Improving Requirement Boilerplates Using Sequential Pattern Mining

Maxime Warnier, Anne Condamines

Improving Requirement Boilerplates Using Sequential Pattern Mining

Maxime Warnier[1,2] and Anne Condamines[1]

[1] CLLE-ERSS (Université Toulouse – Jean Jaurès & CNRS), Toulouse, France
[2] Centre National d'Études Spatiales, France
{maxime.warnier, anne.condamines}@univ-tlse2.fr

Abstract. In the field of requirements engineering, the use of the so-called *boilerplates* (i.e. standard phrases and sentences containing gaps to be filled in) is a popular solution to reduce variation among requirements and writers, and thus to improve the clarity of technical specifications. However, the examples of boilerplates found in the literature are often very general, as they need to be applicable to projects as varied as computer software and aircraft or industrial machines. As a result, they only partially fulfill their role, leaving a lot of freedom to the writers in charge of filling in the gaps. Instead, we would like to propose a bottom-up approach for discovering more specific sequences that could constitute either boilerplates or elements to instantiate these boilerplates. To this end, we investigate whether sequential data mining techniques can be used on a small corpus of genuine requirements written in French at CNES (Centre National d'Études Spatiales), the French Space Agency.

Keywords: pattern mining, boilerplates, requirements, corpus linguistics, phraseology.

1 Introduction

This study is part of a research project that aims at improving requirements writing at CNES (Centre National d'Études Spatiales) – the French Space Agency –, where technical specifications are currently written mostly in French. Unfortunately, natural languages (such as French or English) are known to be inherently ambiguous [1] and vague [2]. While these properties rarely have adverse consequences in everyday communication, they become serious threats when writing and reading requirements[1], as comprehension is crucial; indeed, a misunderstanding may lead to non-compliance with the requirements, which in turn can result in delays, additional costs, litigation (because requirements represent a legal obligation for stakeholders) or in the worst case accidents. This explains why requirements engineering has become so important

[1] *requirement*: "a condition or capability that must be met or possessed by a system or system component to satisfy a contract, standard, specification, or other formally imposed documents." [3] A *specification* is a collection of requirements.

in the last decades [4] and why so many solutions have been proposed to prevent or at least to limit this risk.

One of those solutions is what is now commonly called a Controlled Natural Language (CNL), that is, "a constructed language that is based on a certain natural language, being more restrictive concerning lexicon, syntax, and/or semantics, while preserving most of its natural properties" [5]. There is a wide variety of CNLs, used in many different fields; some of them remain quite similar to their "base" language, while others are close to formal notations (such as programming languages or even mathematical notations [6]), each with advantages and disadvantages. Our goal is to propose a CNL perfectly adapted to requirements writing (in French) at CNES. In other words, we want it to be as close as possible to what engineers are already used to write and read, because we are convinced that if we want our solution to be applied and to be efficient, it should be tailored and not too disruptive.

To achieve this, we propose a methodology for creating CNLs for technical writing that takes into account (1) existing CNL rules and principles – as they may be viewed as the state of the art in the field – and (2) actual samples of writing used by the members of the discourse community [7] (in our case, requirements written in unconstrained natural language for past projects) that will serve as a basis for a corpus analysis [8]. This methodology combines a corpus-based and a corpus-driven approach [9-10]: in the former case, our linguistic hypotheses (based in particular on recurring prescriptions found in other CNLs) are tested on the corpus, whereas in the latter case, linguistic regularities themselves emerge from the corpus.

These regularities (which constitute part of the grammar of the textual genre [11]) – if they exist – can then be considered when defining (or adapting) rules and when proposing standard requirements. Therefore, although the CNL we wish to create is (necessarily) normative, in our opinion it is essential that it is preceded by a descriptive analysis of the final users' language practices; this should insure that it really meets their needs – which would not be the case if it was too general.

Such an analysis looks at recurring sequences of multiple words (or categories of words) – and at their meaning – that can be given different names, depending on their status but also on the point of view (and goals) of the researcher. Here, we will simply call them *(textual) patterns* [12-13-14] and emphasize that they should be specific to a (sub)genre.

A feature of our corpus is that it includes terminological and phraseological units that may belong either to the textual genre (in this case, requirements) or to the technical domain (in this case, space projects); this may be a challenge if one tries to distinguish between the two using automated tools. Since we are interested mostly in the former (we want to describe requirements writing at CNES, regardless of the project), we suggested a solution to filter the results of a terminological extraction [15].

