Tailoring and Evaluating Non-Functional Interests Towards Task-Oriented Functional Requirements

Philipp Haindl
Department of Business Informatics
Johannes Kepler University Linz
Linz, Austria
philipp.haindl@jku.at

Reinhold Plösch
Department of Business Informatics
Johannes Kepler University Linz
Linz, Austria
reinhold.ploesch@jku.at

Christian Körner
Corporate Technology
Siemens AG
Munich, Germany
christian.koerner@siemens.com

***Abstract*—Without a specific functional context, non-functional requirements can only be approached as cross-cutting concerns and treated uniformly across all features of an enterprise system. This neglects, however, the heterogeneity of non-functional requirements that arises from the domains of stakeholders and the distinct functional scopes these systems, which mutually influence how these non-functional requirements have to be satisfied. Earlier studies showed that the different types and objectives of non-functional requirements result in either vague or unbalanced specification of non-functional requirements. We propose a task analytic method for eliciting and modeling user tasks and the stakeholders' pursued interests towards the enterprise system. Stakeholder interests are structurally related to user tasks and each interest is specified individually as a quantitative constraint for a specific user task. These constraints can automatically be evaluated throughout the system's lifecycle to assure that the respective stakeholder interest is fulfilled. Eventually, this allows to proactively counteract violations of constraints and thus stakeholder interests. We propose a structured method, intertwining task-oriented functional requirements with non-functional stakeholder interests to specify constraints on the level of user tasks. We also present results of an exploratory case study with domain experts, which reveals that our task modeling and interest-tailoring method facilitates shared understanding of stakeholder interests, clarity and quality of software constraints, prioritization of engineering efforts, and the impact of stakeholder interests on functional requirements.**

***Index Terms*—Non-Functional Stakeholder Interests; Requirements Negotiation; Task Modeling; Constraint Specification; Requirements Evaluation.**

I. Introduction

Balancing functional and non-functional requirements (NFRs) articulated from stakeholders is a challenging endeavor in software projects of any scale. Also, due to the primarily engineering-oriented scope of NFRs, the practical relevance of business, strategic, operational, legal, and privacy interests is often neglected during requirements specification. As a result, these types of requirements are neither monitored nor evaluated in the software lifecycle.

We define the term *stakeholder interest* to emphasize a broader understanding of NFRs and to also capture relevant non-engineering related qualities of software, that must be addressed. In industrial settings, these stakeholder interests are typically elicited and defined on a high level as cross-cutting concerns. This however does not take into account the relevance, applicability, and characteristic of a specific interest for a certain software feature. As the efforts required to satisfy stakeholder interests differ between features, this vagueness in requirements specification results in undetected non-functional dependencies between components and features in development, as well as increased efforts during software operation [1]. Functional units of works can be better elicited through taking a user perspective and focusing on the most relevant tasks that will be performed with the software. Contrarily to isolated features, tasks can be seen as functional units that support the users' goals and also show the required interactions between software features. During requirements elicitation, this also makes the individual relevance of competing NFRs for a single user task more tangible for all stakeholders. In our work, we understand constraints as refinements of NFRs. By specifying these constraints for a user task using quantitative measures, constraints can be automatically validated for fulfillment. Monitoring the fulfillment of stakeholder interests facilitates requirements negotiation [2] and assessment of tradeoffs in satisfying constraints [3]. Also, this allows buyers of customized individual software products and enterprise systems (ES) to evaluate if non-functional requirements are implemented by vendors as contractual agreed.

This paper presents the TAICOS *(Task-Interest-Constraint Satisfycing)* method as a modification of hierarchical task analysis [4] for eliciting functional and non-functional requirements and monitoring the fulfillment of these requirements for enterprise systems. This facilitates taking appropriate counteractions throughout the lifecycle to meet the stakeholders' requirements on a very detailed level. The contributions of the method are twofold: (a) it provides requirements engineers a structured sequence for eliciting functional requirements from user tasks and non-functional requirements from stakeholder interests towards enterprise systems; (b) it provides a compact language for refining these interests into quantified constraints. These constraints can eventually be operationalized through respective instruments that acquire typical measures for specific constraints, e.g., the execution time or maintainability characteristics of the task's software implementation. The prime objective of these constraints is to *satisfice* the original interest, i.e., to *satisfy* an interest *sufficiently* and not better than required. The operationalization and evaluation of these constraints eventually gives engineering teams important feedback to practically handle these constraints in software design, implementation,

Copyright © 2020 for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

CEUR-WS.org/Vol-2581/aesp2020paper1.pdf

and operation. Also, the TAICOS method is complemented by a dedicated operational software quality model [5] to facilitate evaluating the fulfillment of interests in an automated manner in a DevOps context.

The remainder of this paper is organized as follows. In Section II we present a motiviating example to stress the practical importance of our method. An overview of related work and the demarcation of our method to other approaches are given in Section III. Following, Section IV describes our method to eliciting, modeling, and tailoring functional requirements and stakeholder interests. Subsequently, Section V presents how our method can be utilized in a DevOps context for automatically evaluating the fulfillment of stakeholder interests. Following, Section VI presents selected evaluation results of a case study with domain experts in which we examined particular aspects of the method. Following, we describe the possible threats to validity in Section VII before we conclude our work and sketch possible directions for future work in Section VIII.

