Requirements People Debt: Conceptualizing Human Factors in Requirements Engineering Debt

Larissa Barbosa Leôncio Pinheiro[1][0000−0003−1900−029X], Sávio Freire[2][0000−0002−3989−9442], and Rita Suzana Pitangueira Maciel[1][0000−0003−3159−6065]

[1] Federal University of Bahia
[2] Federal Institute of Ceará and State University of Ceará
larissa.leoncio@ufba.br, savio.freire@ifce.edu.br, rita.suzana@ufba.br

Abstract. Context. Requirements debt (RD) refers to decisions regarding which requirements the development team should implement or how they should be implemented. While prior work has investigated RD from multiple perspectives, limited attention has been given to explicitly conceptualizing the role of human factors in its emergence. Goal. This work introduces requirements people debt (RPD), a human-centered perspective on RD that explains how human-related conditions lead to observable deficiencies in requirements artifacts and activities. Method. We cross-analyzed findings from two complementary empirical studies on desirable and undesirable attributes of requirements engineers. Results. We identified 12 causes of RPD, with difficulties in interpersonal relationships and lack of communication being the most frequently mentioned. We also found 55 preventive practices, the most notable being participation in training activities, maintaining frequent communication with stakeholders, and improving business understanding. Conclusion. We organized the findings on RPD into a conceptual map, which may support initiatives aimed at addressing RPD items in software projects and motivate future research in this area.

Keywords: requirements engineering · requirements debt · conceptual map.

1 Introduction

Technical debt (TD) represents the technical compromises that provide short-term benefits, such as increased productivity and reduced costs, but may negatively affect the long-term sustainability of software projects [8, 15]. Initially, TD was primarily associated with issues found in source code [4]. However, TD can also manifest in several other software artifacts, including architecture, documentation, people, and requirements [10, 14].

More specifically, **requirements debt (RD)** refers to decisions made regarding which requirements the development team should implement or how

Proceedings of the 29th Workshop on Requirements Engineering (WER2026), August 11-13, 2026, La Plata, Argentina.
https://doi.org/10.29327/1884517.29-14

they should be implemented [10, 14]. Examples include requirements that are only partially implemented, addressed in limited scenarios, or implemented in ways that fail to fully satisfy non-functional requirements. Previous studies have examined RD from multiple perspectives: defining its concept [9], investigating how to identify [11], manage [5], and measure it [11, 13]. Moreover, different types of RD have been identified. For example, Mendes et al. [12] and Barbosa et al. [1] investigated the impact of requirements documentation debt, which arises from mismatches between stakeholder needs and software implementation as reflected in requirements specifications. Freire et al. [6, 7] explored the causes, effects, and managerial practices of requirements process debt, which refers to inefficiencies or shortcomings in the requirements engineering (RE) process. Although these studies reveal the multifaceted nature of RD, they did not consider human aspects affecting RE activities.

In our previous studies, we examined various human and process aspects of RE. First, we surveyed software practitioners to understand the causes, effects, and mitigation strategies of requirements documentation debt [1]. Subsequently, we investigated the desirable [3] and undesirable [2] attributes of requirements engineers. By cross-analyzing findings from both studies, we identified a new perspective on RD, which we call **requirements people debt (RPD)**. It represents the intersection between requirements-related debt (including documentation and process debt) and the human factors influencing RE activities.

This study introduces the concept of RPD, outlining its items, causes, and preventive practices. To this end, we revisited our prior research to collect and synthesize empirical evidence related to RPD. In total, we identified 12 causes of RPD, with *difficulty in relationships* and *lack of communication* among the most frequently mentioned. Regarding prevention, and 55 preventive practices, most notably *participating in training*, *having frequent communications with stakeholders*, and *improving business understanding*. We organized all RPD elements into a conceptual map.

This new concept provides software practitioners and researchers with a more comprehensive understanding of how requirements-related debt accumulates and affects the software development cycle. It integrates previously discussed types of RD while emphasizing the human dimension. In particular, RPD highlights how inadequate interpersonal skills, poor communication, and ineffective leadership contribute to TD accumulation in software projects. These issues directly affect RE activities leading to specifications that are incomplete, ambiguous, or misaligned with stakeholder expectations. Ultimately, RPD compromises both the quality of requirements and the resulting software product. Moreover, practitioners can use the conceptual map to guide or enhance initiatives for identifying and mitigating RPD.

Besides this introduction, this paper is organized into five additional sections. Section 2 presents the related and our previous work. Section 3 introduces the RPD concept. Section 4 reports how we assess the strategies identified in our previous studies. Section 5 outlines the threats to validity. Finally, Section 6 concludes the paper and offers future work.

