Eckhardt, Jonas; Vogelsang, Andreas; Femmer, Henning

An approach for creating sentence patterns for quality requirements

Eckhardt, J., Vogelsang, A., Femmer, H. (2016): An Approach for Creating Sentence Patterns for Quality Requirements. In: 2016 IEEE 24th International Requirements Engineering Conference Workshops. IEEE. https://doi.org/10.1109/rew.2016.057

An Approach for Creating Sentence Patterns for Quality Requirements

Jonas Eckhardt
Technische Universität München
Garching b. München, Germany
eckharjo@in.tum.de

Andreas Vogelsang
DCAITI
Technische Universität Berlin
andreas.vogelsang@tu-berlin.de

Henning Femmer
Technische Universität München
Garching b. München, Germany
femmer@in.tum.de

***Abstract*—Requirements are usually categorized in functional requirements (FRs) and quality requirements (QR). FRs describe "things the product must do" while QRs describe "qualities the product must have". Besides the definition, classification, and representation problems identified by Glinz, there are two further problems with current definitions of quality requirements: (i) the definitions are imprecise and thus difficult to understand and apply, and (ii) the definitions provide no guidance or support for their application in a given organizational context. To tackle these two problems, we propose an approach that—given a quality attribute (e.g., performance) as input—provides a means to specify quality requirements by sentence patterns regarding this quality attribute. In this paper, we contribute a detailed presentation and description of our approach and a discussion of our lessons learnt while instantiating it for performance requirements. Additionally, we give guidance on how to apply our approach for further quality attributes. Through this approach, we aim at encouraging researchers to help us improve the precision of definitions for quality requirements and support practitioners in eliciting and documenting better quality requirements.**

***Index Terms*—Quality Requirements, Sentence Patterns**

I. Introduction

Requirements are usually categorized in *functional requirements* (FRs), *quality requirements* (QRs) and *constraints* [1]. FRs are characterized as "things the product must do" contrasting QRs as "qualities the product must have" and constraints as "organizational or technological requirements". Although the importance of QRs for software and systems development is widely accepted, up until now, there is no commonly accepted approach for the QR-specific elicitation, documentation, and analysis [2]–[4]. This lack can result in high maintenance costs in the long run [3].

Besides Glinz's definition, classification, and representation problem [5], there are two further problems with current definitions of quality requirements: (i) the definitions are not overly precise and thus not easily understandable and applicable, and (ii) the definitions do not provide guidance or support for their application in a given organizational context.

To tackle these two problems, we propose an approach that—given a quality attribute (e.g., performance) as input—provides a means to precisely specify requirements regarding this quality attribute. Our approach is based on the identification of content elements, i.e., different types of information characterizing the quality attribute (e.g., the desired *latency* of a system for performance requirements). In particular, given a quality attribute, our approach provides (i) a precise and explicit definition of content elements that are needed to specify requirements concerning the quality attribute, and (ii) a set of sentence patterns for practitioners to specify requirements concerning the quality attribute for a given organizational context. We achieve the precise and explicit definition by a structured identification of relevant content elements that requirements of a specific quality attribute may consist of. Furthermore, we use the idea of activity-based quality models [6], [7] for the customization of these content elements to a given organizational context and sentence patterns for guidance and support for their application in practice.

We already instantiated our approach for one specific quality attribute (performance) and conducted an empirical evaluation with respect to its applicability [8]. The results indicated that the approach is applicable and besides the constructive nature of our approach, further supports analytic quality assessment with syntactic analyses. For example, the question how can we assess that all information necessary are documented in a given textual requirement (i.e., the completeness[1] of the individual requirement)?

In this paper, we contribute a detailed presentation and description of our approach, a discussion of our lessons learnt while instantiating it for performance requirements, and provide guidance for how to apply our approach for further quality attributes. The objective of this paper is to encourage other researchers to create more precise definitions for quality requirements.

The remainder of this paper is structured as follows: In Section II, we present our approach and discuss its application to performance requirements in Section III. We discuss the threats to validity of our approach and lessons learnt in Section IV. Finally, in Section V, we report on related work and conclude in Section VI.

II. Approach

Fig. 1 shows an overview of our approach: the approach takes a specific quality attribute as input and creates a precise and explicit definition and customized sentence patterns for requirements concerning this quality attribute. The resulting

[1]Completeness can be considered on two levels: complete requirements specifications as a whole or complete requirements, i.e., all information necessary for single requirements. In the following, we focus on the latter.

