Assessment of EMF Model to Text Generation Strategies and Libraries in an Industrial Context

Christophe Ponsard, Denis Darquennes, Valery Ramon and Jean-Christophe Deprez
CETIC Research Centre, Charleroi, Belgium

Keywords: Model-based System Engineering, Model to Document, Tool Support, Industry Transfer, Eclipse Modelling Framework.

Abstract: Model-Based System Engineering is increasingly adopted. However it is centred on the notion of model while the industry is still strongly document-based. In order to enable a smooth evolution between those paradigms, the interplay between documents and models need to be managed, especially models can be used to efficiently derive up-to-date documents matching standard templates from initial requirements to final certification. The purpose of this paper is to review and to assess current strategies to manage model for document generators in order to meet industrial requirements like document complexity, document/model scalability, quick generation time, maintainability and evolvability of document templates. After exploring different generation strategies, our work focuses on toolchains based on the Eclipse Modelling Framework and compares a few shortlisted libraries in the light of our requirements and using document derived from industry cases.

1 INTRODUCTION

Because of its capability of abstraction/reasoning on real world artefacts before building them, modelling has been widely used across many engineering disciplines like construction, electronics or aeronautics. Model-Based System Engineering (MBSE) is a strong trend to apply modelling at system level and across disciplines. For the industry, it is however a paradigm shift because domain models become the central part for the exchange between engineers rather than documents. Model-Driven Engineering (MDE) is a similar trend focusing only on the software development process (Schmidt, 2006). Such approaches can rely on standardised and widely adopted modelling languages like SysML (OMG, 2005) at system level, UML (OMG, 1997) for software and increasingly Domain Specific Languages (DSLs). They provide a visual syntax enabling the design and communication activities of the engineers but also have precise semantics to enable automation of parts of the System Development Life Cycle (SDLC).

Being able to synchronise model and text is a key element for the industrial adoption of model-based toolchains because documents are and will certainly remain central to industrial development and validation workflows for a long time, even though the goal is to progressively reduce them. Documents will stay mandatory for the identification of requirements, review and validation at various SDLC steps and for specific activities like certification. This synchronisation involves different interactions between models and documents, like identifying concepts from documents, generating documents from models or keeping both artefacts synchronised over the system evolution. In our previous work, we have investigated different kinds of such interplays using diagrams, tables and texts (Ponsard et al., 2015) and for re synchronising models from documents (thus achieving rountrip with document generation) (Ponsard and Darimont, 2017).

This work focuses on model to text document generation. This may seem easy to achieve at first sight but requires to fulfil a number of key requirements, both functional and non-functional, in order to meet industry standards. Based on our experience with industrial customers, we could identify such requirements and also evaluate how currently available technologies fulfil them. Then, we review different document generation strategies with different possible trade-offs, e.g. favouring multiple output formats through an intermediary format/API or directed towards a specific format with more features and performance. Our work is more specifically addressing modelling solution based on the Eclipse Modelling Framework (EMF) because it is an open industrial standard and also because it fitted our industrial con-

Ponsard, C., Darquennes, D., Ramon, V. and Deprez, J.
Assessment of EMF Model to Text Generation Strategies and Libraries in an Industrial Context.
DOI: 10.5220/0009155804330440
In *Proceedings of the 8th International Conference on Model-Driven Engineering and Software Development (MODELSWARD 2020)*, pages 433-440
ISBN: 978-989-758-400-8; ISSN: 2184-4348

text. However, the same requirements and evaluation process may be reused for other modelling technologies, including closed ones. The last part of our work further investigates and compares a few solutions. It benchmarks prototypes against a test suite covering the target requirements and relying on real-world documents produced by industrial partners in two real-world and complementary toolchains, one Eclipse-based (Capella) (Polarsys/CIS, 2015) and the other web-based with a SaaS model (Respect-IT, 2005; Darimont et al., 2016). Note however that for confidentiality purposes excerpts presented are based on the public in-flight entertainment model (Bonnet, 2015).