Related Work

Mendes et al. [12] investigated the impact of documentation debt on projects developed using agile requirements. Through a retrospective study of 132 maintenance tasks, they found that 65 were related to documentation debt, resulting in approximately 47% additional maintenance effort and 48% extra cost relative to the initial project estimates.

Perera et al. [13] conducted a systematic mapping study to develop the RD quantification model, demonstrating that although RD shares similarities with code-related TD, it also possesses unique components. Specifically, the "interest" on RD, i.e., the additional cost from suboptimal requirement decisions, can arise during both RE and implementation phases.

Lenarduzzi et al. [9] proposed a holistic definition of RD, identifying three types: incomplete user needs, requirement smells, and requirements violations. Their work provides a new perspective on RD. In other related work, Freire et al. [6] analyzed public discourse about RD causes by collecting and examining web discussions through content analysis, identifying 128 causes, 60 effects, 93 preventive practices, and 70 mitigation practices.

Frattini et al. [5] conceptualized how the debt metaphor applies to RE, conducting in-depth expert interviews to propose a theory of RD that connects TD concepts to RE specific phenomena. Finally, Melo et al. [11] performed a systematic literature review of 66 primary studies, identifying causes, identification strategies, and measurement metrics for requirements-related technical debt.

By analyzing the related work, we perceived that RD has different perspectives, each one related to the activities performed in RE. However, while prior studies have acknowledged socio-technical aspects of RE, limited attention has been given to explicitly conceptualizing how human factors contribute to RD.

2.1 Our Previous Studies

In our previous study [1], we investigated the current state of requirements and requirements documentation debt, identifying its causes, effects, preventive and mitigation practices, as well as the practice avoidance reasons related to the non-adoption of these practices. We quantitatively and qualitatively analyzed survey responses from software development professionals concerning requirements and requirements documentation debt and its elements (causes, effects, prevention, and correction). As a result, we identified 55 causes, 33 effects, 26 prevention practices, 3 PARs related to non-prevention, 18 correction practices, and 16 PARs associated with non-mitigation. Finally, these elements were synthesized into a conceptual map.

In subsequent studies [2,3], we explored the **desirable and undesirable attributes of requirements engineers**, their interrelationships, and strategies for developing them. Our method combined a survey with 18 Brazilian software industry professionals from the same country and follow-up interviews with 11 of them. We invited these professionals from our industry partners. The participants had between one and thirty years of experience (average = 10 years)

and had worked for an average of six companies (range: one to twelve). They represented organizations of different sizes: small (up to 50 employees; n = 2), medium (51–1,000 employees; n = 6), and large (more than 1,000 employees; n = 10). The sample included ten software engineers and eight project managers.

To analyze the interviews, we transcribed them and two researchers applied open coding to half of the transcripts to capture the main ideas in each response, resolving disagreements with a third researcher. In the end, we produced a consolidated set of codes, representing the key concepts of each attribute.

Regarding the desirable attributes [3], we found 22, where *investigative ability in stakeholder communication*, *judiciousness*, and *business understanding* were the most cited. These attributes were organized into four categories, and 38 strategies were identified to enhance RE skills—such as *participating in training*, *communicating with all stakeholders*, and *acquiring domain knowledge*. We represented the relationships between attributes and strategies in a conceptual map and visualized their interconnections through a Sankey diagram.

Finally, we found seventeen undesirable attributes [2], with *relationship and communication difficulties* being the most frequently cited. These were grouped into four categories (*communication*, *domain knowledge*, *personality*, and *technical skills*) and organized into a conceptual map to support visualization and analysis.

Although we have examined RE from multiple perspectives, we have not yet explored its relationship with human factors. The following sections present this connection and explain how we conceptualized and defined **Requirements People Debt (RPD)**.

3 Building the Requirements People Debt Concept

When we analyzed the causes, effects, and preventive and mitigation practices of requirements and requirements documentation debt, we realized that human factors were present in each element [1]. For example, the cause *lack of commitment*, the effect *stress with stakeholders*, the mitigation practice *communicating TD items to customers*, and the preventive practice *discipline* all reflect human-related aspects. Therefore, we identified that the human factor was the main factor responsible for incurring this type of TD in the project. These results motivated us to further understand how human factors influence software requirements engineers when performing their activities.

Subsequently, we conducted a multi-method study to investigate the desirable [3] and undesirable [2] attributes of software requirements engineers. As presented in Subsection 2.1, we first asked practitioners to list these attributes and then interviewed them to gather deeper insights. The analysis resulted in a conceptual map that synthesizes the findings and facilitates their interpretation.