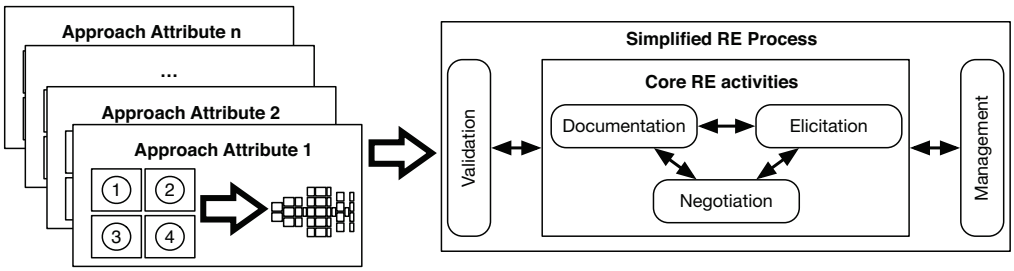

Fig. 1. Overview of the quality requirments definition approach and its integration in a simplified RE process (according to Pohl [1]).

definitions and sentence patterns can then be integrated in the overall RE process to support the elicitation, documentation, validation, and management of requirements in the given organizational context. Thus, our approach needs to be conducted in advance for a given set of quality requirements and a given context. Then, the results can be (re)used as, for example, a company standard to specify and elicit quality requirements.

A. Goals of the Approach

Before we describe the approach in detail, we first discuss the goals of the approach. Given a quality attribute, as for example performance, we try to achieve the following four goals:

1) **Identification of relevant content elements:** In literature there exists a large amount of publications concerning individual quality attributes. The challenge is to collect this large amount of qualitative data and extract the relevant content elements in a structured and reproducible way that guarantees that all relevant content elements are considered. The quality of the overall results of our approach heavily depends on the content elements that are identified to be *needed to specify requirements concerning the quality attribute*.
2) **Precise definition of relevant content elements:** Given a set of relevant content elements, a further challenge is how to define these precisely such that all stakeholders have the same understanding. This is a challenging yet creative activity. For example, we may define each content element by means of a glossary entry or give a formal definition by a mapping to a system model. The challenge is to find a way to define the content elements such that they are adequately represented. This activity is highly dependent on the context (e.g., involved stakeholders).
3) **Customization to a given organizational context:** Another challenge is to assess whether the content elements are actually relevant for a given organizational context. The simple answer here is to provide a one-size-fits-all solution. However, we argue that such a one-size-fits-all solution does not work for requirements engineering because the organizational contexts vary heavily. These variations include not only the information and level of detail in which projects document requirements but also how projects use requirements documents in their context. Thus, the challenge is to provide an approach that can be customized for a given organizational context.
4) **Provide a means to specify requirements for practitioners:** Finally, based on the relevant content elements for a given context, we aim to create a means that supports the structured elicitation, documentation, and management of requirements concerning this quality attribute.

B. Overview of the Approach

To meet the goals described above, our approach consists of four steps. Fig. 2 shows an overview of our approach; it takes a specific quality attribute as input (e.g., performance). The approach is separated in two parts: The first part (Step ① and ②) is intended to create a precise and explicit definition of the quality attribute while the second part (Step ③ and ④) is intended to customize the definition to a specific organizational context and to provide a means for practitioners to specify requirements concerning this type.

1) Context-independent Definition: The goal of this step is to create a comprehensive content model that covers all content elements and relationships that are needed to specify requirements concerning the quality attribute. Fig. 3 shows an overview of this step. To get a complete list of content elements, we propose to use qualitative literature analysis (e.g., a structured literature review or expert interviews) with the goal to identify concepts related to the specification of requirements concerning the quality attribute. Then, in a next step, we identify the models that are used for the specification of requirements; These models may be in textual, semi-formal, or formal form (indicated by different icons in the figure). Then, based on these models, we identify content elements and create a consolidated content model that contains and relates all content elements. This qualitative analysis is a highly creative and subjective approach. We suggest to use researcher triangulation to reduce this threat to the overall validity. The result of this step is a content model that ideally is a superset of all aspects concerning the quality attribute in literature.

2) Precise Definition: Given the content model from the previous step, the goal of this step is to give a precise definition for the individual content elements of the content model. For each content element of the content model, we define both, its

