RM2Doc: A Tool for Automatic Generation of Requirements Documents from Requirements Models

Tianshu Bao
School of Computer Science and Technology
Guizhou University
Guiyang, China

Yilong Yang
School of Software
Beihang University
Beijing, China

Jing Yang
School of Computer Science and Technology
Guizhou University
Guiyang, China

Yongfeng Yin
School of Software
Beihang University
Beijing, China

ABSTRACT

Automatic generation of requirements documents is an essential feature of the model-driven CASE tools such as UML and SysML designers. However, the quality of the generated documents from the current tools highly depends on the attached descriptions of models but not the quality of the model itself. Besides, if the stockholders ask to generate ISO/IEC/IEEE 29148-2018 conformed documents, extra templates are required. In this paper, we propose a CASE tool named RM2Doc, which can automatically generate ISO/IEC/IEEE 29148-2018 conformed requirements documents from UML models without any templates. In addition, the flow description can be generated from a use case without additional information. Moreover, it can automatically generate the semantic description of system operations only based on the formal expression of OCL. We have conducted four case studies with over 50 use cases. Overall, the result is satisfactory: The 95% requirements documents can be generated from the requirements model without any human interactions in 1 second. The proposed tools can be further developed for the industry of software engineering.

The tool can be downloaded at http://rm2pt.com/rm2doc, and a demo video casting its features is at https://youtu.be/4z0Z5mrLfBc

KEYWORDS

Automatic Documentation, Requirements, Requirements Model, Requirements Documents

1 INTRODUCTION

In requirements engineering, it is widely accepted that models are the preferred method of capturing and communicating requirements. Models are more formal and semantically precise than natural language, which helps developers to specify and understand requirements. However, the importance of documents should not be overlooked, as models cannot be read directly by non-specialists. In order to avoid problems in later stages, the model should be verified (to be sure developers “developed the requirements right”) and validated (to be sure developers “developed the right requirements”). The verification of requirements can be achieved by automated methods[2][9]. But validation requires agreement between the developers and the stakeholders on what the system is intended to achieve[3]. A common challenge in this context is the communication gap between stakeholders and developers[7]. The communication gap between them makes stakeholders unable to participate in model validation well. Therefore, requirements documents are still required for communication and review with stakeholders.

Generally, requirements documents are written manually, which is a difficult and complex task that requires a great deal of effort. This is also a factor in the lack of (good) documents for many models. Even if documents were created at an early stage, as the change of requirements and the improvement of models, the problem of inconsistencies between the model and the document becomes more and more pronounced. To solve these problems, we present RM2Doc: a tool for automatic documents generation from a requirements model. Compared with other CASE tools, the benefits of our tool are as follows:


2) Automatic generation of the flow description from a use case without additional information.

3) Automatical generation the semantic description of system operations only based on the formal expression of OCL.

The remainder of this paper is organized as follows: Section 2 presents the overview of RM2Doc. Section 3 presents the evaluation results on the four case studies. Section 4 and 5 discuss the related tools and conclude this paper.
2 OVERVIEW

The architecture of RM2Doc is shown in Figure 2. RM2Doc takes a requirements model as input and generates a requirements document describing that model. In this section, we first introduce our requirements modeling. Then, RM2Doc has two critical features to help stakeholders validate requirements: the first is to generate operations description from the contracts of system operations and the second is to generate the requirements document for the requirements model.

The contract of a system operation specifies the conditions that the state of the system is assumed to satisfy before the execution of the system operation, called the pre-condition and the conditions that the system state is required to satisfy after the execution (if it terminated), called the post-condition of the system operation. Figure 1 (c) shows the OCL contract of system operation enterItem. As shown in 1 (c), in our modeling approach, a contract consists of four parts: the signature, the definition section, the precondition section and the post-condition section:

Signature: The signature first specifies the name of the system operation and the name of the use case to which it belongs. The signature then declares the input parameters and return type of the system operation.

Definition Section: In the definition section, the objects used jointly by the precondition section and the post-condition section are defined.

Precondition Section: The precondition specifies the properties of the system state that need to be checked when system operation is to be executed. In addition to the checking of objects and attributes shown in Figure 1 (c), the precondition also includes the checking of links between objects.

Post-condition Section: The post-condition defines the possible changes that the execution of the system operation is to realize. In addition to creating and adding objects, adding links between objects, and modifying the attributes of objects as shown in Figure 2, the post-conditions include the deletion of objects and the removal of links between objects.

2.2 Generation of Operations Description

A system operation is an action performed by the system in response to an external system input event. From the stakeholder’s point of view, system operations define the functionality of the system. Therefore generating a description of the system operation
can help the stakeholder understand the system’s functionality. In this subsection, we describe how RM2Doc translates the definition section, the precondition section and the post-condition section of a contract into natural language.

