Abstract—Frequently requirements are written in natural language which may cause communication problems—misunderstandings. Misunderstandings are sources of risk that can severely impact the whole software development project. Since patterns and templates are tools for improving the quality of requirements, the question arises whether they can be treated as risk guards. In the paper we propose a method based on HAZOP for pattern identification for non-functional requirements (NFRs). Our exploratory case study on cost and effectiveness of the proposed method included 125 NFRs. The analysis of the identified problems led us to propose three process patterns and provided rationale for using a three templates. Generally, the proposed method might be used to improve the existing patterns and templates as well as to develop new ones.

I. INTRODUCTION

Natural language is commonly used for documenting requirements in software development projects [1], [2], [3]. Although this approach does not require any special training, it may cause serious communication problems—misunderstandings. Misunderstandings are sources of risk that can severely impact the whole software development project and its stakeholders, e.g., when a requirement has more than one interpretation [1], [2], [3]. Since patterns and templates are tools for assuring quality of requirements [4], the question arises whether they are suitable to be treated as risk guards.

Thus, in order to answer the question, we decided to develop a systematic approach to identifying patterns and templates for non-functional requirements. The steps of our approach are derived from the well-known risk management approach—M_o_R® [5]. The goal of the first step is to identify possible risks, but the M_o_R® does not suggest any specific single procedure how to achieve it. Therefore, we decided to employ a proven technique for detecting and analyzing hazards in systems called HAZOP [6]. A result, our approach requires performing a review of a set of NFRs to identify possible misunderstandings by using HAZOP keywords. The output of the method, i.e., detected anomalies, their causes, possible consequences (risks) (see Section II for more details), will be used as the rationale for using patterns and templates.

There have been already proposed a few catalogs of patterns for non-functional requirements (e.g., [4], [7]), but none of them have become a widely used standard. Many organizations are affected by the so-called NIH (Not Invented Here) syndrome and adjust every method before its application [8], [4]. With the output of our method containing, e.g., information on probability and consequences of risks, they can more effectively manage requirements risks. Then, the organization might either select or update an already existing pattern, or create a new one to the most important detected anomalies.

In the paper we also report on our exploratory case study (for the design see Section III). The study was focused on efficiency and effectiveness of the identification of misunderstandings and the potential to use them to propose patterns. The results we present in Section IV. Then we discuss the usage of patterns and templates as risk guards for the identified anomalies in Section V. We analyzed the related work and compare in Section VI with them our contribution. Moreover, the results of the validation motivated us to formulate the directions of the future work (Section VII).

II. HAZOP-BASED ANALYSIS

A. General Idea and the M_o_R® Approach

The risk of misunderstandings in non-functional requirements engineering exists since this process involves a lot of communication and natural language is frequently used. By a misunderstanding in the NFR context we understand any deviation in understanding an NFR that may result in the implementation of the NFR that differs from the intention.

We decided to employ a well-known approach to risk management—M_o_R® [5] as a framework defining the process of our method for identification of patterns for non-functional requirements. To support the first and crucial step in the framework that aims at identification of risks we selected a technique for identification of hazards in systems—HAZOP [6] (Section II-B). But some adaptations to the HAZOP's keywords were introduced (Section II-C). Then, the next step of the M_o_R® approach is to assess the risks in order to indicate their importance, probability etc. Thus we proposed our assessment approach for the impact and probability of misunderstanding for NFRs. The third step of M_o_R® suggests that for each identified risk one should select one or more strategies to mitigate the risk, e.g., avoid, reduce. In the context of NFRs we propose that such a strategy would be the usage of certain patterns or templates. Therefore, they can be perceived as risk guards then and applied in a software development...
project as the last step suggest. Figure 1 illustrates the process of M_o_R mapped and adjusted for our method.