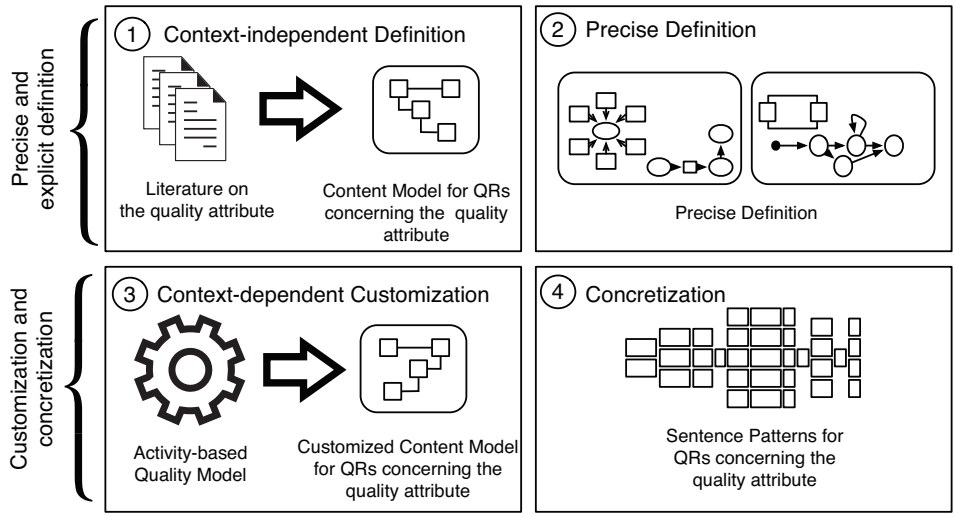

Fig. 2. Overview of the approach

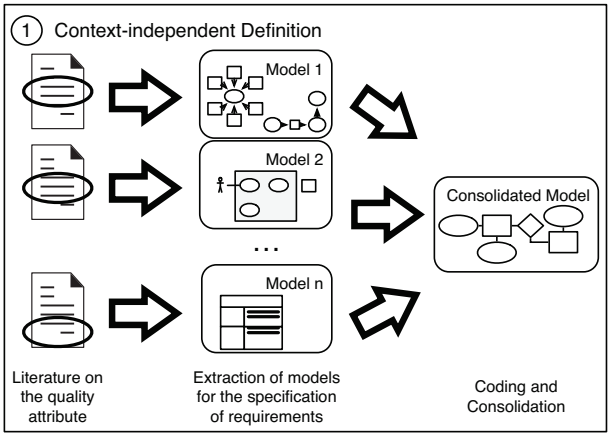

Fig. 3. Overview of step 1: Context-independent definition.

syntax as well as its semantics. Depending on the stakeholders, we may give a definition on differnt levels of detail ranging from an informal glossary entry to a formal definition. For example, if we chose to use a glossary, we may define the syntax of the content element *modality* (of a requirement), as "the modality of a requirement may be one of *exclusion*, *obligation*, *enhancement*" and its semantics as "If the modality of a requirement is *exclusion*, the property described by the requirement must not hold, if it is an *obligation*, it must hold, and if it is an *enhancement*, it may hold". If we want to define the content element more formally, we suggest to describe its meaning in terms of a system model (e.g., [9]). For example, if we aim to define the semantics of a *requirement*, we can map it to a logical predicate, which relates input streams to output streams. As with the previous step, this step is highly creative and depending on the context. The result of this step is a definition of each content element of the content model.

3) Context-dependent Customization: The goal of this step is to achieve a customization of the context-independent content model for a given organizational context. To achieve this, we propose to use the idea of activity-based quality models [6], [7] and use the context-independent content model as input for the creation of the activity-based quality model for the given context. In particular, we use a model of stakeholders and their development activities that take requirements of the quality attribute as input (e.g., *design a test* based on a performance requirement). Based on this model of activities, we successively analyze the content elements that a stakeholder needs in a requirement to complete the activity efficiently and effectively. For example, to perform the activity *designing a test*, it is necessary to know the scope of the requirement. Therefore, we classify content elements as mandatory or optional for an activity. The result of this step is a content model for the quality attribute that is adapted to a specific set of activities and where each content element is justified by at least one of these activities. By this, we achieve a customization of the content model to a given organizational context.

4) Concretization: In the final step, we must provide a means for practitioners to specify requirements concerning the quality attribute. To achieve this, we propose to derive a set of sentence patterns from the context-dependent content model. Sentence patterns have the advantage that they are easy to use for the documentation of requirements and support the structured elicitation and management of requirements. Fig. 4 shows an overview of the step. In particular, for each of the content elements in the content model, we derive a *sentence fragment*. The sentence fragment is intended to represent the meaning of the content element as close as possible. For example, lets assume that the content element *modality* of a requirement may be an *enhancement*, an *obligation*, or an *exclusion*. In this case, we can create the sentence

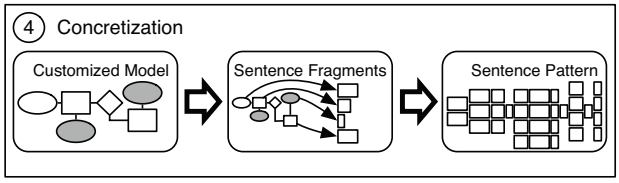

Fig. 4. Overview of step 4: Concretization.

fragments `could/may` for enhancement, `must/shall` for obligation, and `must not` for exclusion. Furthermore, sentence fragments may also contain variables that have to be replaced by values when the pattern is instantiate. For example, if we aim to represent an optional content element which describes a specific start event, we can represent it by the sentence fragment `[start event <A>]`. The angle brackets indicate the variable while the square brackets indicate that this sentence fragment is optional. Finally, we merge these fragments into sentences. The result of this step is a set of sentence patterns for the specification of requirements concerning the quality attribute.