In this paper, we focus on the extraction of frequent patterns that could then be recommended to the engineers who are in charge of writing requirements. In Section 2, we introduce the notion of boilerplate in requirements engineering and we describe our method for suggesting new ones; in Section 3, we briefly comment on some of the results we obtain when we apply the method on our corpus of requirements; finally, we conclude in Section 4.

2 Aim and Method

2.1 Requirement boilerplates

The aim of this study (which is part of the corpus-driven approach of our global methodology) is to analyze our corpus of requirements in order to semi-automatically identify frequent patterns that are possible candidates for boilerplates or fixed elements to be used in future CNES projects (currently, engineers do not use boilerplates).

The term *boilerplate* was first used fifteen years ago in requirements engineering[2] to refer to a textual requirement template. A very simple example of boilerplate is given by "<system> shall <action>", where "shall" is a fixed element and "<system>" and "<action>" are attributes that will be replaced by textual values in actual requirements [16] (e.g. "The vehicle shall respect the speed limit"). They are not necessarily full sentences: some boilerplates are clauses or even phrases that can be added at the beginning or at the end of other boilerplates; all of them are basically more or less complex grammatical (non-discursive) structures.

The concept of requirement boilerplate (or requirement template) is appealing because it is very simple to understand (fill-in-the-gaps texts, frames and similar concepts existed long before requirements[3]) and to use, and yet it is definitely useful when the same kind of needs must be expressed repeatedly, as is often the case in large projects with thousands of requirements (sometimes with only minor variations); this is especially true when using requirements authoring and management tools (IBM Rational DOORS, for instance) that allow to handle boilerplates easily.

It can be seen as a semi-formal solution for writing requirements: texts written according to a collection of boilerplates look exactly like those written in natural language (since in theory, all sentences are grammatically correct and can be understood by people familiar with the terms without any prior training) – except that it is obviously much more repetitive –, but at the same time, they are more formal because they allow less expressivity, which is exactly why they are thought to reduce ambiguity. At the very least, it can reasonably be assumed that they should increase consistency among specifications: although variation is unavoidable in oral and written natural language (because the same need can be expressed by different words and sentences), requirements are usually quite short written texts that should be readable independently and where repetition is not only acceptable, but recommended.

Nevertheless, even though existing boilerplates may be considered convenient and easy to use, the crucial difficulty remains of course to define a set of boilerplates that will be proposed (or possibly imposed) to the writers and that preferably best suit their needs. It appears that the boilerplates[4] suggested by the CESAR (Cost-efficient methods and processes for safety relevant embedded systems) project [17], which are one of the semi-formal "Requirement Specification Languages", are often regarded as

---

[2] It is also used with a similar meaning in computer science ("boilerplate code") and contractual law ("boilerplate language": parts of a contract that are considered standard) [16].

[3] In a sense, it is close to the copy-paste function of all modern computers.

[4] Defined as "semi-complete requirements that are parameterized to suit a particular context".

a reference. (Jeremy Dick's online list of boilerplates is often cited too – including by CESAR's authors – but it is no longer available.)

Examples given by CESAR include: (ex. 1) "The <user> shall be able to <capability>"; (ex. 2) "...at a minimum rate of <number> times per <unit>" and (ex. 3) "While <operational condition>". These can be combined if necessary (e.g. "While <operational condition>, the <user> shall be able to <capability> at a minimum rate of <number> times per <unit>").

They constitute a useful guideline for expressing capabilities and are rather flexible. However, if example 2 requires only to indicate a (positive) number and a unit (of time) and is therefore subject to only slight variations, this is not really the case for examples 1 and 3, where "user", "capability" and "operational condition" could be replaced by almost an infinity of very different values (in the document, "order entry clerk", "raise an invoice" and "(While) disconnected from a source of power" are given as examples, respectively). Although the attribute "user" is rather restrictive (especially if these boilerplates are used in conjunction with an ontology of the domain, where the class "user" is properly defined), "operational condition" is a very large category and the writer has no real clues on how to fill in this gap (which means s/he is free to do as s/he wishes). For instance, one could have preferred a more explicit version (where the subject is mentioned) instead of the participle clause: "While *the photographer is* disconnected from a source of power". On the other hand, as it is, this boilerplate does not allow the writer to replace "While" by "Even when" or another conjunction that might be preferable in the context. In addition to these limitations, some of the boilerplates they propose seem very close, such as example 2 and "… at least <number> times per <unit>", where the only difference is "at least" vs. "at a minimum rate of".