### B. HAZOP

A HAZard and OPerability study (HAZOP) is a systematic technique that aims at the identification of hazards. It was originally created for the chemical industry for process plants, but has also been successfully used in software engineering [6]. Hence, we decided to use it in our process for the identification of risks in NFRs (see the step Identification Figure 1). The idea of HAZOP is to perform a review of a system or a design of system within a series of workshops. During the workshops participants (experts with different experience and knowledge) identify deviations from the design intent that can have severe consequences (rise hazards). Each part of the design is questioned with the use of combinations of guide words: primary keywords (attributes), e.g., temperature, and secondary keywords (guide words), e.g., NO, LESS, MORE. While primary keywords are relevant properties of the items of the design, and change with regard to the design type, there is a set of guide words which indicate a type of deviation from the design intent. Experts try to interpret each pair of keywords to discover possible risks, e.g., Temperature-MORE might indicate that there is a hazard (risk) that temperature would get too high. Each such finding must be documented in a report (Figure 2 presents a template of the report), with its cause, consequences, recommendations for protection against the identified hazard.

### C. Keywords

To analyze risk in non-functional requirements we needed to prepare a set of primary keywords which are in HAZOP defined as relevant attributes of system design. Since a system design constitutes of entities, we for the NFR context we mapped a non-functional requirement documented in natural language to an entity. Then we identified the relevant attributes of a non-functional requirement using the IEEE standard 830-1997[9]. The standard defines that each good SRS should be: correct, unambiguous, complete, consistent, ranked for importance and/or stability, verifiable, modifiable, traceable. We combined these criteria with the SMART characteristics defined by Mannion and Keepence [10] and with the set of the taxonomy of faults in natural language requirements [11]. This resulted in the following list of primary keywords:

- **Interpretation** - an NFR should be specific enough, so the stakeholders can provide one correct interpretation.
- **Level of Detail** - an NFR should be documented with the use of all necessary information but without extraneous information.
- **Achievable** - generally it should be possible to realize an NFR.
- **Realizable** - it should be possible to satisfy an NFR under the constraints defied for a certain system.
- **Real need** - an NFR should be important for a given business case and context of a project.
- **Verifiability** - there should always exist a cost-effective method with the use of which we can show the evidence that an NFR is satisfied.

It should be noted that to analyze an NFR with regard to some of the proposed keywords (relevance, real need and partially verifiability) one also needs access to more information about the project, e.g., business case, budget.

This set of primary keywords can be extended. For example the experts in our exploratory case study found it important to add **Syntactic correctness** defined as *an NFR should be formulated correctly with regard to the rules of language it is formulated in (e.g., grammar rules, orthography)*.

We decided to remain the guide words used in HAZOP for software presented in Redmill et al. [6]: NO, MORE, LESS, AS WELL AS, PART OF, REVERSE, OTHER THAN, EARLY, LATE, BEFORE, AFTER. Figure 3 presents the meaning of pairs of keywords for defined for the non-functional requirements context. We created it based on the IEEE standard 830-1997 [9], works of Mannion and Keepence [10] and Berry et.al. [11], and from the analysis requirements during our case study(Section III).
<table>
<thead>
<tr>
<th>NFR ID</th>
<th>Attribute</th>
<th>Guide word</th>
<th>Cause</th>
<th>Probability</th>
<th>Consequence</th>
</tr>
</thead>
<tbody>
<tr>
<td>NFR-12</td>
<td>Level of detail</td>
<td>LESS</td>
<td>Where to find ABC?</td>
<td>75%</td>
<td>(Dev.) Additional communication (Acceptance) Incorrect implementation</td>
</tr>
</tbody>
</table>