II. Motivating Example

To motivate our method to tailoring stakeholder interests to user tasks and specifying concrete constraints therefor, let us consider a simple example. Your company runs a local book store and you want to provide your customers an online application. Basically this application shall allow customers to search books and write reviews about them, update credit card information and change their shipping address. Your company and the customers most likely will have diverging expectations regarding individual characteristics of these tasks.

But how concretely could these expectations differ among the user tasks? In order to concentrate your software maintainance efforts you could require higher software quality standards for functionality that is part of your revenue stream. In this example this particularly affects tasks where users search for books or update their credit card information, as both tasks are part of the core purchase process. On the contrary, the quality characteristics of the software components allowing the users to write book reviews might be less important for you. Taking the customer perspective it most likely will be important that your application is responsive even if many users concurrently use it. Particulary the response time of your application will be relevant when customers search for a book and less important if they write a book review.

As can be depicted from this example, while there can be general stakeholder interests that the software shall be maintainable or responsive, it actually depends on the concrete user task how an interest concretely manifests itself. Our method facilitates tailoring qualitative stakeholder interests towards user tasks and specifying quantitative constraints that can be evaluated during the lifecycle of an enterprise system.

III. Related Work

Goal- [6], [7] and task-driven elicitation techniques [1], [8] are effective approaches for functional requirements elicitation. As such, they do not support stakeholders in specifying suitable NFRs for enterprise systems. According to Fotrousi et al. [9], the main limitation of goal models is that they make it difficult for stakeholders to understand the impact of unmet NFRs towards a goal during requirements specification.

Riegel et al. [10] present a prioritization method which categorizes non-engineering related NFRs by project-related, financial, customer, operational business performance or business-strategy related benefits. Their work underline the broader notion of NFRs to also capture non-technical qualities of software. Karlsson et al. [11] investigate the applicability of goal models for market-driven software development and stressed the advantages of goal-centered feature elicitation. The authors conclude that these types of models facilitate stakeholder participation during requirements engineering.

The third stream of research examines the different satisfaction criteria of NFRs in the context of user tasks. Zubcoff et al. [12] propose to specify soft goals in the context of user tasks to assist requirements engineers in evaluating trade-offs between NFRs. The authors also underline that NFRs shall be specified individually for functional requirements to improve end-user satisfaction. Ameller et al. [13] present a survey among practitioners about the practice of NFR specification in model-driven design. The interviewees argued that NFRs are not only difficult to specify through models, but even difficult to discover and explicate in measurable terms. As a result of this imprecise definition, the fulfillment of NFRs usually can only be evaluated and counteracted very late in the software development process. An interview study of Svensson et al. [14] also shows that NFRs typically have lower priority than functional requirements in practice and often are not specified in early stages of development. This lack of integration between functional and non-functional requirements can result in prolonged time-to-market and cost overruns in many software projects [15]–[17].

In summary, recent research had elaborated the importance of balancing functional and NFRs in understanding technical qualities. Though, no structured method has yet been presented that captures stakeholders interests outside the technical domain. The approach presented in this paper provides a structured method for eliciting and specifying constraints from NFRs for individual functional requirements. Also, it is complemented by a compact constraint language and operationalization framework that allows to evaluate the fulfillment of the respective stakeholder interests in an automated manner during the software lifecycle.

IV. The TAICOS Method

Particularly in the context of enterprise systems, with users executing tasks as activities of business processes, it is important to elicit functional requirements from the goals that users pursue by using the software [18]. Hence, the TAICOS method hierarchically decomposes functional blocks of software from the users' goals into concrete tasks. In contrast to eliciting isolated features, this gives a more comprehensive picture about how these different functional blocks are related to each other so that the user can most

effectively achieve the pursued goal. To facilitate a shared understanding among requirement engineers of the users' tasks and goals, each step of the method is repeated until a common understanding has been achieved. Figure 1 illustrates the eight steps of the method, starting with capturing user tasks and collecting stakeholder interests.

A. Capturing User Tasks

① **Elicitation:** To elicit tasks, we rely on hierarchical task analysis [4] with some modifications to carve out the functional scope of the elicited actions. Hierarchical task analysis breaks down a task into goals, subgoals, plans, and operations, focusing on the structure and decomposition of the task into a hierarchy of subtasks in sequential order of tasks so that the goal can be attained. According to this notion, a *task* is a sequence of actions that people perform to attain a goal. The elicitation of tasks in our method follows a sequential procedure that provides structured guidance to detail the functional scope and connections between tasks.

- **Step 1: Define task under analysis.** Which task should be analyzed and what is its objective? Scope boundaries must be clearly defined prior to analysis, typically driven by the underlying business model.
- **Step 2: Collect data for task analysis.** This collection comprises data about task execution, dependencies between tasks, constraints. It originates from interviewing subject matter experts or key users.
- **Step 3: Determine the overall task goal.** This will become the root goal of the hierarchy, i.e., the starting point for decomposition.
- **Step 4: Determine subtasks.** The predecessor task goal will then be used to derive subtasks necessary for achieving the superordinate goal.
- **Step 5: Identify task details.** Derived from the goal of each subtask, identify the intentions of the user in attaining the goal and the resulting responsibilities of the system to support these intentions. We elaborate this step in more detail below.
- **Step 6: Define execution plans.** Execution plans organize how to reach the goal of the task by modeling execution order and dependencies between the subtasks. The objective is to define how the subtasks relate to each other so that the goal of the task can be achieved.