In summary, given a quality attribute, the approach derives a context-independent content model based on qualitative literature analysis, provides a clear and explicit definition of the individual content elements, performs a customization for a given organizational context, and provides a means for practitioners to specify requirements concerning the quality attribute for a given organizational context.

III. Application to Performance Requirements

In this section, we give guidance how the individual steps can be performed. As a running example, we use *performance requirements*, or *performance/efficiency requirements* as they are called in the ISO 25010. In our running example, we explicitly focus on externally visible performance and exclude internal performance (sometimes also called efficiency), which describes the capability of a product to provide performance in relation to the use of internal resources.

A. Step 1: Context-independent Definition

The goal of this step is to create a comprehensive content model that covers all content elements and relationships that we need to specify requirements concerning the quality attribute. In the last section, we proposed to use qualitative literature analysis for this purpose.

For our running example, we reduced the set of relevant literature to classifications and categorizations of non-functional and quality requirements (and software and systems quality models). Fig. 5 gives a high-level overview of the results of the literature review for our running example. In particular, literature differentiates three types of performance requirements: *Time behavior* requirements, *Throughput* requirements, and *Capacity* requirements. Time behavior requirements specify *fixed time constraints* like *"The operation Y must have an average response time of less than x seconds"*, throughput

Performance			
Time Behavior	**Throughput**	**Capacity**	**Aux. Conditions**
- Points in time - Response time - Reaction time - Turnaround time - Time intervals - Latency - Time constraints	- Rate of transactions - Data volume per unit of time - Reaction speed - Processing speed - Operating speed	- Maximum limits - Concurrent users - Communication bandwidth - Size of database or storage	- Measurement location - Measurement period - Load - Platform - Scope of measurement - Measurement assumption

Fig. 5. Running example: overview of performance.

requirements specify *relative time or resource constraints* like *"The system must have a processing speed of x requests/second"*, and capacity requirements specify *limits of the system* like *"The system must support at least x concurrent users"*. Furthermore, literature defines further aspects related to performance requirements that apply for all three types of performance requirements. We call these aspects *auxiliary conditions* (e.g., the location of a measurement).

We then coded the results of the literature review as suggested by Grounded Theory [10] to assemble a conceptual model of the quality attribute in form of a content model. The resulting content model contains content elements of the quality attribute and relations between them. Furthermore, we added content elements that apply to requirements in general (e.g., the scope of a requirement). The result of this step is a content model for performance that ideally is a superset of all performance aspects mentioned in literature. A detailed description of the resulting content model for performance can be found in Eckhardt et al. [8].

B. Step 2: Precise Definition

Given the content model from the previous step, the goal of this step is to provide a precise definition for the individual content elements of the content model (see step ② in Fig. 2).

We started with an informal definition of the content elements by a simple glossary. To create the glossary, we iterated through the content model. We discussed the meaning of each content element in several refinement rounds. Table I shows a part of the glossary for performance requirements.

For a more formal definition of the content elements, we mapped them to FOCUS, a formal modeling theory [9], [11] and its probabilistic extension as introduced by Neubeck [12] because we found in a previous study [13] that performance requirements describe probabilistic and timed behavior of a system.

C. Step 3: Context-dependent Customization

In the third step, our goal is to achieve a customization of the content model for a given operational context (see step ③ in Fig. 2). To achieve this, we follow the idea of activity-based quality models [6], [7] and use the context-independent content model as input for the creation of the activity-based quality model for the given context. Here, we first consider all stakeholders and analyze their development activities that take

TABLE I
GLOSSARY ENTRIES FOR PERFORMANCE REQUIREMENTS.

Content Element	Definition (Syntax and Semantics)
Modality	The modality of a requirement may be one of *exclusion*, *obligation*, *enhancement*. If the modality of a requirement is *exclusion*, the predicate described by the requirement must not hold, if it is an *obligation*, it must hold, and if it is an *enhancement*, it may hold.
Time Behavior Requirement	A Time Behavior requirement has a *Time Quantification* and a *Time Property*. A time behavior requirement demands that the time property complies with the time quantification.
Time Quantification	A Time Quantification has a quantification, i.e. one of $\leq, <, =, >, \geq$ a time value, i.e. a natural number, and a time unit. An example would be "$\leq$ 100ms".
Time Property	The time property may be one of *Response Time*, *Processing Time*, *Latency*.
...	...

requirements of the quality attribute as input, such as *design test* of the *test designer* in case of performance requirement. We identify necessary and important content elements that these requirements must contain to complete the development activities efficiently and effectively. We accordingly classify content elements, marking crucial content elements as mandatory and the contributing content elements as optional. The result of this step is a content model that is customized for a given operational context and each content element is justified by at least one development activity.