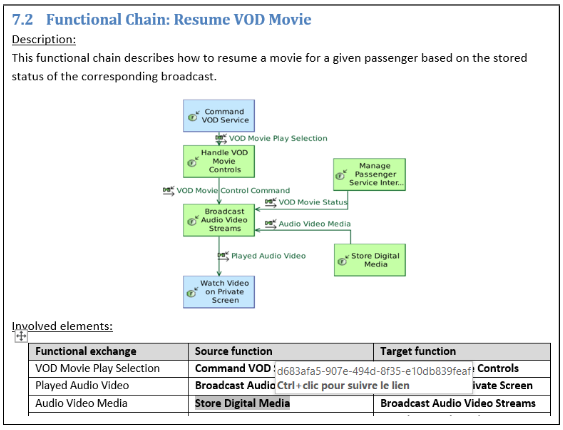
This paper is structured as follows. Section 2 reminds about key industry requirements. Then Section 3 surveys possible generation strategies based on both literature and industry practices. Section 4 details different candidate libraries focusing on direct generation for an EMF toolchain. Section 5 presents our benchmarking process, test data and raw results which are analysed in Section 6. Finally Section 7 concludes and discusses some complementary work.

2 INDUSTRY REQUIREMENTS FOR DOCUMENT GENERATION

In order to base our comparison on sound criteria, key industrial requirements have been gathered and grouped in two categories: functional and non-functional. Criteria identified in bold are evaluated later in this paper and are independent of any specific context (e.g. EMF).

2.1 Functional Requirements (FRs)

Industrial specifications are complex in nature and are characterised by a well-defined structure usually based on templates. An elegant way to introduce MBSE is to use **enriched templates** specifying how to populate the document, in the same format of existing corporate templates. The document generator needs to support a variety of **standard formats** used in the industry, typically Microsoft (docx) or LibreOffice (odt). Non editable formats like PDF and HTML may be useful too although easy to derive. Other target formats may also be interesting, e.g. spreadsheets or presentations. On the model side, our focus is the **support of Eclipse Modelling Framework** standard.



7.2 Functional Chain: Resume VOD Movie

Description:

This functional chain describes how to resume a movie for a given passenger based on the stored status of the corresponding broadcast.

Command VOD Service
VOD Movie Play Selection
Handle VOD Movie Controls
Manage Passenger Service Inter...
VOD Movie Control Command
VOD Movie Status
Broadcast Audio Video Streams
Audio Video Media
Played Audio Video
Store Digital Media
Watch Video on Private Screen

Involved elements:

Functional exchange	Source function	Target function
VOD Movie Play Selection	Command VOD d683afa5-907e-494d-8f35-e10db839feaf	Controls
Played Audio Video	Broadcast Audio Ctrl+clic pour suivre le lien	ivate Screen
Audio Video Media	Store Digital Media	Broadcast Audio Video Streams

Figure 1: Excerpt of a complex specification document mixing text, diagram and tables.

As shown in Figure 1, such documents also usually combine

- *text descriptions*, e.g. for requirements or function descriptions with specific writing conventions. This typically requires the use of **rich text** (as bold, italic, underline, bullets...).
- *figures*, where especially the **inclusion of diagrams** from the model should be easy, using standard formats (e.g. png, svg) and with a good quality.
- *tables*, to more systematically present well-structured concepts (e.g. requirements tables with identifier, definition, priority, effort...), system decomposition (subcomponents, interface components) or some specific view (e.g. traceability matrix). Although tables can be managed using spreadsheets, the direct **generation of tables** is mandatory.

The overall structure and consistency requires **automated section numbering** and **styles**. Other important features are the generation of **bookmarks** to ease the navigation in large documents. Table of contents are already managed by standard templates.

2.2 Non-Functional Requirements (NFRs)

The following classification is inspired from the SQuaRE classification (ISO, 2014) used as checklist, key NFRs are:

- *Costs* relates to a number of attributes. About acquisition costs, the library maybe **Free of charge** or even **Open Source**. However more important cost factors relate to the adaptation and maintenance of the solution which may require long term support. This can be evaluated by the global **Maturity** of the product (known bugs, documentation, tutorials...) and its support, either commercial or through a community.

- *Scalability* is a mandatory requirement. The size of industrial system means large model and thus the ability to **generate large documents** generally also based on **complex templates**.
- *Efficiency* is less critical in case of batch/server based generation. However, for laptop generation or when frequently update of a work document or developing a template, good **speed of generation** (i.e. under a minute) is important. Reducing the generation scope may also be used to accelerate modelling/generation cycles.
- *Usability* is important for industrial adoption. We should not assume the developer/maintainer of a generator template has a programming background and favour declarative and easy to learn tools. Two complementary levels need to be easy to use: how to retrieve information from the model and how to present that information in the document under the adequate form (see FR). This requires a **good learning curve** of the associated language (preferably standard like OCL), **good writing support** for both of the above levels and **good readability**, i.e. to maintain and evolve the generator template, preferably with some modularity feature. In order to invoke the tool, a **good user interface integration** with the modelling tool is required while a portable API (e.g. Java) is also useful for non UI invocation, e.g. for batch/nightly document regeneration.