While hierarchical task analysis allows infinite refinement of tasks to the point that tasks are purely operational for a user, our method only allows refinement of tasks by means of *task details* tables. This forces the development team to define the granularity of tasks upfront. If the refinement is still too vague to be operational, a separate model should be created for refinement. This assures that tasks and constraints can be properly treated and described at the refined level.

For the structured elicitation of task details, our method offers two perspectives: (1) *user intentions*, the user's interactions during execution of the task; and (2) *system responsibilities* to support these user intentions through the system. Both perspectives are compared with each other in tabular form and refined with *pre-* and *postconditions*, as well as *information objects*, which describe the information generated or required for task execution. The task detailing step helps to map the fine-grained intentions of the user to suitable functional requirements by carving out the minimal and satisfactory technical solution to execute the task effectively.

② **Modeling:** In the next step, the control flow and decision points between the subtasks are modeled. Figure 2 shows an exemplary task of searching for a book at an online store. This example was also used for presenting our method in the expert interviews.

Each subtask is modeled as a rectangular gray box labeled with the corresponding *user intention* and incorporates one or multiple rounded rectangles (i.e., *system responsibilities*) that reflect the required software functionality. Information objects are denoted by colored rectangles, with the direction of the arrows indicating whether the information object is generated or consumed by the respective subtask. The objective of this modeling is to outline the control flow among the subtasks for all stakeholders and to facilitate sketching the required software features. In this work we refer to features as concrete software implementations [19] which realize the actions that are required to implement a task. Also, the granularity of the actions executed within a subtask can be chosen arbitrarily to foster a common understanding of the scope and goal of the features among the involved stakeholders. In contrast to process modeling, this type of modeling primarily focuses on the flow among the activities and also relates them to the technical counterparts needed for their execution. For the sake of simplicity, we abstain to describe conditional loops or concurrent executions of subtasks in our modeling method.

③ **Prioritization:** Next, only those user tasks are selected for the subsequent steps, which are most important (following the idea of a minimum viable product), bear a competitive advantage, or have been selected by an agile team for development in the next program increment. Typically, the question of importance can only be answered by considering the underlying business model. Also, the required assessment of the tasks' value contribution prevents specifying details or exhaustive constraints for rarely executed tasks.

B. Collecting Stakeholder Interests

④ **Elicitation:** Due to their primary technical focus, the notion of NFRs does not satisfactorily cover non-technical objectives of stakeholders having an impact on a software system. This specially comprises all objectives that must be considered continuously throughout the software lifecycle, from software development and operation to its decommissioning. Existing classification schemes [20], [21] for these non-technical objectives, which we understand as *stakeholder interests*, also reflect their relevance for software quality and need for precise specification. These stakeholder interests are

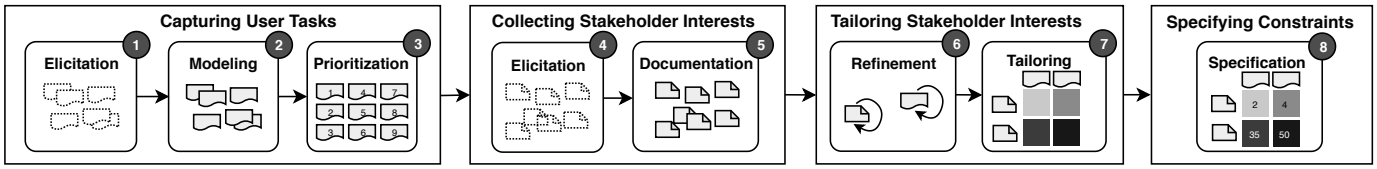


Figure 1: The TAICOS method defines a structured sequence from capturing user tasks to collecting and tailoring stakeholder interests for specifying task-dependent constraints.

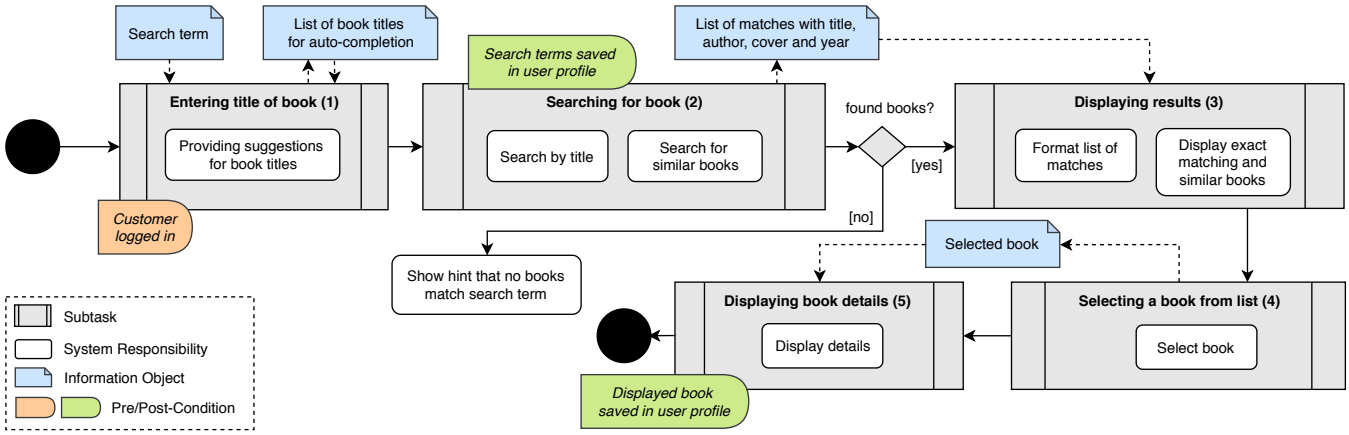


Figure 2: Modeling Subtasks, System Responsibilities, Information Objects and Conditional Flows.