Thus, these boilerplates are probably good examples, but they also show the main limitations of such templates: it may be difficult for the user to choose an expression over another; they are sometimes too restrictive; and most of all, they are often too permissive, because they are too generic (and their number must remain rather low). They do help limit variation, but they could probably be refined by taking better into account the environment where they will be used, so that they are more specific and adapted. According to us, this can only be done by first analyzing genuine samples of written productions thanks to corpus processing tools.

Our idea is to move beyond the canonical concept of boilerplate and to combine a few generic structures (such as the ones given above) with more specific fixed phrases (such as "…by taking into account…") that can fill in the gaps in a flexible way. Of course, these phrases will be much more numerous, and it would not be possible for a writer to memorize them all; but they could easily be added to requirements authoring software with two major benefits. First, when a user is typing the first word(s) of a frequent phrase, the remaining words could be automatically suggested (similarly to the word completion feature proposed by some software or search engines); this would guide him/her in the difficult writing process and also save him/her time. Second, if an expression is recognized, but another one should be preferred (e.g. if "equal to or greater than" is considered clearer than "not lower than"), it could be automatically highlighted and the preferred expression could be suggested (similarly to the

spell check function of text processors) – it is then up to the writer to accept it or not. This would be a further step in reducing variation (without reducing expressiveness).

2.2 Corpus, tool and method

In order to automatically identify these phrases, we constituted a corpus made of requirements extracted from several specifications written (in French) at CNES for two past projects, namely Pleiades (two very-high-resolution satellites for Earth observation) and Microscope (a microsatellite, whose main objective is to verify a physical principle). About 2,500 requirements were extracted for Pleiades and approximately 1,000 for Microscope (a smaller project). After all tables and figures were removed (as an automatic analysis would be much more difficult), the Pleiades subcorpus was composed of slightly more than 118,000 words, and the Microscope subcorpus was composed of more than 43,000 words, for a total of 161,403 words in the corpus. (Confidentiality reasons explain why we could not get access to more data.)

We assume that this corpus is representative because it includes authentic requirements from two different projects (15 specifications for each). Although these specifications were written under similar circumstances and represent all the levels of the so-called "product trees" (system, subsystem, interface) for both projects, Pleiades and Microscope have totally different scales and purposes (as mentioned above) and the writers were not the same engineers. Consequently, the corpus as a whole is consistent and the two subcorpora (which share essential similarities but also differ in some respects) are comparable (despite the difference in size).

To automatically extract the patterns from this corpus, we used the sequential data mining tool SDMC[5] (Sequential Data Mining under Constraints) [18]: given an input text file and several parameters (threshold, minimum and maximum length of the patterns, etc.), it returns a comprehensive list of patterns found in this file. The major drawback of this method is that, depending on the size of the text and on the parameters (in particular threshold and type of pattern: frequent, closed, maximal), the number of results may be extremely large, and a manual revision would obviously be too time-consuming if it exceeds a few thousands.

To reduce the number of results and keep only those that could be relevant for our study, we extracted frequent patterns from the two subcorpora independently[6]. The minimal length was 2 items and the threshold was 2 utterances (i.e. the minimal number for the pattern to be considered "recurring" (as noted by Sinclair and Tognini-Bonelli, quoted in [19])). Given that both subcorpora are rather small, we were forced to perform the extractions on lemmas[7], not on word forms – therefore this is not a strict corpus-driven approach, according to Biber [19].

---

[5] SDMC can be used online for free at <https://sdmc.greyc.fr/>. However, we had to use an offline version instead – again for confidentiality reasons.

[6] All named entities were previously replaced by the same tag (NAM), so that they are considered the same item by SDMC. For example, thanks to this replacement, the sentences "The system must provide ORAMIC with all the data…" and "The system must provide GAZMIC with all the data…" would be considered the same pattern.

[7] SDMC relies on the results of TreeTagger [20] for lemmatization.

We obtained more than 63,000 patterns for Microscope and almost 100,000 patterns for Pleiades, but we kept only the 3,854 patterns that were common to both lists. This is a much more reasonable number and insures that they are recurring in different projects (and not specific to only one[8]).

To further lower it, we performed a third extraction, this time on a collection of newspaper articles (from *Le Monde*) composed of exactly ten times as many words as the requirements corpus; all the patterns that were present in these results were then removed from our previous list, since they are common also in another genre and thus cannot be considered specific to requirements (SDMC does not use a reference corpus for comparison). Since the minimum number of utterances in the whole requirements corpus was 4 (2+2; a relative frequency of 1 per 40,000 words approximately) and the newspaper corpus is ten times bigger, the threshold for patterns with the same relative frequency would be 40 (4×10). Instead, we decided to set it to 10, meaning that we kept only patterns that were at least four times more frequent in requirements than in newspaper articles. Thanks to this final step, an additional 1,413 patterns[9] were eliminated.