3 MODEL TO DOCUMENT GENERATION STRATEGIES

Different strategies may be considered to produce a document from a model. In this section, we review the available design space and give some high-level guidelines that can help drive the choice based on typical trade-offs that need to be considered.

Template-based Generation. The most obvious choice is to use a single transformation tool able to convert a model into the target document format based on a transformation specified through some template which can take the form of a specific format (e.g. XML with a specific schema) or a Domain Specific Language (DSL). This simple design helps to give good control on the available features and performance. However, restricted choices may be available for the model source/target document format combination. If multiple document formats need to be considered, developing independent converters for each format is not a good idea because the template design will not be reused. An additional converter might be chained to transform the target document into other formats if the converter is reliable enough (e.g. to produce a PDF from most editable formats). In this context, we obviously restrict to Model-to-Text generation tools. This transformation can be managed by an independent library, e.g. Gendoc (Eclipse, 2018) or M2Doc (Obeo, 2019) for EMF) or directly inside a Word processor through internal macro/scripting. The latter is not recommended because the scripting language is not effective and maintainable for such transformation and for accessing a model repository. An editor is also not optimised for generation and should not be required at this step.

Generic Reporting Tools. Efficient reporting tools exist and are able to generate many different target document formats either editable (office documents but also spreadsheets or even presentations) or read-only (PDF). Birt (Eclipse Foundation, 2005) and Jasper (Jaspersoft, 2005) are examples of such platforms. They usually rely on an internal report description format that is populated from one or several data sources. The usual scenario is rather to generate business reports from spreadsheets or relational databases. However, data sources can be extended through additional drivers through a protocol such as the Open Data Access (ODA) which includes support for connecting, accessing meta-data, and executing queries. It is not restricted to classical relational databases and can also be used for model-based repositories. For example, an EMF connector is available and can be used together with Birt (Eclipse Foundation, 2008) with queries in OCL. The developed template can then be run in server mode. The performance might be impacted by the efficiency of the query engine. Alternatively, an intermediate data file in a specific format might also be produced from the model and fed more efficiently in the reporting tool.

Existing Modelling Platforms. Some modelling platforms integrate reporting capabilities and some also support model interchange, e.g. the MagicDraw tool can process EMF and also generate files using Velocity Template Library (Apache Foundation, 2006). However, this setting is hard to automate and reuse outside of this specific toolchain even though some parts might be open source.

Relying on an Intermediary Format. Is an interesting option because such a format can stay abstract enough to capture the essence (semantics) of the report without focusing on the concrete presentation. From that format, different target formats may be generated. Reporting tool template follows that principle but other options are possible and an open format should be favoured. For example, XML DocBook (Walsh and Hamilton, 2010) format can be processed using W3C Formatting Object (XSL-FO) and then to standard format such as PDF and RTF to some extend. Such an approach was applied by JPL on MagicDraw (Delp et al., 2013). Another more lightweight option is to use a variant of markdown language relying on plain text formatting syntax (Gruber, 2004). Based on this format, many concrete document formats can be generated using universal transformers such as Pandoc (MacFarlane, 2006). Of course, the final quality should be carefully assessed w.r.t. all required features described in Section 2. An interesting variant here is also to introduce a meta-model for the document itself which enables to use generally more powerful Model-to-Model transformation like the QVT Query/View/Transform standard (OMG, 2011) for dealing with the transformation with an additional serialisation to the textual output format (Willink, 2018). A more elaborated form of intermediary is to use a Domain Specific Language directly embedding the model transformation language. pxDoc detailed later is based on this approach (Pragmatic Modeling, 2019).

An open discussion point is whether the target format should be editable or not. While the latter ensures all changes are made inside the model, most of the time an editable format is preferred to enable integration, style adaptations and unfortunately some corrections which are then lost unless some synchronisation mechanism is used (Ponsard and Darimont, 2017).