still unsatisfactorily covered by existing, solely technically oriented standards such as ISO/IEC 29148:2011 [22] or ISO/IEC 25030:2007 [23], which often results in NFRs being only vaguely elicited.

The objective of this step is to capture all stakeholders' quality-related, operational, business, legal, and other non-behavioral interests that influence how the later software system needs to support the tasks of the users.

⑤ **Documentation:** In this step, the elicited stakeholder interests are documented informally to express the stakeholders' objectives and expectations towards the software system. It is important to document each stakeholder interest in a manner comprehensible for all involved stakeholders to foster shared understanding and also to later assess its individual relevance. In this step of the method an interest does not need to be documented on a quantitative basis. The concrete tailoring and deriving of constraints for each user tasks is done in a subsequent step. Stakeholder interests can be documented in the following ways:

- "The software must be responsive to user inputs."
- "... handle peaks of concurrent users."
- "... be deployable automatically."
- "... recover quickly after outages."
- "... be operated in the cloud."

Our method explicitly separates interest elicitation from documentation. The former being conducted by all stakeholders in a brainstorming like manner striving to unveil as many possibly relevant interests that could impact the software lifecycle; the latter just to document the results from the elicitation step.

C. Tailoring Stakeholder Interests

⑥ **Refinement:** Similar to the refinement of user tasks, also stakeholder interests are iteratively refined until they show a delimited and comprehendible scope for the later tailoring. Refined stakeholder interests again are documented conjointly with the relevant stakeholders. This is done to cross-check that there is common understanding about the interest even after its refinement. Stakeholder interests which are too large in scope are decomposed down to a suitable level. The overall objective of this step is to prevent ambiguities about an interest's scope among the stakeholders. Uniform comprehension about the scope of an interest is the prerequiste for its effective tailoring and the specification of suitable constraints therefore.

⑦ **Tailoring:** In this step, the stakeholder interests and user tasks are analyzed pairwise to evaluate the relevance of an interest and how it can be satisfied in each narrow task context. This detailed analysis also assures that for all elicited stakeholder interests the relevant measures can later be operationalized in the context of individual user tasks.

Stakeholder interests and user tasks are then related to each other in a two-dimensional *task-interest matrix*. Following, each interest is tailored individually to each user task so that it can be fulfilled exactly for the respective user task. Resulting from this tailoring is a qualitative ordering of the cells indicating the relevance of the interest for the respective user task, e.g., through coloring cells darker gradually with relevance. The tailoring and qualitative ordering of the interests' relevance is

shown in the (task-interest matrix, top) in Figure 3. At that point of the method no quantitative fulfillment criteria for the stakeholder interests are defined. The focus of this step is on getting a common understanding about the relevance of an interest for a particular user task. As a practical example, the interest that *"the software must be responsive to user inputs"* might be subjected to different qualitative expectations depending on the concrete user task. Responsiveness might be much more important for a user when searching for a book than when writing a book review, but less important when changing the shipping address. As a further example from the perspective of the software manufacturer, the interest *"the software shall be modularized into separate units."* is important when changing credit card information or the shipping address of the customer. These two tasks are part of the billing process and thus the associated source code need to be better modularized than for other tasks.

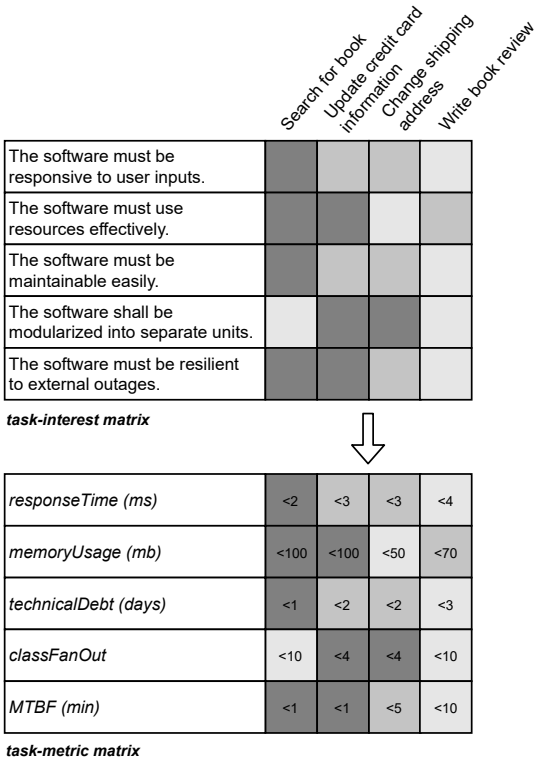


Figure 3: Deriving quantitative measures for user tasks after qualitatively evaluating relevance of stakeholder interests.