Fig. 2. Report created during a) HAZOP b) HAZOP-based Analysis of NFRs

<table>
<thead>
<tr>
<th>INTERPRETATION</th>
<th>LEVEL OF DETAIL</th>
<th>ACHIEVABILITY</th>
<th>REAL NEED</th>
<th>VERIFIABILITY</th>
<th>SYNTACTIC CORRECTNESS</th>
</tr>
</thead>
<tbody>
<tr>
<td>NO</td>
<td>Expert cannot find any interpretation for the NFR.</td>
<td>Impossible to achieve the NFR.</td>
<td>NFR is irrelevant for the given business case.</td>
<td>No cost-effective method to demonstrate that the NFR is satisfied.</td>
<td>NFR needs reformulation, to be rewritten.</td>
</tr>
<tr>
<td>MORE</td>
<td>There exist more than one interpretation of NFR.</td>
<td>There is too much information/details.</td>
<td>NFR requires too much.</td>
<td>The verification method is difficult or hard.</td>
<td></td>
</tr>
<tr>
<td>LESS</td>
<td>There are some parts that are not clear to expert and greatly influence the interpretation.</td>
<td>There is missing some information, details.</td>
<td>NFR requires too little.</td>
<td>NFR is formulated in a way that requires few improvements, e.g., style.</td>
<td></td>
</tr>
<tr>
<td>AS WELL AS</td>
<td>For informal/urban communication would be clear.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PART OF</td>
<td>There is one word/phrase not clear to expert but does not significantly influence the interpretation.</td>
<td>NFR can be partially achieved.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>REVERSE</td>
<td>The expert finds an opposite meaning of NFR.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EARLY</td>
<td>Should be specified with a greater level of detail before given to developers.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LATE</td>
<td></td>
<td></td>
<td>It will be possible to fully verify NFR after the system is deployed.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AFTER</td>
<td></td>
<td></td>
<td>Provided that other errors in NFR are corrected, it will be possible to verify it.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Fig. 3. The meaning of primary-secondary keyword pairs—indicating deviations in a non-functional requirement

D. Assessment of Identified Risk in NFRs

The M.o.R approach suggests to assess each risk using two aspects: impact and probability. For the risks identified in NFRs we propose to assess impact by identifying the activities that a certain misunderstanding might influence. For example the missing version number in the NFR System shall work in Chrome, might result in: doing too much by supplier (when they try to support all versions of Chrome), additional communication for negotiation (when supplier would like to discuss it the customer), not satisfied expectations of customer during acceptance tests (when it turns out that the system supports only the version of the browser that was the latest when signing the contract). Then, either simply the number of the activities that are identified as impacted might indicate the value of impact, or one may provide priorities (weights) for the activities and calculate the value of impact using other methods, e.g., weighted sum, utility.

We decided that the probability of misunderstanding would be indicated by the experts on a 5-point scale of probability expressed in percentage. One, when selecting the value, shall find the best fit to finish the following sentence: It is <value> that the misunderstanding will appear in the project., where <value> can have the following meanings: 90%–certain or almost certain; 75%–highly probable; 50%–fifty-fifty; 25%–rather difficult; 10%–almost impossible.

E. Analysis Process

Our analysis process can be executed within a workshop or as an individual review of an existing set of non-functional requirements. For guidance how to organize an effective workshop, one may follow the recommendations provided by Gottesdiener [12]. Independently on the selection if the execution approach the process shall include the steps presented in Figure 4.

In HAZOP experts analyze entities (parts of the system design) to discover anomalies from the design intent. The same is achieved in the Step 4. Use keywords to identify the problems in the NFR of our HAZOP-based analysis. An expert takes into consideration one NFR with the associated information (e.g.,
In order to investigate and better understand the HAZOP-based approach to patterns identification of non-functional requirements method we designed a case study according to the guidelines of Runeson et al. [14]. The goal of the case study was to characterize and understand the method focusing on the effectiveness and efficiency of misunderstandings identification and on the possibility of pattern identification from the point of view of a requirements engineer interested in pattern-based requirements improvement approach in the context of software development projects. We refined our goal into a set of the following research questions: RQ1 What problems in non-functional requirements are discovered with the use of the method? RQ2 What is the reliability of the method? RQ3 How much time and resources does it take to use the method? RQ4 To what degree the identified problems can be solved by patterns?