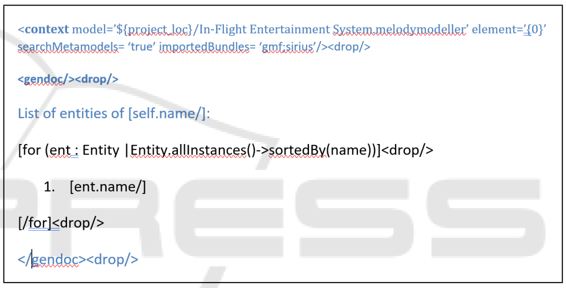
Finally note that although we tried to provide a consistent classification, the above categories are not mutually exclusive. For example, generic reporting tools also rely on a template. Exiting tools may use an intermediary format or use an external generation library. This cannot always be assessed (e.g. for closed tool). A key point guiding the classification design was the kind of interface and integration capability.

4 TEMPLATE-BASED GENERATION LIBRARIES

The rest of this document will focus on the single step generation strategy. Although it is less able to cope with multiple formats, it was selected for further study in both of our industrial cases. After discarding low-level generations libraries lacking any modelling support such as POI or VTL (Apache Foundation, 2006), we shortlisted three candidates which are presented hereafter. All of them rely on OCL (or variant) as model query language.

4.1 Gendoc

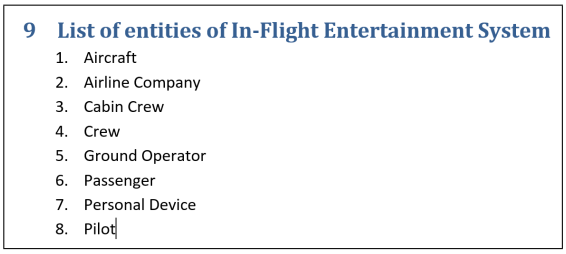
Gendoc (Eclipse, 2018) is an Open Source library developed by the Eclipse Foundation under its EPL license. It supports the generation of documents based on EMF models in different formats from Microsoft Office (docx, xslx, pptx) and Libre Office (.odt). Gendoc relies on existing templates for the general structure and styling while the content is specified using XML tags embedding commands for generating rich text, diagrams and tables. It strongly relies on the Acceleo Query Language (AQL), an OCL variant to dynamically retrieve content from the model. It is also compatible with the Papyrus (Eclipse, 2010) and Capella (Polarsys/CIS, 2015) modellers.



```
<context model='${project_loc}/In-Flight Entertainment System.melodymodeller' element='{0}'
searchMetamodels= 'true' importedBundles= 'gmf;sirius'/><drop/>

<gendoc/><drop/>

List of entities of [self.name/]:

[for (ent : Entity |Entity.allInstances()->sortedBy(name))]<drop/>

    1.   [ent.name/]

[/for]<drop/>

</gendoc><drop/>
```

Figure 2: Fragment of Gendoc template.

Figure 2 shows an example of template built using Gendoc. It is divided in two parts respectively marked by the *context* and *gendoc* tags. The *context* contains the absolute path to the model, the *searchMetamodel* and *importedBundles* information. The *gendoc* section contains a static text ”List of entities”, followed by a variable *self.name* which is the name of the model. A for loop lists inside the model the names of the entities resulting from the AQL request *allInstances()->sortedBy(name)*”. Figure 3 shows the result generated from this fragment.



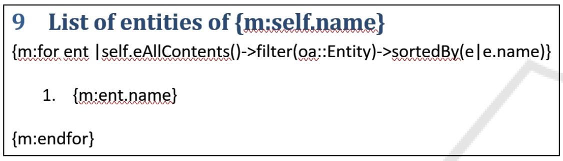
9 List of entities of In-Flight Entertainment System

1. Aircraft
2. Airline Company
3. Cabin Crew
4. Crew
5. Ground Operator
6. Passenger
7. Personal Device
8. Pilot

Figure 3: Result of the fragment of Gendoc template.

4.2 M2Doc

M2Doc (Obeo, 2019) is an Open Source library (EPL) developed by Obeo to generate documents from EMF models. Like Gendoc, it relies on the combination of a document template with AQL. The main differences are the limitation to Word documents (.docx) and the use of special fields of the form { *m: <content>* } rather than XML tags for retrieving the model elements, resulting in a better readability. M2Doc relies on ApachePOI for creating docx files. Figure 4 shows an example of fragment of M2Doc template functionally equivalent to the Gendoc fragment presented in Figure 2 and producing the same result as in Figure 3. The elaborated document presented in Figure 1 was also produced using M2Doc from the public in-flight entertainment model (Bonnet, 2015).