Analyzing these interests outside the context of individual user tasks neglects the fact that some tasks are more important than others and thus need more attention. The fulfillment of interests typically requires intertwining multiple teams, e.g., requirements engineering and DevOps teams. This even more emphasizes the need to evaluate its relevance conjointly with all relevant stakeholders in the narrow context of a user task. Also, stakeholder interests evaluated with little relevance for multiple user tasks should be revised as this indicates ambiguities regarding its scope or its actual irrelevance.

D. Specifying Constraints

⑧ **Specification:** In the last step of our method, the *task-interest matrix* created in the last step serves as input for eventually specifying concrete constraints. These constraints are specified individually for each user task based on threshold values to *satisfice* the stakeholder interests. Hence, when specifying concrete constraints the objective is to fulfill the stakeholder interests *satisfactorily and not optimally*, so that a majority of stakeholder interests can be fulfilled.

The previous qualitative evaluation of individual relevance of an interest for a user task helps to find concrete measures and criteria for specifying these constraints. To foster a shared understanding among the stakeholders of the threshold values used in the constraints, it is important to use a well known unit of measurement. Ideally, one should start with determining metrics and threshold values for the most relevant user task for a specific stakeholder interest. Taking the example illustrated in Figure 3, the task to search for a book needs to be specially responsive. Determining the response time as metric and values below 2 milliseconds as being very responsive, the other constraints can be derived thereof straightforwardly. This procedure is repeated until

- for each cell in the matrix there is a concrete constraint defined, or alternatively
- the stakeholders agree that the interest is of little relevance for a certain user task and does not need to be specified.

Figure 3 shows the transition from the *task-interest matrix* (top) into a *task-metric matrix* (bottom) defining metrics and threshold values for each user task and stakeholder interest. Lastly, using the information of the *task-metric matrix* we can specify concrete constraints for each user task. To faciliate expressing these constraints our method provides a compact constraint language. The units of measure used in the *task-metric matrix* do not necessarily need to be the same as used for the constraints, but depend on the operationalization of a measure. As an example, while the technical debt is expressed in days in the *task-metric matrix*, constraints use hours as unit of measure. The same applied to the *mean time between failures (MTBF)* which is defined in minutes in the *task-metric matrix* and expressed in seconds in the constraints.

```
search_book: responseTime < 2, memoryUsage < 100,
             techDebt < 24, fanOut < 10, mtbf < 60.
update_card: responseTime < 3, memoryUsage < 100,
             techDebt < 48, fanOut < 4, mtbf < 60.
change_address: responseTime < 3, memoryUsage < 50,
             techDebt < 48, fanOut < 4, mtbf < 300.
write_review: responseTime < 4, memoryUsage < 70,
             techDebt < 72, fanOut < 10, mtbf < 600.
```

Figure 4: Example constraints to evaluate stakeholder interests using quantitative measures for each user task.

In Figure 4 we illustrate how the information of the *task-metrics matrix* can be expressed as concrete constraints for the elicited user tasks. These constraints allow to evaluate

the fulfillment of the respective stakeholder interests through concrete measures, which themselves are operationalized through dedicated software instruments. For instance, the interest that *"the software must be responsive to user inputs"* is only fulfilled if the response time of the user task's software implementation is below the defined thresholds. For the sake of brevity we skip the details of this constraint language and refer to our other publications in this context [5].

V. Evaluating Fulfillment of Interests

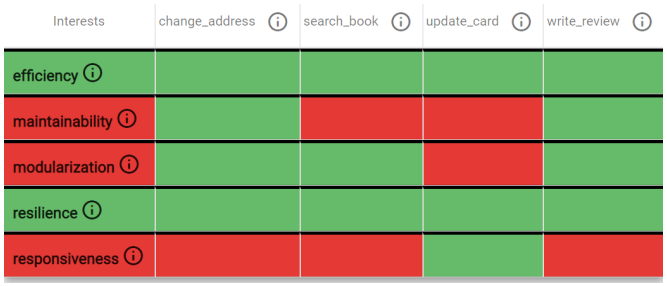
When developing software components for enterprise systems, the TAICOS constraint language can be used to evaluate whether the explicated requirements are fulfilled. Specially for software components that concertedly realize business processes, unfulfilled software quality requirements can lead to delays and disruptions of the affected business processes. For the evaluation of the interests in a development context, the measures used in the respective constraint specification are acquired from development, operational or other enterprise systems. Afterwards, the actual value of a measure can be compared against the threshold value explicated in the constraint. Our constraint language provides basic comparison operators as well as time series operations and time filters for acquiring the measures.

The evaluation results are then displayed on a web application (cf. Figure 5) in a tabular form similar to the *task-interest matrix*. In the *overview* (cf. Figure 5a) the colors of the cells (red, orange, green) give a condensed view about violated interests for a specific user task. If at least one stakeholder interest is violated for a single user tasks, the respective interest is colored in red. As an example, the cell of the maintainability interest in Figure 5a is highlighted in red, even if it is only violated for the user task to update the credit card information.