As described in subsection 2.1, the definition section defines the objects. The precondition section checks the objects, their attributes and the links between objects. The post-condition section includes creation, addition, and deletion of objects, modification of attributes, and addition and deletion of links. For these different operations, we defined a total of 25 transformation rules for the three sections. Transformation rules are presented in this form:

\[
\text{Rule: } \text{OCL Expression} \quad \text{Natural Language}
\]

The transformation rule contains two parts: the above section is an OCL expression in the contracts, and the bottom part is the corresponding natural language.

Algorithm 1 shows how to apply the rules to translate a contract into natural language.

Algorithm 1: Transformation Algorithm

**Input:** Contract

**Output:** Natural Language

\[
\begin{align*}
\text{def} & \leftarrow \text{parseDefinition}(	ext{Contract}); \\
\text{pre} & \leftarrow \text{parsePrecondition}(	ext{Contract}); \\
\text{post} & \leftarrow \text{parsePostcondition}(	ext{Contract}); \\
\text{for } & \text{exp } \in \text{def} \text{ do} \\
\text{num} & \leftarrow 0; \\
\text{num} & \leftarrow \text{matchRule1to6}(\text{s}) \\
\text{nl} & \leftarrow \text{translate}((\text{exp}, \text{num}) \\
\text{nl} & \text{.append}(\text{nl}) \\
\text{end}
\end{align*}
\]

\[
\begin{align*}
\text{for } & \text{exp } \in \text{pre} \text{ do} \\
\text{num} & \leftarrow 0; \\
\text{num} & \leftarrow \text{matchRule7to14}(\text{s}) \\
\text{nl} & \leftarrow \text{translate}((\text{exp}, \text{num}) \\
\text{nl} & \text{.append}(\text{nl}) \\
\text{end}
\end{align*}
\]

\[
\begin{align*}
\text{for } & \text{exp } \in \text{post} \text{ do} \\
\text{num} & \leftarrow 0; \\
\text{num} & \leftarrow \text{matchRule15to25}(\text{s}) \\
\text{nl} & \leftarrow \text{translate}((\text{exp}, \text{num}) \\
\text{nl} & \text{.append}(\text{nl}) \\
\text{end}
\end{align*}
\]

\[
\begin{align*}
\text{return } \text{nl};
\end{align*}
\]
By using these 5 Xtend templates, RM2Doc can generate a requirements document from a requirements model. In the rest of the requirements document, we have predefined guidelines for writing in accordance with the ISO/IEC/IEEE 29148-2018.

3 EVALUATION

We will introduce case studies first, then propose the evaluation results based on the case studies.

3.1 Case Studies

In order to evaluate the RM2Doc, we reuse four case studies from the paper [12]. Those case studies are widely used systems in our daily life: Supermarket System (CoCoME), Library Management System (LibMS), Automated Teller Machine (ATM), and Loan Processing System (LoanPS). More details of the requirements models can be found at GitHub\(^1\). The complexity of those requirements models are shown in Table 1, which totally contains 17 actors, 51 use cases, 6 system sequence diagrams, 138 system operations, 38 entity classes, and 49 associations of entity classes.

Table 1: The Complexity of Requirements Models

<table>
<thead>
<tr>
<th>Case Study</th>
<th>Actor</th>
<th>Use Case</th>
<th>SSD</th>
<th>SO</th>
<th>Entity Class</th>
<th>Association</th>
</tr>
</thead>
<tbody>
<tr>
<td>ATM</td>
<td>2</td>
<td>6</td>
<td>3</td>
<td>15</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>CoCoME</td>
<td>3</td>
<td>16</td>
<td>1</td>
<td>43</td>
<td>13</td>
<td>20</td>
</tr>
<tr>
<td>LibMS</td>
<td>7</td>
<td>19</td>
<td>0</td>
<td>45</td>
<td>11</td>
<td>17</td>
</tr>
<tr>
<td>LoanPS</td>
<td>5</td>
<td>10</td>
<td>2</td>
<td>35</td>
<td>12</td>
<td>8</td>
</tr>
<tr>
<td>Sum</td>
<td>17</td>
<td>51</td>
<td>6</td>
<td>138</td>
<td>38</td>
<td>49</td>
</tr>
</tbody>
</table>

1 SSD is the abbreviation of system sequence diagram. SO is the abbreviations of system operations.

1https://github.com/RM2PT/CaseStudies

To illustrate the capabilities of RM2Doc, this section demonstrates the generation of a requirements document from a requirements model. We use CoCoME as an example. The use case diagram in Figure 1 (a), contains 3 participants and 16 use cases. RM2Doc generates the "Product functions" and the "User characteristics" of the document, as shown in Figure 6 (a).