A. Case and Subjects Selection

We decided to conduct our study with the use of two industrial specifications (ABC, XYZ) that come from software projects developed in different domains (ABC–financial, XYZ–public administration) and that describe different types of systems (ABC–a component to be used within a web application, XYZ–a web application). This choice was dictated by our desire to assure variability of requirements as it is early validation of the proposed idea.

Although the client in project ABC has its software development unit, it outsourced a piece of the system to be created by an external company. Both companies are medium-sized enterprises. They created their set of requirements for the project with the use of the ISO25010 [15] standard within a brainstorming session, analyzed it, and included into the contract. The system was a set of components of a so-called backend system that allowed to realize management of several functions. The non-functional requirements gathered for the XYZ project were gathered just before the call for a tender procedure. The project goal was to create and deploy a web application for an administrative unit.

Our case study was divided into two parts. In the first part two experts analyzed together from each set 30% of requirements that were randomly selected. Then one expert used our method as an individual review technique and analyzed all two sets of NFRs. All experts have at least 5 years of experience in requirements engineering. The experts worked in slots dedicated to analyze 10 NFRs and then they were taking at least 5 minutes after each slot. We prepared assessment sheets with the keywords listed for each NFR and sheets to document the probable consequences (impact). The experts had access also to a document presenting vision of each project. The task for the experts was to follow the process described in Section II and identify any risks in each NFR. The experts were to take the perspective as if they would be responsible to implement NFRs.
B. Data Collection and Analysis Procedures

The data that we collected during our study were the outputs of the methods (the reports) and the lessons learned gathered from the study participants. We also asked experts to measure the time spent on the analysis. Descriptive statistics are the main tool used to analyze the data.

C. Threats to Validity

We did not address internal validity as it should be examined when cause-effect relations are to be found. Our case study was not designed to provide statistically valid conclusions for all software development projects or organizations. Its aim was to understand the proposed method in the given context. We tried to provide a detailed description of the context in Section III-A and to assure variability of analyzed project. Thus we believe that such approach might be used in software projects in which the content of NFRs is documented in regular natural language. The threats with regard to construct validity we addressed by describing the HAZOP-based risk analysis method and comparing it to other related works. In order to assure reliability of the study we linked the data to project repositories. The experience of the experts might influence the results of the study. Thus further analysis should involve more people.

IV. RESULTS

RQ1 What problems in non-functional requirements are discovered with the use of the method? The following results include the errors that were identified by all three experts. If there was a conflict if there exist an anomaly in an NFR between the results of the individual expert and two experts, the final decision was made by one of the authors of the papers.

There were only 26.4%(33) non-functional requirements with no risks identified. Altogether we identified 155 possible misunderstandings, the majority of which were due to improper level of detail (36.8%(57)). In Figure 5 we present the percentage of all errors per category (primary keyword).

Looking closer at the results with respect to level of detail we noticed that 23.2%(36) of them are due to providing insufficient information, e.g., using expressions such as “e.g.”, “other” elements, lack of a reference to implied documents. The experts found NFRs that were at first glance detailed enough, but documented in a too general way for implementation. They shall not be given straight to developers but require further elaboration which might be risky. For example, an NFR requiring the system to comply with Personal Records Protection Law shall be first analyzed and then divided into related NFRs such as regarding the strength of passwords, the way the security is assured etc.

We found that many risks with regard to interpretation are due to problems in interpretation (5.8%(9)) or using some unclear words (9.7%(15)) (e.g., the term ‘transaction’ might mean a database transaction, a functionality, a name of functionality). Moreover, there were identified NFRs (5.8%(9)) with more than one interpretation. We also noticed that there was one requirement in which the risk of being understood reversely was present and one with a so-called logical error, i.e., it stated that some transactions shall be rolled back when certain ;task1; and ;task2; would be completed but from the context the experts deduced that there should be ’or’ instead of ’and’. The summary of errors for interpretation as well as for syntactic correctness are presented in Figure 6.