The resulting list is composed of 2,441 patterns (the longest one is composed of 9 items and the average length is 2.8 items) that were manually reviewed. Some of them are commented in the next section.

3 Examples

Unlike those proposed by CESAR, very few of the patterns that were extracted can be used directly as full sentences or even as clauses. Among them are "Il est à noter que…" ("It should be noted that…") – where "Il est" is actually often omitted – and "Il doit être possible de…" ("It must be possible to…").

Unsurprisingly, "être" ("to be") and "devoir" ("must") are by far the most common verbs found in the patterns. The copula "être" may be followed by an adjective: "actif" ("active"), "autonome" ("autonomous"), "cohérent" ("consistent"), "compatible", "responsable" ("responsible"), etc. From these patterns, it may be possible to propose a very generic boilerplate, e.g. <*subject*> *must be* <*adjective*>, or somewhat more restrictive ones, such as <*system*> *must be* <*adjective*>, where <*adjective*> could be any of the above mentioned values. However, "compatible" would not be an acceptable parameter if the subject was <*human operator*>.

As the auxiliary of the passive voice, "être" is also often followed by another verb (a past participle): "activé" ("activated"), "calculé" ("calculated"), "sauvegardé" ("saved"), "validé" ("validated"), etc. The modal "devoir" too is followed by several verbs (infinitives): "exécuter" ("to execute"), "générer" ("to generate"), "fournir" ("to provide"). Interestingly, some of these verbs can be found after "être" and after "devoir", which proves that a same need is sometimes expressed in the active voice ("Le

[8] For instance, "différer l'envoi" ("defer sending") appears twice in Pleiades, but never in Microscope, while "être calculé à bord" ("to be computed on board") has three utterances in Microscope, but none in Pleiades.

[9] Such as "par rapport à" ("in relation to") and "la position de" ("the position of").

CECT doit conserver une copie locale": "*must keep* a local copy") and sometimes in the passive voice ("la donnée est conservée en ligne": "*is kept* online")[10]. Some verbs are peculiar, because they are found after the negation of "devoir"[11]: "entraîner" ("to cause"), "dépasser" and "excéder" (both meaning "to exceed").

Some expressions are very specialized such as "gérer en configuration" (used only in software configuration management), while others exist in general language but seem to be specific collocations in the corpus, such as "en phase de [routine/qualification]" ("in routine/qualification phase"), or even "en phase de recette en vol", which seems to be used only at CNES.

Finally, we would like to show that several expressions may be used with the same purpose. This is the case, among many other examples, with the prepositional phrases "à l'aide de [l'IHM]"/"grâce à [la fonction]"/"au moyen de [l'outil]"/"au travers de [cette fonction]", that can be all translated by "through", "thanks to" or "by means of"; the past participles "[être] décrit/défini/spécifié/détaillé [dans le document…]" ("[be] described/defined/specified/detailed [in document…]"); the verbs "avoir en charge [la réception]"/"être responsable de [la gestion]"/"gérer [la liste des clés]" ("to be in charge of"/"to be responsible for"/"to manage"); the patterns "être conforme à [la spécification]"/"conformément à [la définition]"/"dans le respect de [la spécification]"/"doit respecter [les exigences de sécurité]" ("in compliance with"/"in keeping with"/"must respect"); or "La durée maximale/maximum [avant délivrance] est de <nombre> <unité>"/"La durée [de stockage] est limitée à/ne doit pas dépasser/ne doit pas excéder/doit être inférieure ou égale à <nombre> <unité>" ("The maximum duration is <number> <unit>"/"The duration is limited to/must not exceed/must be less than or equal to <number> <unit>"). This last case is very frequent and, as can be seen, there are plenty of possible variations; a simple boilerplate such as *The <maximum|minimum> <duration/length/…> <complement> is <number> <unit>* might be useful to maintain consistency among requirements. In the other cases, one could simply recommend to use only one of the phrases whenever possible.

4 Conclusion

In this paper, we evaluated the feasibility and interest of a simple method (which can still be refined) for extracting frequent patterns in a corpus with the aim of reducing linguistic variation in requirements, either by suggesting boilerplates or by identifying "competing" phrases expressing the same need. Of course, for each of them, the experts will be in charge of choosing the most relevant one before the list can be added to requirements authoring tools for automatic suggestion. This decision could be made through a survey where variants are presented in context and must be rated [21].