3.1 The Concept

By cross-analyzing the findings from both studies [2,3], the concept of **requirements people debt (RPD)** emerged as

Requirements People debt specifically manifests when human-related causes lead to observable deficiencies in requirements artifacts or activities. It is characterized by a causal chain in which antecedent human conditions (e.g., lack of knowledge, poor communication, or cognitive biases) lead to the creation of deficient requirements artifacts or processes (debt items), which in turn incur negative consequences (interest), such as rework, delays, and reduced software quality.

The concept comprises three components: (i) *causes*, defined as human-related conditions (e.g., skills, behaviors, interactions, and cognitive or emotional states) that negatively influence RE activities; (ii) *debt items*, referring to observable deficiencies in requirements artifacts or activities resulting from these causes; and (iii) *effects*, corresponding to the additional costs (interest) incurred due to the presence of RPD items, impacting development outcomes and team dynamics. For example, *lack of communication* (cause) may lead to *superficial specifications* (RPD item), which may result in *rework in coding activities* (effect). In this sense, RPD complements existing RD types, such as requirements documentation debt and requirements process debt, by explicitly representing the human-related causes that contribute to their emergence.

From our data, we identified RPD items, their causes, and preventive practices. Undesirable attributes [2] were interpreted as causes when they represent latent human conditions (e.g., lack of knowledge or communication issues), and as RPD items when they manifest as observable deficiencies in requirements artifacts or activities (e.g., superficial specifications). This classification is based on observability: causes describe internal human conditions, whereas RPD items represent externally observable issues in requirements practices.

We also identified that strategies aimed at reducing undesirable attributes (e.g., *developing prototypes* and *seeking to be confident*) and acquiring desirable ones [3] (e.g., *participating in training* and *communicating with stakeholders*) can be understood as preventive practices for RPD items, as they enhance the ability of requirements engineers to effectively perform their tasks.

The next subsections introduce the RPD items, their causes, and their preventive practices.

Requirements People Debt Items. Table 1 presents the RPD items identified in this study, representing observable deficiencies in requirements artifacts and activities that emerge from human-related causes.

Causes of Requirements People Debt. In total, we found 12 causes of RPD. Table 2 shows the five most cited causes. The complete list of causes is shown in Figure 1.

In the context of this work, *Difficulty in relationships* cause means the challenges that affect team dynamics, as described in: "You think things and don't share them, plan things and think you'll solve everything alone. Often what

Table 1. RPD items with their definitions

RPD item	Definition	Quote
Incomplete stakeholder coverage in requirements	It refers to the omission or insufficient inclusion of relevant stakeholders during requirements elicitation and validation activities, resulting in requirements that reflect only a partial view of the system needs.	"...you'll have several stakeholders, and each one has their own vision, so identify all the stakeholders [...] you have to identify these people and listen to them."
Lack of cross-team validation in requirements	It refers to situations in which requirements are defined without sufficient understanding of technical input, leading to decisions that are not feasible for the team or that overlook the team's actual capabilities.	"If you lack basic knowledge, you might do things that aren't feasible for the team, for example, saying you're going to deliver a certain feature when the team doesn't have someone who can do it, or talking to more technical people and not understanding much of what they're saying."
Make superficial specifications	It is associated with a type of documentation that does not provide sufficient or accurate information about the requirements and expectations of the system to be developed.	"Be as concise as possible, be able to write the requirements in a concise and clear way, using rules, tools, screens, prototyping, and minimal interaction."
Not reviewing what was raised	It is characterized by the absence of validation and review activities over elicited or specified requirements.	"The thing is, everything you produced during the requirements specification phase needs to be reviewed and validated in some way. So if you don't review it, you might have missed something, or there might be a gap you didn't see, or some inconsistency."
Unstructured requirements documentation	It refers to the absence of a clear and logical structure in the documentation of requirements, functionalities and other aspects of the system	"He doesn't write in an organized way. He doesn't separate it into folders, in specific places, he doesn't fill out his tool. Loose sentences and larger definitions scattered in other places."

Table 2. The five most cited causes

Cause	#CC	%C
Difficulty in relationships	11	24%
Lack of communication	9	20%
Lack of business knowledge	5	11%
Inexperienced	4	9%
Not being proactive	4	9%

Caption: #CC: Number of mentions of a cause. %C: Percentage of #CC in relation to the total number of mentioned causes (46).

you're doing has consequences and you can't deal with the complexity alone, but somehow, whether you like it or not, you end up taking on more responsibility than you can handle, not in terms of capacity, but in terms of time."