The experts questioned one NFR with respect to its acceptability and found three single errors with regard to the real need aspect. One NFR required more than needed, one shall require another value, and one was too weak. The possible
To answer the stated research question RQ4 To what degree the identified problems can be solved by patterns? we asked the experts to suggest new patterns or templates of propose new ones that might prevent the identified misunderstandings to happen. We ordered the risks that were identified by the experts by the probability of misunderstanding and number of consequences they may cause (Figure 7). Taking the first-things-first we started to work on guards. As a result of our analysis we propose the following 3 patterns and 3 templates.

A. Evolutionary Approach

**Context.** This pattern is intended to be used when one formulates an NFR that is about any regulation, law, domain standards etc., e.g., that the system must comply with Personal Records Protection Regulations of a given country.

**Problem.** The NFR in this context frequently contains the name and the reference to the document describing the regulations, law etc. But it requires from the supplier some additional work to get familiar with the standards, documents and interpret it. Sometimes such documents contain clear requirements, e.g., that the password length should be of 6. However, usually there appear several imprecise statements or such that can be implemented in a different manner. For example: *data about what personal records, when and with whom were shared must be recorded* can be either satisfied as a functionality of the system or as a procedure (somebody can take notes manually about such facts). The Impact might be with the acceptance of the product when it appears that the client had some different expectations, or for the latter case it may mean too much unnecessary work of the supplier.

**Solution.** We propose to identify such requirement first, mark it. Then, one shall work on it with the stakeholders to make it specified to the level clear for the developers. It shall also agreed in any analysis of law, regulations is to be done to include it in the project budget and set responsible people.

B. Procedural Approach

**Context.** People tend to think about non-functional requirements in terms of adjectives and formulates NFRs having multiple interpretations, e.g., the system shall be easy to use for my employees.

**Problem.** The supplier extremely rarely has the same understanding of adjectives as the other stakeholder who formulated the NFR. Then it may Impact the shape and scope of the software product which in result would not satisfy expectations of the customer. The supplier might also have difficulties in developing and the requirement and then testing.

**Solution.** We propose to combine the NFR with a procedure for assuring it. For example one may describe a series of workshops during which the GUI design of the system will be created together with the employees to suit their needs.

C. Adding Verification Method

**Context.** When formulating an NFR people usually think about the qualities in terms of adjectives they expect. But they tend to forget about the procedure they would execute to accept the software.

**Problem.** Defining NFRs which are costly to test or for which there only exist a method (one or more) that can provide some indicators that the requirement would be satisfied. The Impact might be that the end product is not accepted and
the involved parties can not find agreement how to conduct acceptance testing etc. An example requirement is: the system shall be easy to operate for my employees.

**Solution.** We propose to identify such requirement first, and then explicitly connect it with a verification method and the moment in the project when it would be executed. Frequently such requirements should be guarded also with the Procedural Approach pattern.

**Comment.** the idea is similar to the idea of Robertson and Robertson [16] and IEEE 830 [9] that each requirement should have a fit criterion but we extend it by the time when testing is expected to be performed.

**D. Using NoRTs**

**Context and Problem.** Formulating an NFR requires to assure proper level of detail. But from the analysis of the NFRs there is problem to provide all necessary information about the expectations and nothing more.

**Solution.** We propose use Non-functional Requirements Templates(NoRTs) [17]. NoRTs are sentences that require just to fill in some blanks. The blanks are the properties that a stakeholder shall or may require. The following NFRs could be improved with the use of NoRTs:

- **NFR-40:** Users should use the system by a web browser—NoRT66: System shall work for <browser_name;text> <browser_manufacturer{text}>? <browser_version{text}> +.
- **NFR-11:** user guide (…) and shall be delivered electronically—NoRT32: User guide ['for' <actors:$actor$*>] shall (comply with <template:$docref$> — contain <section_name{text}> <section_description{text}>) and be delivered (’printed book’, ’pdf file’, 'doc(x) file’, <other_format{text}>)+.

1The following NFRs and NoRTs were created in Polish so there is threat of introducing errors while translating into English.