For our running example, we used *testing activities* as described in the (rational) unified process (RUP) [14]. For each of the stakeholders' activities, we identified the corresponding necessary content elements from the content model to complete the activity efficiently and effectively. As the description of the activities in the RUP is rather high-level and does not provide detailed insights about the required artifacts for an activity, we performed an in-depth analysis of the description of the respective activities. Then, in a pair of researchers, we discussed the activities and identified the necessary content elements of a requirement for that activity: We marked a content element as necessary when we agreed that its absence would require a stakeholder to invest additional effort for completing the activity or would even make the activity impossible. Table II shows the resulting mapping between the necessary content elements and the activities. For example, for the activity *design test* by the *test engineer*, the *time/throughput/capacity property* of the requirement is necessary as its absence would make it impossible to set up an adequate test environment. Furthermore, the scope of the requirement is necessary for the activity *plan test*, as the test engineer needs this information for assigning the test to a person/team responsible. The final context-dependent content model can be found in Eckhardt et al [8].

D. Step 4: Concretization

In the final step, we aim to provide a means for practitioners to specify requirement concerning the quality attribute. To achieve this, we propose to derive a set of sentence patterns from the context-dependent content model (see step ④ in Fig. 2). In particular, for each of the content elements in the content model, we derive a *sentence fragment*. The sentence fragment is intended to represent the meaning of the content element as closely as possible. Finally, we merge these fragments into sentences. The result of this step is a set of sentence patterns for the specification of requirements concerning the quality attribute.

In our running example, we iterated through the set of content items in a pair of researchers and discussed how to adequately represent this content element in terms of a sentence fragment. The complete set of patterns can be found in Eckhardt et al [8]. An exemplary instance of a sentence is

> The system must have a processing time of < 10 ms between event ''receiving a request'' and event ''answering a request'', when under a maximal load. Measurement takes place on production hardware. Included is browser render time.

IV. DISCUSSION

In this section, we discuss limitations and threats and direct implications of our approach.

A. Limitations and Threats

The quality of the results of our approach heavily depends on how the individual steps are performed. Furthermore, all steps require a high amount of creative and qualitative work and thus may be error-prone. To mitigate this threat, we provided guidance in this paper that shows how to perform the individual steps on the example of performance requirements. We described how we performed the individual steps and the respective results in detail and provided hints how to ensure quality.

1) Context-dependent Definition: In the first step of our approach, there are some threats that affect the generalizability and applicability of the results. The initial collection of literature may miss some important work, the extraction of models may miss models or include unimportant models, and finally the coding and consolidation of the models may lead to inconsistent or inadequate models. We try to mitigate these threats by using a structured and reproducible approach (e.g., a structured literature review) and by performing the extraction and coding steps in a pair of researchers (researcher triangulation). Furthermore, we suggest to validate the resulting models with quality requirements from practice or perform validating interviews with practitioners.

2) Precise Definition: The goal of the second step is to create a precise definition such that we reduce misunderstandings. We propose to use either a glossary or a formal definition by means of a system modeling theory. However, in both cases, it is a highly challenging and creative activity and the individual content elements can be contradictory or inadequate. To mitigate this, we propose to perform a validation in form of interviews with researchers as well as with practitioners.

TABLE II
RUNNING EXAMPLE: NECESSARY CONTENT ELEMENTS TO COMPLETE DEVELOPMENT ACTIVITIES EFFICIENTLY AND EFFECTIVELY.

Stakeholder	RUP activities	Necessary content element
Test designer	Plan Test	Modality, Scope
	Design Test	Scope, Time Property, Throughput Property, Capacity Property
	Implement Test	Scope, Quantifier, Time Property, Throughput Property, Capacity Property, Time Quantification, Time Value, Unit, Throughput Quantification, Change Value, Change Object, Capacity Quantification, Capacity Value, Capacity Object
	Evaluate Test	Scope, Quantifier, Time Property, Throughput Property, Capacity Property, Time Quantification, Time Value, Unit, Throughput Quantification, Change Value, Change Object, Capacity Quantification, Capacity Value, Capacity Object
System tester	Execute System Test	Scope, Quantifier
Performance tester	Execute Performance Test	Scope, Quantifier
Designer	Design Classes and Packages	Scope, Time Property, Throughput Property, Capacity Property
Implementer	Implement Components and Subsystems	Scope, Time Property, Throughput Property, Capacity Property, Time Quantification, Time Value, Unit, Throughput Quantification, Change Value, Change Object, Capacity Quantification, Capacity Value, Capacity Object

3) Context-dependent Customization: The result of this step is highly dependent on how the customization is performed. We propose to use activity-based quality models that try to make the relation between activities, artifacts, and quality attributes explicit. However, the quality of the results still depends on the level of detail and adequacy of the activity-based quality model. In our running example, we build our customization based on the activities for testing as described in the RUP. However, the description of these activities was on a very high level of detail, and thus, we discussed each activity in a pair of researchers. In summary, to mitigate this threat, we propose to either use a detailed activity-based quality model or perform a cross validation or researcher triangulation.