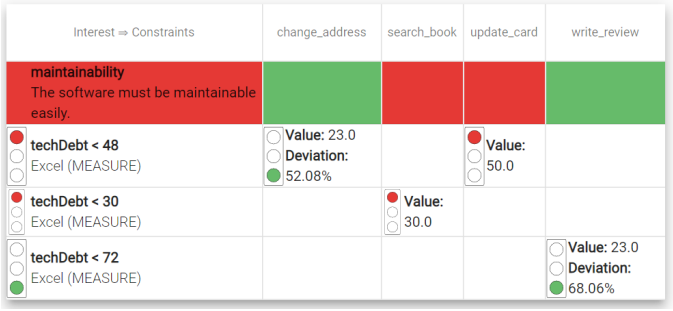
Also, we provide a detailed view of evaluation results for each constraint. Figure 5b shows an exemplary *detailed view* for the maintainability interest and its constraints. If the constraint is fulfilled, its concrete value and deviation from the constraint threshold is also shown. This shall give an early hint that a constraint has only been scarcely fulfilled and that counteractions should be taken. Our main objective was on keeping these visualizations comprehendible specially for non-engineering stakeholders. The combination of visual cues and quantitative information about constraint fulfillment shall ease communication between technical and non-technical stakeholders.

VI. Exploratory Case Study

We conducted an exploratory face-to-face interview study with 11 domain experts to examine how they rate the TAICOS method. Particulary we studied the benefits and weaknesses the experts anticipated from eliciting functional requirements from the most relevant user tasks of an enterprise system and specifying concrete non-functional requirements for it. These domain experts had typical roles in the context of enterprise systems - from requirements engineering to development, operation, and product management.



(a) Overview of user tasks and evaluated stakeholder interests.



(b) Detailed evaluation results of the maintainability interest for each user task, along with the specified constraints.

Figure 5: Visualization of evaluation results of stakeholder interests for user tasks, (a) overview, (b) detailed view.

Taking the guidelines by Runeson and Höst [24] as a blueprint we designed a questionnaire covering practices, challenges, and problems that arise during specifying NFRs with different stakeholders. Also, we conducted a pilot interview as suggested by Yin [25] with one highly experienced expert and included his feedback to improve the questionnaire itself. Particulary, we refined certain phrases in the questionnaire and used more common terminology to assure a good understanding during the interviews among the experts. The questionnaire comprised 20 open questions and 4 closed questions on a 4-point Likert scale and was separated into 3 parts: The first part captured educational and company background, roles in projects and years of experience with requirements engineering. In the second part we asked questions to find out how the companies currently model functional requirements and NFRs, what challenges they are confronted with, and what types of stakeholder interests typically need to be taken into account thereby. Finally, in the third part we presented the experts our method for tailoring stakeholder interests to concrete constraints in the context of user tasks. During the interviews we used Figure 2 to present the method to the experts. In this final part we asked them to specially reflect about the anticipated benefits and drawbacks of the method.

Before asking the experts to rate our method, we presented them a selection of tasks from a well-known online book store and a list of generally understandable performance, privacy, and legal interests. Then, we illustrated how these interests can be used in our method to derive constraints on the level of user tasks. Finally, we asked the experts 4 questions to evaluate our method for tailoring these interests in the context of individual

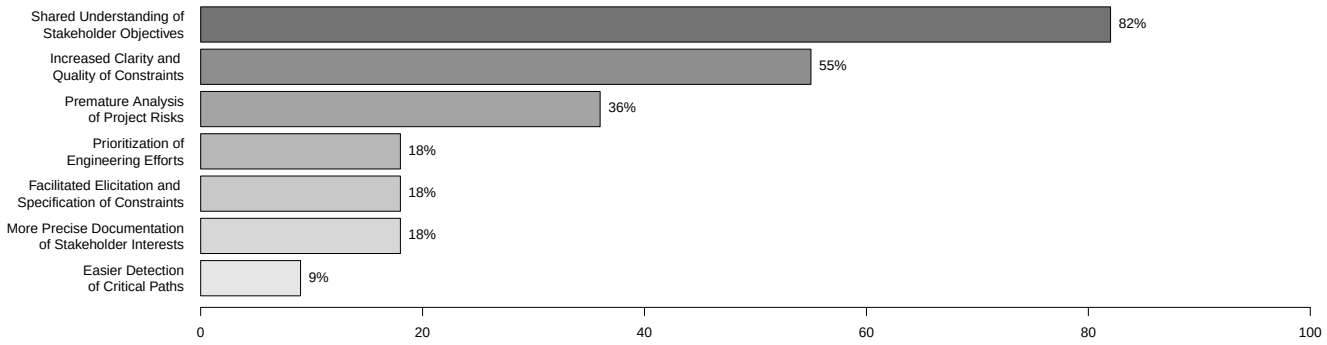


Figure 6: Anticipated benefits of tailoring stakeholder interests to user tasks.

user tasks.

A. Anticipated Benefits and Weaknesses of the Task-Oriented Modeling Method for Enterprise Systems

In one part of the interview, the experts were asked to elaborate the anticipated benefits and weaknesses of the method. They argued that especially due to the granularity and focus on user tasks, they expect the following main benefits of the method:

- fostering shared understanding of functional requirements among stakeholders especially across domains;
- easier and more precise structuring of the functional scope compared to e.g., user stories or epics;
- additional guidance for eliciting functional requirements through the structured decomposition of goals into tasks and subtasks.