9 List of entities of {m:self.name}

{m:for ent |self.eAllContents()->filter(oa::Entity)->sortedBy(e|e.name)}

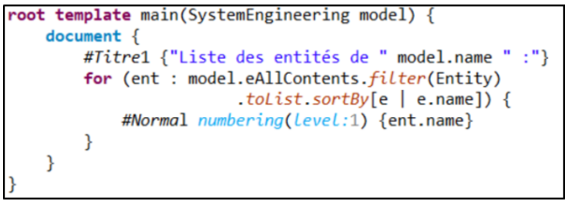
1. {m:ent.name}

{m:endfor}

Figure 4: Fragment of M2Doc template.

4.3 pxDoc

pxDoc (Pragmatic Modeling, 2019) is a commercial library used to generate Microsoft Word documents from EMF models. It requires an end-user license. Contrary to Gendoc and M2Doc, it does not extend an existing document template with queries to retrieve model content. Instead, a textual Domain Specific Language (DSL) is used for transforming the model into text and applying styles referenced from an existing and unmodified template. Figure 5 shows how to retrieve the same content than in the Gendoc and M2Doc sections, by using model navigation expression in a format closed to OCL and AQL queries. Styling is applied using styling reference like *#Title1* and specific command like *numbering* to control the numbering level. The DSL is implemented with EMF and XText technologies that are completely integrated with the Eclipse development environment and Eclipse-based modellers like Papyrus (Eclipse, 2010) and Capella (Polarsys/CIS, 2015).



```
root template main(SystemEngineering model) {
    document {
        #Titre1 {"Liste des entités de " model.name " :"}
        for (ent : model.eAllContents.filter(Entity)
                        .toList.sortBy[e | e.name]) {
            #Normal numbering(level:1) {ent.name}
        }
    }
}
```

Figure 5: Fragment of pxDoc template.

Compared to GenDoc and M2Doc, pxDoc is closer to coding and has a direct integration with Java and Eclipse. The DSL looks like a simplified form of Java and is not complex to learn. The DSL editor provides productivity features like syntax checking and content assist, including to navigate into the target model when writing a query. This greatly accelerates the development of a document generator. It also does not require Word in this phase.

5 ASSESSMENT OF SELECTED GENERATION LIBRARIES

This section presents our assessment of the libraries shortlisted in Section 4. Our assessment was based on prototyping generation of realistic industrial documents without prior knowledge of the libraries. For the models and templates, we used the publicly available in-flight entertainment system (Bonnet, 2015) and models provided by two industrial partners: one producing large design documents for railways systems and the other producing very elaborated requirements documents for large call for tenders.

Table 1 and 2 present the assessment respectively for the functional and non-functional criteria defined in Section 2. Criteria are ranked either on a binary

Table 1: Comparison of selected libraries for the functional criteria.

	Gendoc	M2Doc	pxDoc
EMF support	x	x	x
Standard formats supported	docx, odt, xlsx, pptx	docx	docx
Use of enriched templates	x	x	-
Rich text inclusion	x	-	x
Diagram inclusion	x	x	x
Table generation	x	x	x
Section numbering	x	x	x
Style support	x	x	x
Bookmark/hyperlink generation	x	x	x

Table 2: Comparison of selected libraries for the non-functional criteria.

	Gendoc	M2Doc	pxDoc
Cost			
Free of charge	x	x	-
Open Source	x	x	x
Maturity	++	+	++
Scalability	+	++	+++
Efficiency			
Large document	+	++	+++
Complex template	+	++	+++
Usability			
Learning curve	+	+	-
Writing support	+	++	++
Readability	+	+++	++
Interface integration	++	+	++

