstract links' or 'Classification links'. These are used to create cross-references based on the classification of existing modules. The cross-reference is set to a target defined by the classification rather than a permanently assigned module ID.

Setting the jump label variably based on the metadata combination of the modules has advantages. For instance, the link does not need to be updated when modules are versioned. Even with existing module variance and use in different publication structures, the maintained links work without subsequent adjustment, provided the metadata concept is stable and the function is used correctly.

Despite the extended functionality, authors still face challenges with the placement and creation of cross-references. The integration of cross-references into the content makes it difficult to separate their creation from the content creation process. While quality assurance is supported, authors are responsible for maintenance. In addition, the use of the functions can be complex, requiring appropriately trained and qualified personnel.
4 Logic-Driven Approach

The core problem persists regardless of the systems used. The core problem persists regardless of the systems used. It involves manually enriching content with cross-references. The proposed solution in the research is to separate cross-references from the content and automatically enrich them during publication. According to a study, 73.3% of industrial companies surveyed use CCMS [4]. One way to implement this is by using a CCMS in combination with semantic technologies. The content is stored separately in the CCMS, and the logic regarding the modules to be linked can be managed in an ontology. Cross-references are maintained separately from the content, which should improve the quality of the modules and ultimately the publications in terms of consistency and functionality.

4.1 Procedure

The implementation of the outlined idea in a case study involves three steps: defining the use cases, extracting the structure, and enriching it with cross-references. Additionally, data must be exported from the CCMS, and the desired publication format must be generated. However, these two steps are not the focus of the study.
4.1.1 Use Case Definition

The use cases determine the logic of the cross-references. They are identified and mapped using SCR. In the case study, there are sequences of actions for securing a product in a vehicle environment. For some action sequences, certain settings or actions are recommended or even necessary. Therefore, a cross-reference between the action sequences and the settings is a derived set of rules. Another example involves procedures that require contacting customer service under certain conditions. The resulting rule establishes a connection between the potential procedures and the contact information.

It is important to note that the rules do not link modules by IDs with each other. This case study utilizes the PI-Class classification metadata concept to enrich the modules with metadata in a unique way [6]. For instance, the combination of metadata 'setting' and 'headrest' can identify the module applicable for the publication without requiring the direct knowledge of the module ID.

4.1.2 Extracting Structure

The CCMS contains the structure of the publication and the modules used within it. The structure and metadata of the modules can be extracted through an XML export and subsequent XSL transformation. This extracted information serves as the foundation for further procedures. It can be used to verify the metadata of individual modules to ensure their validity for certain SCR rules. The validity check is carried out by comparing sets of metadata using a programming language to iterate through each module of the structure.

Fig. 4. The module structure is shown.

4.1.3 Enriching Cross-References

To perform a validity check, compare the metadata tuples of the module (set A) and the SCR InRule (set B). If set B is a subset of set A (B ⊆ A), this indicates the starting point of a cross-reference. The target of the cross-reference is determined using the same procedure as the starting point. The SCR OutRule's metadata tuple represents set C, while a module's metadata tuple represents set D. If set C is a subset of set D (C ⊆ D), then this is the target of the cross-reference.

Both InRules and OutRules can contain any combination of metadata selectors. For example, a module can be enriched with product component metadata 'headrest' and specific task metadata 'adjust'. Suppose a separate module has the metadata 'headrest' in combination with 'clean'. Depending on the use case, it may be useful to query all modules enriched with 'headrest'. In this example, both modules are found because they both match the comparison.
4.2 Linking Mechanism

As already described, the linking approach is based on available modules and logical links at the meta level. The content is enriched with the corresponding cross-references by checking for the existence of logical links to the module metadata under consideration. Prerequisites are the classified modules and an ontology containing the module metadata. In addition, the ontology is extended by the basic definition of the SCR and the logical rules of the cross-references.

The technical implementation is done using the Python programming language and the Python library SPARQLWrapper. The ontology is available in the GraphDB software, which is installed in a local environment. The Python library SPARQLWrapper allows to access this local GraphDB instance and query information.