4) Concretization: In the final step, the creation of sentence patterns is straight-forward. However, the quality of the overall approach depends on how well practitioners can apply the sentence patterns to requirements and are how much they are willing to use the patterns. To mitigate this, we propose to validate the resulting patterns with quality requirements in practice and furthermore conduct interviews with practitioners concerning their willingness to use the patterns.

B. Syntactic Analyses: Challenging Incompleteness

Besides the constructive nature of our approach, we can further support analytic quality assessment with syntactic analyses. For example, through such patterns, we can syntactically detect that a textual individual requirement does not document a specific information, such as the location for a performance requirement in our example (i.e., the completeness of the individual requirement).

One benefit of our approach is that it creates a context-dependent content model for a given quality attribute. The model is context-dependent in the sense that for a given context, the content model contains all necessary information to complete subsequent activities efficiently and effectively. We can now leverage this fact to support syntactic analyses and introduce a notion of (syntactic) completeness for requirements of this type.

We then define the completeness of requirements for a given quality attribute with respect to the presence of all mandatory content elements in the context-dependent content model. In particular, we call a requirement complete if all mandatory content elements are present in the textual representation of the requirement. There are three cases for the presence of mandatory content in the textual representation of a requirement:

- The requirement **does not contain** the content. For example, in case of a performance requirement stating *"The delay between [event A] and [event B] shall be short"*, the content regarding the quantifier is not contained.
- The requirement **implicitly** contains the content. With implicit, we mean that the content is contained in the requirement, but we need to interpret the requirement to derive the content. For example, in case of a performance requirement stating *"The delay between [event A] and [event B] shall typically be 10ms"*. In this case, regarding the quantifier, we can interpret "typically" as "median".
- The requirement **explicitly** contains the content. With explicit, we mean that the content is contained without interpretation. For example, in case of a performance requirement stating *"The delay between [event A] and [event B] shall have a median value of 10ms"*. In this case, regarding the quantifier, the content is explicitly contained.

We can now derive the following definitions for strong and weak completeness and for incompleteness of requirements of a given quality attribute:

Definition (Strong Completeness)**.** *A requirement of a given quality attribute is **strongly complete**, if all mandatory content elements (w.r.t the context-dependent content model of the attribute) are **explicitly** contained in its textual representation.*

Definition (Weak Completeness)**.** *A requirement of a given quality attribute is **weakly complete**, if all mandatory content elements (w.r.t the context-dependent content model of the attribute) are **explicitly or implicitly** contained in its textual representation.*

Definition (Incompleteness)**.** *A requirement of a given quality attribute is **incomplete**, if at least one mandatory content*

*elements (w.r.t the context-dependent content model of the attribute) is **missing** in its textual representation.*

We argue that this definition of completeness for requirements of a given quality attribute can be used to detect incompleteness and thus to pinpoint to requirements that are hard to comprehend, implement, and test. For example, requirements of class *incomplete* are not testable at all, requirements in class *weakly complete* need to be interpreted by the developer and tester and therefore bear the risk of misinterpretations, and requirements in class *strongly complete* contain all content necessary to be implemented and tested. Thus, we argue that our approach further provides a helpful and actionable definition of completeness for quality requirements. This definition of completeness can then be used to support analytic as well as constructive quality control.

C. Analyses of the Content of Quality Requirements

Besides the assessment of completeness, one can further leverage our approach to analyze the content of quality requirements in practice. Our approach results in a context-independent content model for a given quality attribute and in a context-dependent content model for that attribute. The context-independent content model provides a general definition of the content elements of the quality attribute and the context-dependent model provides a justification for each content model.

We can now analyze textual quality requirements and map the content elements found in the requirements to the content model. If we have a sufficiently large data set, we can now analyze observations and draw conclusions about the content elements of quality requirements in general. For example, a common point of view of quality requirements is that they are cross-functional and consider the system as a whole. When analyzing performance requirements, we also included the scope of a requirement in the content model. This allows us to quantitatively analyze the distribution of the scope of performance requirements found in practice.

D. Implications for Industry

Our approach is a step towards increasing the completeness of quality requirements. Not only the concretization via sentence patterns could be easily implemented in a requirements authoring or management tool. Such a tool may provide instant feedback to the requirements engineer about missing or optional content elements, similar to requirements smells [15], [16]. Furthermore, the tool might check the terms used in a requirement with respect to an underlying domain model. The tool could then uncover terms that are neither part of the consolidated terminology nor defined through the pattern semantics.