Concerning the weaknesses of the method for task-oriented modeling of functional requirements, 27% of the experts anticipated no weaknesses for the practical introduction of the method. We codified the open answers of the 73% experts naming weaknesses, summarizing their statements as follows. 34% responded that, as with every structured method, our method requires knowledge of how to use it effectively in projects. The experts also assumed that the method would require only little training time for its practical introduction, due to its simplicity. The additional maintenance effort to adapt diagrams to changing requirements was expressed as a weakness by 22% of the experts, and the complexity of the model elements was also mentioned by 22% of the experts. Flexibility and additionally introduced complexity were each mentioned by 11% of the experts.

B. Anticipated Benefits and Weaknesses of the Interest-Tailoring Method for Enterprise Systems

Finally, we asked the experts 2 open questions to elaborate the anticipated benefits of our interest tailoring method on the level of user tasks. We condensed their answers to these questions into 8 categories, which are illustrated in Figure 6. Increased comprehensibility of the system was expressed as a benefit by 82% of the experts, namely by increasing the clarity of objectives pursued through an interest. In 55% of the interviews, the experts mentioned the increased specification quality of constraints derived from interests, and 36% said it would help them to assess project risks by better understanding interests and their interdependencies.

In 18% of interviews, experts expressed the prioritization of interests, the time savings accrued by deriving constraints from interests, and the ease of documentation as expected benefits of the method. Only 9% of experts mentioned that our method could also help them to detect critical paths. Based on the open answers examining the weaknesses, we codified the experts' answers into 3 groups. 45% of experts believed that the method would introduce an additional specification effort but also expressed that the expected benefits outweighed these tradeoffs accompanying any structured method. 27% of experts mentioned the complexity of the method as a drawback, and a further 18% anticipated that our method would result in explicitly specifying standard industry constraints that usually need no special documentation (e.g., a default availability, common security requirements).

VII. Threats To Validity

In this section we outline the possible threats to the validity of our method. Particulary this affects the design, execution, and interpretation of the exploratory interview study.

We see a threat to **construct validity** in the different interpretations of the questions by the experts, which is mainly due to their different roles and experiences. We addressed this threat by showing each expert concrete definitions of the terminology used in the interview and discussed any ambiguities. When summarizing the experts' answers, we also considered the background and role of each expert to determine from what view and with what intention the statement was given. Also, as the objective of the exploratory case study was on gathering preliminary feedback for methodological improvement of the method, a further empirical validation is necessary after the experts have applied our method.

The foremost threat to **internal validity** can be seen in some experts' trend to answer in confirmation of our theories. This could have led to confirmation bias, but we regard this as negligible because in response to this trend to answer towards confirming our theories, we asked follow-up questions to capture the experts' actual experiences.

We addressed the threat to **external validity** by selecting experts who operate in different industry sectors, and we also selected only one expert per company. However, we see a threat to the generalizability of the results to other industries due to the different size and maturity of requirements engineering practices in the companies.

Particularly, an empirical validation of the overall approach needs to be conducted to gather empirical evidence about its applicability and suitability in industrial settings.

VIII. Conclusion and Future Work

Utilizing the proposed hierarchical task analysis method to structurally decompose tasks into subtasks seems to offer a promising approach for clarifying the core functionality needed to support the users' goals in enterprise systems. Stakeholder interests can be conjointly refined into quantitative constraints within the context of a user task so that their fulfillment can be automatically evaluated during the software lifecycle. Due to the specification of stakeholder interests and derived constraints on the level of user tasks, our method explicitly addresses challenges arising from the interdependencies between functional requirements and NFRs in software systems.

We have focused our exploratory case study on the elicitation practice and the granularity of NFR specification in the companies. The main objective of this research design was to effectively address the issues and challenges gathered from the expert interviews in our method. Together with an industry partner we are currently planning an empirical validation of the method to ensure the generalizability and the applicability of the results in an industrial context. A special emphasis of this empirical validation will be put on the completeness and expressiveness of the constraint language.

Future work shall specially concentrate on ensuring scalability of the method for large-scale software engineering projects and particularly for efficiently handling multiple heterogenous interests being attached to the same task. Also, the existing tool support for operationalization of measures shall be extended to also integrate data accruing in ERP systems, such as data relating to internal business-process and procurement performance, customer relationship management, and product-related revenue indicators.

References

[1] D. Zowghi and C. Coulin, "Requirements Elicitation: A Survey of Techniques, Approaches, and Tools," in *Engineering and Managing Software Requirements*, A. Aurum and C. Wohlin, Eds. Springer Berlin Heidelberg, 2005, pp. 19–46.

[2] J. D. Blaine and J. Cleland-Huang, "Software Quality Requirements: How to Balance Competing Priorities," *IEEE Software*, vol. 25, no. 2, pp. 22–24, Mar. 2008.

[3] P. Berander and A. Andrews, "Requirements Prioritization," in *Engineering and Managing Software Requirements*, A. Aurum and C. Wohlin, Eds. Berlin, Heidelberg: Springer, 2005, pp. 69–94.

[4] J. Annett, "Hierarchical Task Analysis," in *The Handbook of Task Analysis for Human-Computer Interaction*. London, UK: Taylor & Francis, 2003, pp. 67–82.