4.3 Product Extrinsic Metadata

The data (XML) is provided pre-filtered from the CCMS. The filter consists of the Product Extrinsic (P_ext) metadata, so the data is valid for a product variant selected during the export. This specificity of the case study means that it is not necessary to use the full functionality of SCR, such as the scr:equals relation, which describes the product context of the correlations.
4.4 Variant Characteristics

Variant characteristics have a direct impact on the generation of cross-references. They determine whether the validity between the basic metadata ("product intrinsic" and "information intrinsic") and the in-rule selects is really given. The background to this is that there are four possible constellations:

a) The source module has a variant property and points to a target module that does not have a variant property. In this case, no scr:equals relation may be set for the OutRule - the consequence would be that no matching target module is found, since no module with a matching variant feature exists.

b) The source module has a variant property and points to a target module with a variant property. The scr:equals relation must be used to ensure that the existing variance is directly valid. However, depending on the use case, it may be desirable to display the other modules as well, since the user may be in the wrong variance and want to switch to the correct constellation.

c) The start module does not have a variant characteristic and points to a target module without a variant characteristic. There is no variance, so scr:equals is not required.

d) The source module does not have a variant characteristic and points to a target module with a variant characteristic. There is no variance at the source, so scr:equals is not required. The cross-reference goes from the general to the specific and should show all variants. Ultimately, the reader must decide which path to take.

4.5 Lessons from the case study

Testing the model and its implementation in the case study had both positive and negative effects. On the positive side, the XML data exported from the CCMS is filtered. This means that the modules are already standardized in terms of product extrinsic validity. It is therefore not necessary to check the consistency of this metadata.

On the system side, the CCMS used in this case study does not have unique IDs for the variant characteristics. These are only available as plain text identifiers. Problems arise when identifiers exist more than once. If two identifiers with the same name from different variant drivers meet, the comparison leads to incorrect results. Since it was not possible to implement IDs in the case study, the taxonomy of the metadata was used to distinguish between the identifiers. The class can be queried using the rdfs:subClassOf property [7].

![Fig. 7. This graphic shows a section of the ontology. The focus is on the variant characteristics. There are two characteristics that have the same value '3 Point Belt'.](image_url)
5 Discussion

The majority of companies use CCMS to create modular content for document creation [4]. If an ontology is also available, the creation and maintenance of cross-references can be done on a logical level. This approach has advantages, limitations and possible extensions.

5.1 Separating content and cross-references

A clear advantage is the separation of logical links from content and the support it provides for authors. Authors can concentrate on their main task: creating content. In addition, the separation results in consistent placement and use of cross-references through automatic generation. There is a loss of manual placement, which needs to be discussed depending on the use case. Cross-references can only be generated at topic level. Cross-references at paragraph or even sentence level cannot be implemented in this release unless they are managed as identifiable objects.

5.2 Extending the SCR

It is conceivable to extend the SCR by subtypes of the relation scr:hasCorrelation [1]. This could affect the positioning of cross-references within modules or publications. If the cross-reference is intended to refer the reader to prerequisites or further content at the end of the module, this logic could be anchored in the SCR with an appropriate relation.

5.3 Media Neutrality

So far, the example of an HTML publication has shown that cross-references can be generated generically outside of content creation. However, this does not mean that this method is dependent on the medium to be generated. Other media formats, such as PDF, can also be generated and benefit from the advantages shown.

Fig. 8. Generated cross-references in an HTML publication.
5.4 Possible extensions

The generation has been tested, but it is possible to refine the order of the cross-references. In principle, functions such as grouping or sorting can optimize the cross-references used in the publication in terms of usability. A conditional application of these functions based on certain rules would also be interesting to investigate.

6 Conclusion and outlook

The implemented prototype has confirmed that it is possible to maintain cross-references on a logical level separated from the content. The framework consisting of XSLT, Python, SPARQL and SCR is suitable for testing purposes regarding the use cases to be defined and the implementation as a specific SCR rule set. The performance and maintenance of the SCR has already been mentioned in previous studies [1]. With reference to this, further investigations are useful for the implementation in a productive system in order to achieve the best possible performance and to track changes by means of versioning.

In addition, the archiving of SCR and the cross-references provided should not be ignored [8]. Best practices for the use of logical rules should be investigated with respect to the entire publication process in order to remove any implementation hurdles for interested users. The proposed enhancements could increase the likelihood of implementation. It is therefore planned to further refine the methodology and to test useful extensions. Of particular interest are the indicated subtypes of the scr:hasCorrelation relation, which would represent a useful extension of the SCR not only for cross-reference generation, but also for other areas of application of the SCR methodology.

7 References

4. D. Straub, W. Ziegler, Technical Communication on the Road to Digitization (tcworld, Stuttgart, Germany, 2023)