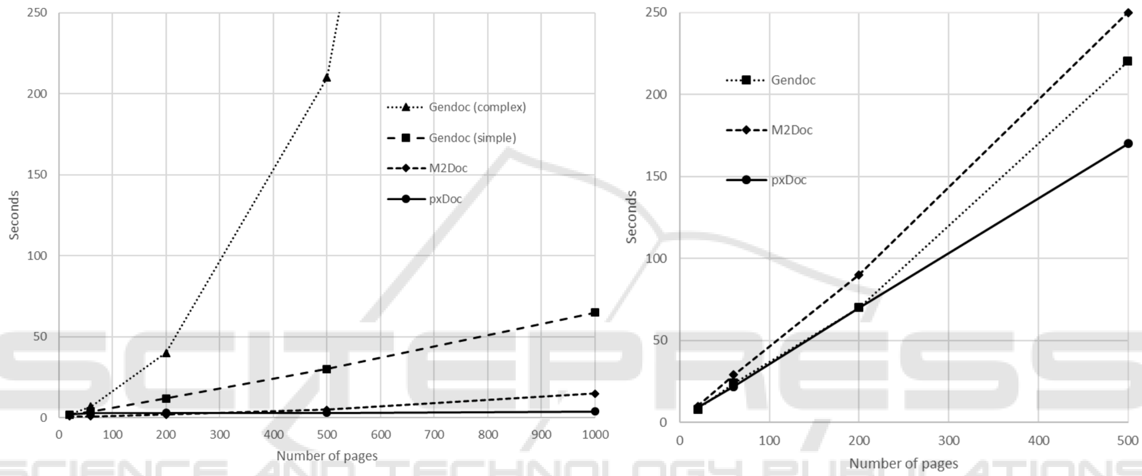


Figure 6: Comparison of generation time for pure text document (left) and documents with figure (right).

scale (fulfilled=”x” and not fulfilled=”-”) or on a qualitative scale (from ”—” for very poor to ”+++” for very good). For the scalability criteria, quantitative benchmarks were run but are reported on a qualitative scale with some additional comments. Note that due to the preselection, the global ranking is rather good.

Table 1 shows that three preselected libraries provide good coverage of all functional criteria. A weaker point for M2Doc is the current lack of support for rich text import. Gendoc is the only to support the Open Source odt format. Table 2 shows more contrasted results for the non-functional criteria that are further discussed in Section 6.

About efficiency, our prototypes were benchmarked using the same models to produce large documents (up to 1000 pages) on a standard laptop (Core I7 2.8GHz with 16GB RAM running Windows 10). We considered simple templates (mostly section, subsections and lists) and more complex templates (including figures and tables resulting from complex queries). Figure 6 summarises our comparison for Gendoc (V0.7.2), M2DOC (V2.0.2) and pxDoc (V1.1.1):

- for pure text documents, the generation time remains acceptable in all cases: less than 1 minute
- globally pxDoc is faster, probably due to the overhead for interpreting AQL in M2Doc and Gendoc.
- Gendoc is slower, especially for generating large documents with a complex template. For 1000 pages, it did not crash but took 15 minutes. For M2Doc and pxDoc template complexity has a limited impact resulting in the same curves.
- when considering figures, the results are slower and more balanced: between 3 and 4 minutes for 500 pages. This is probably due to the similar figure transfer time from Capella to the document. Gendoc is slightly better than M2Doc here.

6 DISCUSSION: WHAT TO USE AND IN WHICH CONTEXT ?

We provide here some guidelines to help in the selection of libraries within the scope of the libraries studied in Section 3. Section 2 gives more general guidelines about other possible strategies but without digging in concrete solutions. More detailed guidelines are out of the scope of this paper but part of our future work.

Globally, pxdoc is quite different because it does not extend existing templates but it requires a licence and to learn a DSL. However it provides very effective tool support for writing templates. The solution can be suitable for a big company which can afford a license (e.g. a few floating might be enough) and with usually quite stable templates and a dedicated trained person in charge of the corporate templates.

Gendoc is a good solution for not too overly large documents. However, XML tags are not easy to read and also reduce the maintainability. The direct support for odt is a strong point if this is a standard format adopted by the company. It allows a fully Open Source toolchain. The long term support needs to be monitored as the project does not seem very active.

M2Doc is a more recent solution which still lack some features like rich text and adapted if the focus is on the docx format only. It is quite efficient and easy to learn and maintain due to the lighter form of embedding. The use of a docx template requires careful writing and a few cycle of debugging which is decently supported by errors reported in a validation document or directly inside the generated document. The library is still actively evolving, although the last version is quite behind schedule.

About the dependency over Eclipse, M2Doc is quite easy to isolate and run outside of Eclipse, e.g. to develop a web-based tool. On the contrary, Gendoc has intricate Eclipse dependencies difficult to isolate. A workaround lacking efficiency and robustness is to run Gendoc as console-mode Eclipse service.