An additional benefit of our approach is that it makes content in natural language requirements explicit and traceable through content elements. This allows connecting specific content elements of requirements with specific content elements in related artifacts such as test cases or components within the implementation. Updates within requirements may then be propagated directly to corresponding test cases for example, making maintenance activities more efficient and effective.

V. Related Work

There is a variety of work on requirement patterns in RE. Franch et al. [17] present a metamodel for software requirement patterns. Their approach focuses on requirement patterns as a means for reuse in different application domains and is based on the original idea of patterns by Alexander et al. [18], i.e., each pattern describes the core of a solution of a problem that occurs over and over again. In particular, the PABRE framework contains a catalogue of 29 QR patterns [19], 37 non-technical patterns [20], and a method for guiding the use of the catalogue in RE [21]. Their approach for creating the patterns catalogue is similar to ours, as it is also based on requirements literature and a content analysis. However, they provide solutions for recurring problems while our sentence patterns provide a means for the specification of customized requirements.

Supakkul et al. [22] present four kinds of NFR patterns for capturing and reusing knowledge of NFRs and apply these patterns in a case study. Their patterns and, in particular, the *objective pattern* can be used to identify important NFRs for a context or capture a specific definition of an NFR from the viewpoint of a stakeholder. Thus, their patterns define important content elements of a quality attribute in terms of soft goals, which is similar to our context-dependent content model [23], [24]. Our approach provides a structured way to define and customize these content elements and also provides sentence patterns to specify requirements. However, their patterns can be used to define the specific quality attribute but furthermore provide solutions and alternatives and thus go one step further into the architecture or design of a system. Our approach focuses on definition, customization, and concretization of requirements concerning a specific quality attribute.

Withall [25] presents a comprehensive pattern catalogue for natural language requirements in his book. The pattern catalogue contains a large number of patterns for different types of requirements. In contrast to their work, in our approach, we derive patterns from literature and customize them to a specific application context. Almeida Ferreira and Rodrigues da Silva [26] introduce RSL-PL, a language for the definition of requirements sentence patterns. Their pattern definition language can be used to represent our sentence patterns.

Kopczyńska and Nawrocki [27] present a method for eliciting non-functional requirements, which is composed of a series of brainstorming sessions driven by the ISO 25010 quality sub-characteristics. Elicitation is supported by Non-functional Requirements Templates (NoRTs), which are statements that require some completion to become a well-formulated NFR. Similar to our sentence patterns, the authors differentiate between core parts, parameters, and optional parts within the templates. The sentence patterns derived by our approach are additionally adapted to specific classes of quality requirements.

Mylopoulos et al. [23] propose a comprehensive framework for representing and using QRs in the development process.

Similar to our approach, they propose a means to integrate QRs in the development process, however, they do not provide a structured approach for explicitly stating the content elements for specifying requirements concerning quality attributes and do not provide a means for specifying QRs.

VI. Conclusion

We provided an approach that—given a quality attribute as input—provides a means to precisely and explicitly defines the content elements that are needed to specify requirements concerning this quality attribute, and provides a means for practitioners to specify these requirements for a given organizational context based on sentence patterns. The approach consists of four steps:

1) **Context-independent Definition:** Relevant content elements are identified by means of qualitative literature analysis and coding.
2) **Precise Definition:** The resulting content elements are precisely defined by e.g., a glossary or formalization by means of a mapping to a system model.
3) **Context-dependent Customization:** The content elements are customized to a given organizational context by using the idea of activity-based quality models.
4) **Concretization** Sentence patterns are used as a means for practitioners to specify requirements concerning the quality attribute.

As our main goal was to provide guidance for the application of our approach, we furthermore discussed threats to validity and lessons learnt while instantiating it for performance requirements. Finally, we argue that our approach is applicable for performance requirements and besides its constructive nature, provides a means for various statical analyses, as for example completeness analyses.

We are planning to apply our approach for further quality attributes, in particular, for availability as a direct next step. As a broader vision, we are planning to unify the resulting content models in one content model for quality requirements.

Acknowledgements

We would like to thank M. Broy, S. Eder, and M. Junker for their helpful comments on earlier versions of this work. This work was performed within the project Q-Effekt; it was partially funded by the German Federal Ministry of Education and Research (BMBF) under grant no. 01IS15003 A-B. The authors assume responsibility for the content.

References

[1] K. Pohl, *Requirements Engineering: Fundamentals, Principles, and Techniques*, 1st ed. Springer, 2010.

[2] A. Borg, A. Yong, P. Carlshamre, and K. Sandahl, "The bad conscience of requirements engineering: An investigation in real-world treatment of non-functional requirements," in *3rd Conference on Software Engineering Research and Practice in Sweden (SERPS)*, 2003.