Lack of communication cause refers the absence or deficiency in the exchange of information, ideas and feedback between individuals or teams, resulting in misunderstandings, lack of coordination and inefficiencies, for instance, "It is a recurring problem and is when a person, unintentionally or sometimes intentionally, keeps information to themselves."

Lack of business knowledge cause is related to the lack of adequate understanding of the context, objectives and needs of the business for which the software is being developed, as described in "Sometimes the analyst is very new to the area, to the company, and doesn't know the business, can't explain the business rules, there are very complex business areas, so he doesn't know them and can't explain them to the developer. The developer ends up needing to understand a little of this, which should have been passed on by the requirements analyst, detail the requirements, be able to make clear what needs to be done."

Not being proactive cause refers to lacking the ability to act in advance to resolve or prevent problems, as shown in: "An analyst who lacks the ability to act proactively to resolve or prevent future problems."

Lastly, *Inexperienced* cause is associated with the low level of experience of the practitioner performing the RE activities. For instance, "When you lack experience in the IT field, it's difficult to propose agile solutions that simply solve the client's problem at the moment the system requirements are being gathered."

By constantly comparing the codes, we grouped them into the following categories:

- **Communication:** It encompasses three causes to express how requirements engineers communicate. These causes are: *difficulty in relationships*, *lack of communication*, and *Imprecision*.
- **Domain Knowledge:** It groups the following causes to support the business knowledge: *lack of business knowledge*, *thinking you already know the client's business,* and *wait for the user to have all the answers.*
- **Technical:** It is composed of two causes associated with technical skills that are used in software requirements activities. The causes are: *inexperienced* and *lack of knowledge of requirements engineering practices and artifacts.*
- **Personality:** It has the following causes related to express how requirements engineers think, feel, and behave in their interaction with stakeholders: *inaccurate*, *not being proactive, not being able to adapt to chaos,* and *anxiety.*

Preventive Practices for Requirements People Debt. In total, we identified 55 preventive practices to avoid RPD items. Table 3 shows the five most cited practices. The complete list of practices is shown in Figure 1.

Table 3. The five most cited preventive practices

Preventive Practice	**#CP**	**%P**
Participating in training	17	12%
Having frequent communications with stakeholders	11	8%
Improving business understanding	11	8%
Acquiring a domain knowledge	9	6%
Talking to stakeholders	7	5%

Caption: #CP: Number of mentions of a preventive practice. %P: Percentage of #CP in relation to the total number of mentioned preventive practices (140).

Participating in training means improving the skills of the requirements engineer through courses, reading specialized material, and getting to know new methodologies. For instance, "Theoretical knowledge, which teaches how to apply the techniques, what is relevant, what kind of attitude the analyst has to have, how he has to prepare, the points of attention he has to have."

Having frequent communications with stakeholders refers to maintaining consistent and effective contact with stakeholders, as illustrated in the following

statement: "Frequent conversations with clients, especially with end users, always involving them in these discussions. Of course, there are negotiation or pricing talks where end users are not as involved—those are more for managers—but when it comes to understanding needs, it's essential. The techniques I mentioned earlier, such as personas and journey mapping, can also be used to make this process easier. Constant communication, along with these techniques, helps facilitate the conversation."

Improving business understanding refers to the effort to collect and deepen knowledge about the business processes, as shown in "Personal work requires being aware that effort is needed to understand the business—whether as a whole or at least the part they will be working on."

Acquiring domain knowledge refers to know a specific field for which the system is under developed, as described in "studying that area in which the system is inserted" and "knowing your (system) domain."

Lastly, *Talking to all stakeholders* is related to keep in touch with customers, users, and software teams to understand and explain the requirements, such as "Talking, demonstrating their ideas, talking both with the customer and the software developers, listening, exchanging information."

We also grouped the preventive practices into categories. We reused the ones we categorized the RPD items (technical, communication, domain knowledge, and personality) and defined the following new ones:

- **Management**: it groups eleven preventive practices related to the software project management, such as *Acquiring domain knowledge* and *Searching for solutions that fit the user's reality.*
- **Social**: it encompasses five preventive practices that involve human interaction and communication during requirements activities. For example, *Talking to stakeholders* and *Seeking how to listen.*

3.2 Organizing Requirements People Debt Elements

We organized the RPD elements (causes and preventive practices) into a conceptual map. Figure 1 presents the complete version of this map, displaying each element and its corresponding category.

In the map, the different RPD management elements are represented by solid-line rectangles, while their categories are enclosed in dashed-line rectangles. Each category and its elements are associated with a percentage, computed by summing the number of occurrences of each element and dividing it by the total number of mentions in which that element appeared.