- **NFR-108:** Maximal average processor utilization by other parts of the system: XX%—NoRT49: (’System’ — <parts_of_system{text}>+) shall not utilize more (’processor’ — ’RAM’ — <resource_type{text}>) than <value{double}> (during <period{timespan}> — for <transactions_set{text}>).

**VI. RELATED WORK**

Many authors have noted the problems of errors in requirements and their further impact on the software project and on the resulting products and services. Thus there are works systematize the problems that occur, e.g., [11]. There are also some works that define a set of qualities that a requirement should exhibit. For example Manion and Keepence [10] define such characteristics for a requirement no matter if it is functional or non-functional one.

Then, there are some works which also propose the methods for detecting problems in requirements, e.g., [11], [2]. There is a significant set of works analyzing the errors from the linguistic perspective, e.g., [18], and those trying to systematize existing ambiguities in requirements, e.g., [1], [2]. However, to our up-to-date knowledge there exists no work about systematical identification of problems from different perspectives focusing on NFRs.

There are certain ideas on how to help to avoid problems in requirements. One approach would be to assure the quality by executing certain process carefully and correctly, e.g., elicitation methods [1]. Another approach is use a notation that is less prone to introducing errors in requirements, e.g., UML notation or formal languages. Then, some who are aware of ease of expressing the requirements in natural language propose to follow a defined structure of a requirement, e.g., VOLERE [16] or use cases [19]. One may also use patterns and templates for requirements, which is a quite frequently applied approach in the field of NFRs. For example there were proposed set of patterns, e.g., for security requirements [20], for awareness [21], but there are also extensive catalogs of

<table>
<thead>
<tr>
<th>NFR ID</th>
<th>Attribute</th>
<th>Guide word</th>
<th>Cause</th>
<th>Probability</th>
<th>Consequence</th>
</tr>
</thead>
<tbody>
<tr>
<td>1596</td>
<td>Level of Detail</td>
<td>LATER</td>
<td>Not enough information for development, needs to be evolved</td>
<td>90%+</td>
<td>(Development) Work to understand regulations; Wrong Implementation</td>
</tr>
<tr>
<td>1789</td>
<td>Interpretation</td>
<td>MORE</td>
<td>easy to use can be understood in various ways</td>
<td>90%+</td>
<td>(Development); Wrong Implementation; Too much work; (Acceptance) Expectations are not met</td>
</tr>
<tr>
<td>7458</td>
<td>Verification</td>
<td>NO</td>
<td>There are too many possible verification methods</td>
<td>90%+</td>
<td>(Acceptance) Disagreement with regard how to verify the NFR</td>
</tr>
<tr>
<td>325</td>
<td>Level of Detail</td>
<td>LESS</td>
<td>No information on the goal, contents, format of documentation</td>
<td>90%+</td>
<td>Development; Wrong Implementation; Too much work; (Acceptance) Expectations are not met</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Fig. 7. A fragment of the report from risk identification in NFRs
patterns e.g., [7], [4], or templates [17]. Although it is known that a pattern library (or a catalog) should exhibit high retrievability and relevance, there is scarcity of information about the methods that allow to maintain it. The intention behind our method was to construct a method which would allow to systematically maintain such catalogs.

Therefore we decided that our proposal will be based on well defined methods, i.e., M_o_R and HAZOP. The choice of HAZOP was also based on the fact that there are several known applications of HAZOP in the software engineering domain. For example Jarzebowicz [22] proposed to use HAZOP in order to assess the quality of UML diagrams and detect anomalies. Srivatanakul et al. [23] used HAZOP for analysis of safety critical systems. Allenby and Kelly [24] applied HAZOP to determine security requirements based on use cases. Jurkiewicz et al. [25] used HAZOP to identify exceptional events in use cases while eliciting requirements.