[3] R. B. Svensson, T. Gorschek, and B. Regnell, "Quality requirements in practice: An interview study in requirements engineering for embedded systems," in *Requirements Engineering: Foundation for Software Quality*, ser. Lecture Notes in Computer Science. Springer, 2009, vol. 5512.

[4] L. Chung and B. A. Nixon, "Dealing with non-functional requirements: three experimental studies of a process-oriented approach," in *17th International Conference on Software Engineering (ICSE)*, 1995.

[5] M. Glinz, "On non-functional requirements," in *15th IEEE International Requirements Engineering Conference (RE)*, 2007.

[6] H. Femmer, J. Mund, and D. Méndez Fernández, "It's the activities, stupid!: A new perspective on RE quality," in *2nd International Workshop on Requirements Engineering and Testing (RET)*, 2015.

[7] F. Deissenboeck, S. Wagner, M. Pizka, S. Teuchert, and J. Girard, "An activity-based quality model for maintainability," in *23rd IEEE International Conference on Software Maintenance (ICSM)*, 2007.

[8] J. Eckhardt, A. Vogelsang, H. Femmer, and P. Mager, "Challenging incompleteness of performance requirements by sentence patterns," in *24th IEEE International Requirements Engineering Conference (RE)*, 2016.

[9] M. Broy and K. Stølen, *Specification and development of interactive systems: focus on streams, interfaces, and refinement*. Springer, 2001.

[10] S. Adolph, W. Hall, and P. Kruchten, "Using grounded theory to study the experience of software development," *Empirical Software Engineering*, vol. 16, 2011.

[11] M. Broy, "Multifunctional software systems: Structured modeling and specification of functional requirements," *Science of Computer Programming*, vol. 75, no. 12, 2010.

[12] P. Neubeck, "A probabilistic theory of interactive systems," Ph.D. dissertation, Technische Universität München, 2012.

[13] J. Eckhardt, A. Vogelsang, and D. Méndez Fernández, "Are non-functional requirements really non-functional? An investigation of non-functional requirements in practice," in *38th International Conference on Software Engineering (ICSE)*, 2016.

[14] I. Jacobson, G. Booch, J. Rumbaugh, J. Rumbaugh, and G. Booch, *The unified software development process*. Addison-Wesley Reading, 1999.

[15] H. Femmer, D. Méndez Fernández, E. Juergens, M. Klose, I. Zimmer, and J. Zimmer, "Rapid requirements checks with requirements smells: Two case studies," in *1st International Workshop on Rapid Continuous Software Engineering (RCoSE)*, 2014.

[16] H. Femmer, D. Méndez Fernández, S. Wagner, and S. Eder, "Rapid quality assurance with requirements smells," *Journal of Systems and Software*, 2016.

[17] X. Franch, C. Palomares, C. Quer, S. Renault, and F. De Lazzer, "A metamodel for software requirement patterns," in *Requirements Engineering: Foundation for Software Quality*, ser. Lecture Notes in Computer Science. Springer, 2010, vol. 6182.

[18] C. Alexander, *The Timeless Way of Building*. Oxford Books, 1979.

[19] S. Renault, Ó. Méndez-Bonilla, X. Franch, and C. Quer, "A pattern-based method for building requirements documents in call-for-tender processes," *International Journal of Computer Science & Applications*, vol. 6, no. 5, 2009.

[20] C. Palomares, C. Quer, X. Franch, C. Guerlain, and S. Renault, "A catalogue of non-technical requirement patterns," in *2nd International Workshop on Requirements Patterns (RePa)*, 2012.

[21] X. Franch, C. Quer, S. Renault, C. Guerlain, and C. Palomares, "Constructing and using software requirement patterns," in *Managing Requirements Knowledge*. Springer, 2013.

[22] S. Supakkul, T. Hill, L. Chung, T. T. Tun, and J. C. S. do Prado Leite, "An NFR pattern approach to dealing with NFRs," in *18th International Requirements Engineering Conference (RE)*, 2010.

[23] J. Mylopoulos, L. Chung, and B. Nixon, "Representing and using nonfunctional requirements: A process-oriented approach," *IEEE Transactions on Software Engineering*, vol. 18, no. 6, 1992.

[24] L. Chung, B. A. Nixon, E. Yu, and J. Mylopoulos, *Non-functional requirements in software engineering*. Springer Science & Business Media, 2012, vol. 5.

[25] S. Withall, *Software Requirement Patterns*, 1st ed. Redmond, WA, USA: Microsoft Press, 2007.

[26] D. de Almeida Ferreira and A. Rodrigues da Silva, "RSL-PL: A linguistic pattern language for documenting software requirements," in *3rd International Workshop on Requirements Patterns (RePa)*, 2013.

[27] S. Kopczyńska and J. Nawrocki, "Using non-functional requirements templates for elicitation: A case study," in *4th International Workshop on Requirements Patterns (RePa)*, 2014.