[5] P. Haindl, R. Plösch, and C. Korner, "An Extension of the QUAMOCO Quality Model to Specify and Evaluate Feature-Dependent Non-Functional Requirements," in *2019 45th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*. Kallithea-Chalkidiki, Greece: IEEE, Aug. 2019, pp. 19–28.

[6] A. v. Lamsweerde, "Goal-oriented requirements engineering: a guided tour," in *Proceedings Fifth IEEE International Symposium on Requirements Engineering*, 2001, pp. 249–262.

[7] C. Rolland and C. Salinesi, "Modeling Goals and Reasoning with Them," in *Engineering and Managing Software Requirements*, A. Aurum and C. Wohlin, Eds. Springer Berlin Heidelberg, 2005, pp. 189–217.

[8] S. Lauesen and M. A. Kuhail, "Task descriptions versus use cases," *Requirements Engineering*, vol. 17, no. 1, pp. 3–18, Mar. 2012.

[9] F. Fotrousi, S. A. Fricker, and M. Fiedler, "Quality requirements elicitation based on inquiry of quality-impact relationships," in *2014 IEEE 22nd International Requirements Engineering Conference (RE)*, Aug. 2014, pp. 303–312.

[10] N. Riegel and J. Doerr, "A Systematic Literature Review of Requirements Prioritization Criteria," in *Requirements Engineering: Foundation for Software Quality*. Springer, Cham, Mar. 2015, pp. 300–317.

[11] L. Karlsson, s. G. Dahlstedt, B. Regnell, J. Natt och Dag, and A. Persson, "Requirements engineering challenges in market-driven software development – An interview study with practitioners," *Qualitative Software Engineering Research*, vol. 49, no. 6, pp. 588–604, Jun. 2007.

[12] J. Zubcoff, I. Garrigós, S. Casteleynb, J.-N. Mazón, and Aguilar, "Evaluating different i*-based approaches for selecting functional requirements while balancing and optimizing non-functional requirements: A controlled experiment," *Information and Software Technology*, vol. 106, pp. 68–84, Feb. 2019.

[13] D. Ameller, X. Franch, C. Gómez, S. Martínez-Fernández, J. Araujo, S. Biffl, J. Cabot, V. Cortellessa, D. Méndez, A. Moreira, H. Muccini, A. Vallecillo, M. Wimmer, V. Amaral, W. Bühm, H. Bruneliere, L. Burgueño, M. Goulão, S. Teufl, and L. Berardinelli, "Dealing with Non-Functional Requirements in Model-Driven Development: A Survey," *IEEE Transactions on Software Engineering*, pp. 1–1, 2019.

[14] R. Berntsson 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, M. Glinz and P. Heymans, Eds. Springer Berlin Heidelberg, 2009, pp. 218–232.

[15] L. Chung and J. C. P. Leite, "On Non-Functional Requirements in Software Engineering." Springer, 2009, pp. 363–379.

[16] L. Chung, B. Nixon, E. Yu, and J. Mylopoulos, *Non-Functional Requirements in Software Engineering*, ser. International Series in Software Engineering. Springer US, 2000, vol. 5.

[17] M. Daneva, L. Buglione, and A. Herrmann, "Software Architects' Experiences of Quality Requirements: What We Know and What We Do Not Know?" in *Requirements Engineering: Foundation for Software Quality*, ser. Lecture Notes in Computer Science, J. Doerr and A. L. Opdahl, Eds. Springer Berlin Heidelberg, 2013, pp. 1–17.

[18] E. C. S. Cardoso, J. P. A. Almeida, G. Guizzardi, and R. S. S. Guizzardi, "Eliciting goals for business process models with non-functional requirements catalogues," in *Enterprise, Business-Process and Information Systems Modeling*, T. Halpin, J. Krogstie, S. Nurcan, E. Proper, R. Schmidt, P. Soffer, and R. Ukor, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 33–45.

[19] P. Bourque and R. E. Fairley, *Guide to the Software Engineering Body of Knowledge*, 3rd ed. Los Alamitos, CA, USA: IEEE Computer Society Press, 2014.

[20] M. Glinz, "Rethinking the notion of non-functional requirements," in *Proceedings of the Third World Congress for Software Quality (3WCSQ 2005)*, vol. 2, Munich, Germany, 2005, pp. 55–64.

[21] M. Broy, "Rethinking Nonfunctional Software Requirements," *Computer*, vol. 48, no. 5, pp. 96–99, May 2015.

[22] "ISO/IEC/IEEE International Standard - Systems and software engineering – Life cycle processes –Requirements engineering," *ISO/IEC/IEEE 29148:2011(E)*, pp. 1–94, Dec. 2011.

[23] "ISO/IEC/IEEE 25030:2007 International Standard - Software product Quality Requirements and Evaluation (SQuaRE) - Quality requirements," *25030:2007*, 2018, (accessed 2019/12/04). [Online]. Available: https://www.iso.org/standard/35755.html

[24] P. Runeson and M. Höst, "Guidelines for conducting and reporting case study research in software engineering," *Empirical Software Engineering*, vol. 14, no. 2, pp. 131–164, Apr. 2009.

[25] R. K. Yin, *Case Study Research and Applications: Design and Methods*, 6th ed. Los Angeles: SAGE Publications, Inc, Nov. 2017.