Additionally, in the agile software development projects it is essential to deliver software fast [26], and as a consequence the methods that allow to persist the knowledge gained during development shall support this principle. Currently, it is known that pattern development is a laborious process not only for requirements [4] but also in other software engineering processes, e.g., for design patterns [8]. One of the factor that may greatly influence efficiency of pattern mining in our opinion is scarcity of systematic methods. Moreover, it is little known about the exact efficiency of pattern creation process. A few authors report on their methods proposed for identifying problems in requirements, e.g., Kamsties et al. report that it took in total 4.5h to analyze one requirement document by three people to identify ambiguities [27]. Thus, we contribute with our study with empirical data on efficiency and effectiveness of identifying anomalies in NFRs and on pattern development.

VII. CONCLUSIONS

In the paper we propose a method for pattern identification in non-functional requirements. It proposes to use patterns and templates as guards against possible misunderstandings of NFRs. The method combines the well-known risk management approach–M_o_R© and the widely used system hazard identification technique–HAZOP. The output of our method is a list of risks identified in each NFR with the information of their probability and impact. Our method underwent an initial study in a form of an exploratory case study. The findings indicated that the most frequent problems in NFRs are about too low level of details and using some expressions which are difficult to interpret by others. The study results allowed us to create new patterns for NFRs and to improve the method itself. Moreover, the results on cost(time) of our method extend the knowledge about the efficiency and effectiveness on patterns development and maintenance activities. In the future we plan to analyze more non-functional requirements with our method.

ACKNOWLEDGMENT

We thank the representatives of the organizations for sharing the data on non-functional requirements with us.

REFERENCES


[17] S. Kopczynska and J. Nawrocki, “Uses of patterns e.g., [7], [4], or templates [17]. Although it is known that a pattern library (or a catalog) should exhibit high retrievability and relevance, there is scarcity of information about the methods that allow to maintain it. The intention behind our method was to construct a method which would allow to systematically maintain such catalogs. Therefore we decided that our proposal will be based on well defined methods, i.e., M_o_R and HAZOP. The choice of HAZOP was also based on the fact that there are several known applications of HAZOP in the software engineering domain. For example Jarzebowicz [22] proposed to use HAZOP in order to assess the quality of UML diagrams and detect anomalies. Srivatanakul et al. [23] used HAZOP for analysis of safety critical systems. Allenby and Kelly [24] applied HAZOP to determine security requirements based on use cases. Jurkiewicz et al. [25] used HAZOP to identify exceptional events in use cases while eliciting requirements. Additionally, in the agile software development projects it is essential to deliver software fast [26], and as a consequence the methods that allow to persist the knowledge gained during development shall support this principle. Currently, it is known that pattern development is a laborious process not only for requirements [4] but also in other software engineering processes, e.g., for design patterns [8]. One of the factor that may greatly influence efficiency of pattern mining in our opinion is scarcity of systematic methods. Moreover, it is little known about the exact efficiency of pattern creation process. A few authors report on their methods proposed for identifying problems in requirements, e.g., Kamsties et al. report that it took in total 4.5h to analyze one requirement document by three people to identify ambiguities [27]. Thus, we contribute with our study with empirical data on efficiency and effectiveness of identifying anomalies in NFRs and on pattern development.

VII. CONCLUSIONS

In the paper we propose a method for pattern identification in non-functional requirements. It proposes to use patterns and templates as guards against possible misunderstandings of NFRs. The method combines the well-known risk management approach–M_o_R© and the widely used system hazard identification technique–HAZOP. The output of our method is a list of risks identified in each NFR with the information of their probability and impact. Our method underwent an initial study in a form of an exploratory case study. The findings indicated that the most frequent problems in NFRs are about too low level of details and using some expressions which are difficult to interpret by others. The study results allowed us to create new patterns for NFRs and to improve the method itself. Moreover, the results on cost(time) of our method extend the knowledge about the efficiency and effectiveness on patterns development and maintenance activities. In the future we plan to analyze more non-functional requirements with our method.

ACKNOWLEDGMENT

We thank the representatives of the organizations for sharing the data on non-functional requirements with us.