We believe that in specific contexts, such as technical writing, rules and norms may be very useful to users, as long as they are based on existing regularities, and not on mere intuitions or value judgments, as is unfortunately too often the case.

---

[10] Of course, a combination is possible as well: "must be kept".

[11] "ne pas devoir" is sometimes ambiguous in French: "must not"/"does not have to". Here, the first translation is very likely to be preferred.

Acknowledgment

We would like to thank Daniel Galarreta, Nicolas Deslandres and Jean-François Gory for their strong involvement in this doctoral research, which was granted by CNES and the Regional Council of Midi-Pyrénées (France). We also wish to thank the developers of SDMC, especially because this study would not have been possible without the offline version.

References

1. Kamsties, E., Peach, B.: Taming ambiguity in natural language requirements. In: Proceedings of the Thirteenth International Conference on Software and Systems Engineering and Applications (2000).
2. Zhang, Q.: Fuzziness - vagueness - generality - ambiguity. Journal of Pragmatics. 29, 13–31 (1998).
3. IEEE Standard Glossary of Software Engineering Terminology. IEEE Std 610.12-1990. 1–84 (1990).
4. Nuseibeh, B., Easterbrook, S.: Requirements Engineering: A Roadmap. In: Proceedings of the Conference on the Future of Software Engineering. pp. 35–46. ACM, New York, NY, USA (2000).
5. Kuhn, T.: A Survey and Classification of Controlled Natural Languages. Computational Linguistics. 40, 121–170 (2014).
6. Meyer, B.: On Formalism in Specifications. IEEE Softw. 2, 6–26 (1985).
7. Swales, J.M.: Reflections on the concept of discourse community. ASp. La revue du GERAS. 7–19 (2016).
8. Condamines, A., Warnier, M.: Towards the creation of a CNL adapted to requirements writing by combining writing recommendations and spontaneous regularities: example in a space project. Lang Resources & Evaluation. 51, 221–247 (2017).
9. Tognini-Bonelli, E.: Corpus Linguistics at Work. John Benjamins Publishing (2001).
10. Biber, D.: Corpus-Based and Corpus-driven Analyses of Language Variation and Use. In: Heine, B. and Narrog, H. (eds.) The Oxford Handbook of Linguistic Analysis. Oxford University Press (2009).
11. Bhatia, V.K.: Analysing genre: Language use in professional settings. Longman, London (1993).
12. Quiniou, S., Cellier, P., Charnois, T., Legallois, D.: Fouille de données pour la stylistique: cas des motifs séquentiels émergents. In: Journées Internationales d'Analyse Statistique des Données Textuelles (JADT'12). pp. 821–833 (2012).
13. Sitri, F., Tutin, A.: Présentation. Lidil. Revue de linguistique et de didactique des langues. 5–18 (2016).
14. Legallois, D., Charnois, T., Poibeau, T.: Repérer les clichés dans les romans sentimentaux grâce à la méthode des « motifs ». Lidil. Revue de linguistique et de didactique des langues. 95–117 (2016).
15. Warnier, M., Condamines, A.: A Methodology for Identifying Terms and Patterns Specific to Requirements as a Textual Genre Using Automated Tools. Presented at the Terminology and Artificial Intelligence (TIA'2015) November 4 (2015).
16. Tommila, T., Pakonen, A.: Controlled natural language requirements in the design and analysis of safety critical I&C systems. SAFIR2014 Reference group. 2, (2014).

17. Rajan, A., Wahl, T. eds: CESAR - Cost-efficient Methods and Processes for Safety-relevant Embedded Systems. Springer Vienna, Vienna (2013).
18. Quiniou, S., Cellier, P., Charnois, T., Legallois, D.: What About Sequential Data Mining Techniques to Identify Linguistic Patterns for Stylistics? In: International Conference on Intelligent Text Processing and Computational Linguistics (CICLing'12). pp. 166–177. New Delhi, India (2012).
19. Biber, D.: A corpus-driven approach to formulaic language in English: Multi-word patterns in speech and writing. International journal of corpus linguistics. 14, 275–311 (2009).
20. Schmid, H.: Treetagger: a language independent part-of-speech tagger.
21. Warnier, M., Condamines, A.: A Case Study on Evaluating the Relevance of Some Rules for Writing Requirements through an Online Survey. In: 25th IEEE International Requirements Engineering Conference. To appear.