Considering the system sequence diagram and the use case diagram in Figure 1, RM2Doc can generate a use case description for the use case processSale, as shown in Figure 6 (b). Figure 6 (c) presents the contract of enterItem, which is a system operation in Figure 1 (b). According to the contract, RM2Doc generates a description of the system operation enterItem, as shown in Figure 6 (d). In total, 16 use case descriptions and 43 system operation descriptions are generated, and they form the "Functional requirements" of the document. Based on CoCoME’s conceptual class diagram, as shown in Figure 1 (d), RM2Doc generates the analysis of 13 entities, which are then organized into the "Database requirements" of the document. The Figure 6 (d) shows the generation result of the entity Item.

3.2 Evaluation Results

Table 2: The Generation Result of Operations Description

<table>
<thead>
<tr>
<th>Case Study</th>
<th>NumSO</th>
<th>MSuccess</th>
<th>GenSuccess</th>
<th>SuccessRate (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ATM</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>100</td>
</tr>
<tr>
<td>CoCoME</td>
<td>43</td>
<td>42</td>
<td>41</td>
<td>95.34</td>
</tr>
<tr>
<td>LibMS</td>
<td>45</td>
<td>44</td>
<td>42</td>
<td>93.33</td>
</tr>
<tr>
<td>LoanPS</td>
<td>35</td>
<td>32</td>
<td>32</td>
<td>91.42</td>
</tr>
<tr>
<td>Total</td>
<td>138</td>
<td>133</td>
<td>130</td>
<td>95.02</td>
</tr>
</tbody>
</table>

1 MSuccess is the number of SO which is modeled correctly without external event-call, GenSuccess is the number of SO which is successfully generated, SuccessRate = GenSuccess / NumSO.
Correctness of generation. For the correctness of the generation, we consider two aspects. 1) For the generation of UML diagram results, 17 actors, 51 use cases, 6 system sequence diagrams, 38 entity classes, and 49 associations of entity classes were all generated correctly. 2) As for the OCL contracts, the results of 138 system operations are shown in Table 2, of which, 133 system operations can be modeled correctly. And 130 of the 133 can generate semantic descriptions correctly. The average success rate of the four cases was 95.02%.

Time performance. The main application scenario for RM2Doc's documents generation is to provide natural language reference for stakeholders. In addition, our generation can also provide developers with alternative views of the model. In the latter case, the generation must be fast enough[7]. The experimental settings of RM2Doc are 2.8 GHz Intel Core i5, 8 GB DDR3, and JDK 11. Generation for all 4 case studies spend less than 1 second, we consider that this speed meets our expectations.

4 RELATED TOOLS

The tools closely related to RM2Doc can be divided into two categories according to the results generated.

The tools in the first category, which generate natural language requirements from UML models, are as follows. TESSI[6] generates natural language text for describing the model based on the UML model. GeNLangUML[8] and KeY[1] use UML class diagrams as input to generate natural language descriptions of class diagrams. In addition, there has been some work[3][4] dedicated to extracting well-structured business vocabulary and business rules from use case diagrams and class diagrams. These tools use a single type of model, and the results are mostly a set of disordered sentences or not well-organised. It is very difficult to find valuable content from such results, and the application value is low.

The second type of tool is focused on generating the document from software models. Wang et al.[10] propose a tool that generates software requirement specifications, preliminary design specifications and detailed design specifications from UML models and SmartC models. DocGen[5] generates a report from SysML models. The modelling tools MagicDraw[2] and Modelio[3] have an integrated documents generation function. In addition, some commercial tools such as M2Doc[4], GenDoc[5] and pxDoc[6] also support the generation of structural documents from models. Such tools can generate the organisation of the document, but 1) the content in the document relies too much on the natural language descriptions that come with the model, rather than on natural language generation techniques. 2) Most tools require a template for the document. Some templates are quite complex to produce and for the novice this can be costly. 3) All these tools do not include the handling of OCL, which is a complement to UML.

Compared to the above tools, RM2Doc 1) can generate well-structured documents without additional templates from the user; 2) The natural language in the document does not only rely on the natural language descriptions that come with the model, but can also be generated by the model, e.g., a flow of a use case can be generated from the system sequence diagram of the use case. 3) Our tool can also translate the preconditions and post-conditions of system operations into natural language.

5 CONCLUSION

This paper presents the RM2Doc tool. The tool generates requirements documents from the requirements model and improves the productivity of requirements documents. The generated requirements document helps the stakeholders to perform requirements validation. Four case studies evaluated the capability of RM2Doc.

In the future, we will make some improvements to make our tools more useful. For example, we plan use deep learning to generate more flexible and more readable natural language from requirements model.

ACKNOWLEDGMENTS

This work was supported by National Science Foundation of China Grant NO. 61732019, and the Opening Project of Shanghai Trusted Industirll Control Platform No. KH54327801

REFERENCES