7 CONCLUSION AND FUTURE WORK

In this paper, we stated key industrial criteria for model-based document generation. Based on those criteria, we first identified and discussed possible document generation strategies with different benefits and drawbacks. Then, we focused on a direct generation strategy and assessed the best libraries for this purpose and an EMF modelling context. As a result, we produced guidelines that we believe useful for driving the selection of a generation solution in an EMF context and, more generally, based on the identified criteria and assessment process.

In our future work, we plan to extend our benchmarking to compare different strategies especially with a generic reporting tool and an intermediary markdown format. Different quality attributes will be quantified in order to help selecting the best approach given a specific industrial context. We will also keep monitoring the evolution of the identified libraries and also study new ones like Papyrus Model2Doc which is announced (Lorenzo, 2019). Based on this we could provide more extensive guidelines to help in the decision process of a document generation solution in the context of a set of specific requirements among those we identified.

ACKNOWLEDGEMENTS

This research was partly supported by the IDEES Co-innovation project (nr. ETR121200001379) of the Walloon Region. We thank Respect-IT for its industrial cases and feedback.

REFERENCES

Apache Foundation (2006). Velocity Template Library. https://velocity.apache.org/engine/1.7/vtl-reference.html.

Bonnet, S. (2015). In-flight entertainment system model. https://wiki.polarsys.org/Capella/Samples/IFE.

Darimont, R., Zhao, W., Ponsard, C., and Michot, A. (2016). A modular requirements engineering framework for web-based toolchain integration. In *24th IEEE Int. Req. Eng. Conf., Beijing, China, Sept.*

Delp, C., Lam, D., Fosse, E., and Lee, C.-Y. (2013). Model based document and report generation for systems engineering. In *IEEE Aerospace Conference*.

Eclipse (2010). Papyrus Modeler. https://www.eclipse.org/papyrus.

Eclipse (2018). Gendoc - Generation of Documentation from EMF models using document templates. https://www.eclipse.org/gendoc.

Eclipse Foundation (2005). Business Intelligence and Reporting Tools. https://www.eclipse.org/birt.

Eclipse Foundation (2008). Open data access driver. https://wiki.eclipse.org/Ecore_ODA_Driver.

Gruber, M. (2004). Markdown. https://daringfireball.net/projects/markdown.

ISO (2014). 25000 systems and software quality requirements and evaluation (square). https://www.iso.org/standard/64764.html.

Jaspersoft (2005). Jasper Reports. https://community.jaspersoft.com/project/jasperreports-library.

Lorenzo, V. (2019). EclipseCON, Munich (accepted).

MacFarlane, J. (2006). Pandoc - a universal document converter. https://pandoc.org.

Obeo (2019). M2Doc - generation of Office Open XML documents from EMF models and document template. http://www.m2doc.org/ref-doc/2.0.2/.

OMG (1997). Unified modeling language. http://www.omg.org/spec/UML.

OMG (2005). System modeling language. http://www.omg.org/spec/SysML.

OMG (2011). MOF Query/View/Transformation. https://www.omg.org/spec/QVT/1.1.

Polarsys/CIS (2015). Capella - open source solution for model-based systems engineering. https://www.polarsys.org/capella.

Ponsard, C. and Darimont, R. (2017). Improving Requirements Engineering through Goal-oriented Models and Tools: Feedback from a Large Industrial Deployment. In *Proc. 12th Int. Conf. on Soft. Tech., ICSOFT, Madrid, Spain, July 24-26*.

Ponsard, C., Darimont, R., and Michot, A. (2015). Combining Models, Diagrams and Tables for Efficient Requirements Engineering: Lessons Learned from the Industry. In *INFORSID 2015, Biarritz, France*.

Pragmatic Modeling (2019). pxDoc - generate Word documents from Java. https://www.pxdoc.fr.

Respect-IT (2005). The Objectiver Goal-Oriented Requirements Engineering Tool. http://www.objectiver.com.

Schmidt, D. C. (2006). Guest editor's introduction: Model-driven engineering. *Computer*, 39(2):25–31.

Walsh, N. and Hamilton, R. L. (2010). *DocBook 5: The Definitive Guide*. O'Reilly, Sebastopol, CA.

Willink, E. D. (2018). A text model - use your favourite M2M for M2T. In *Proceedings of MODELS 2018 Workshops, Copenhagen, Denmark, October*.