Causes in the *Communication* category are the most frequent, accounting for 46% of all mentions. Within this category, *Difficulty in relationships* cause stands out, representing 24% of total mentions. Regarding prevention, practices from the *Technical* category are the most frequently mentioned (38% of all mentions), with *Participating in training* being the most prominent, cited in 12% of the projects analyzed.

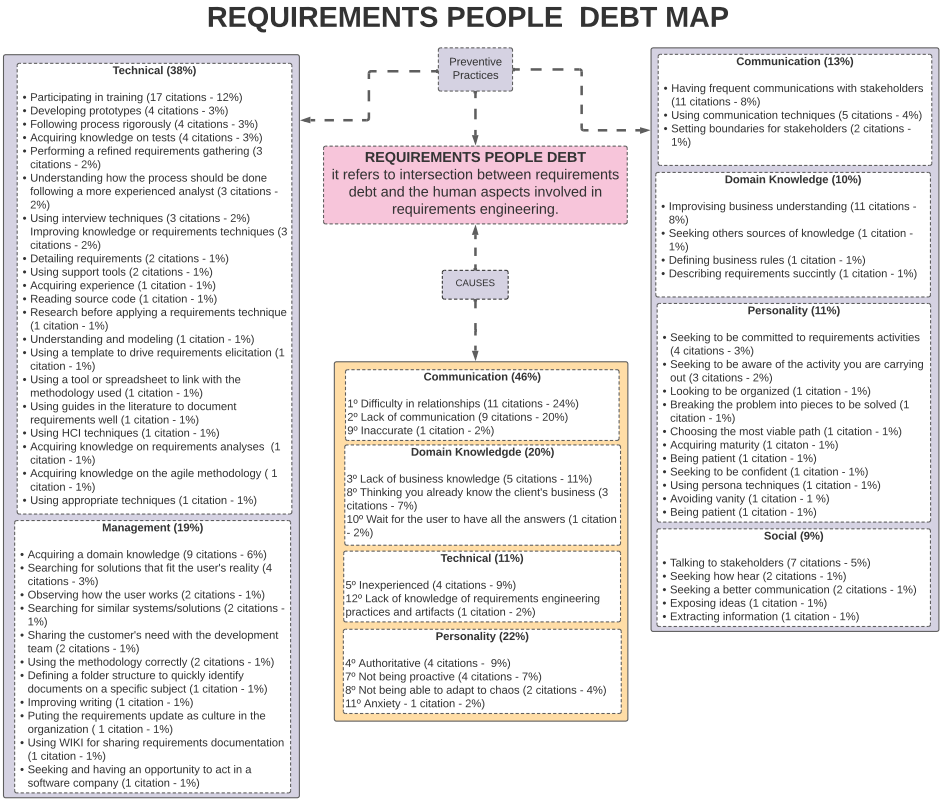

Fig. 1. Requirements People Debt Conceptual Map.

Use of the Conceptual Map. Software practitioners can use the conceptual map to diagnose human-related sources of requirements issues, identify associated RPD items in their projects, and select preventive practices aligned with these causes. In this sense, the map serves as a practical guide to support decision-making and improve RE processes.

As a conceptual guide, the map can also inform actions in response to perceived causes and serve as a comprehensive reference when assessing strategies to avoid them. This is evidence due to the link between causes and preventive practices.

The conceptual map should be interpreted as an analytical framework that links human-related causes to observable deficiencies in requirements artifacts and the practices that may curb them. It is important to note that the map does not indicate a causal correlation between causes and preventive practices. However, it provides valuable associations that can facilitate the development of actionable plans for managing RPD. For instance, if a software team identifies that *lack of communication* is a cause that lead a RPD item existing in its

project, analyzing the categories of preventive practices, the team can realize that applying practices from the *Communication* category can curb this cause.

The map can contribute to creating a more favorable environment for RPD management. By considering the list of of causes and practices as a communication facilitator, helping software practitioners more effectively communicate requirements-people debt-related issues to management and enabling managers to make better-informed decisions regarding the handling of RPD items.

4 Assessing the strategies to deal with undesirable attributes

Our previous study [3] reported an assessment of strategies used to develop desirable attributes. Building on the same research protocol, we assessed strategies to prevent undesirable attributes. These practices are reused as preventive practices for avoiding RPD items. However, the proposed conceptual map has not yet been empirically evaluated.

To assess the perceived importance of each strategy, we asked a follow-up question (see Table 4) to the participants. For each strategy, we presented its definition and asked the participant to indicate the level of importance for a requirements engineer to improve their skills, using a 5-point Likert scale (strongly disagree, disagree, neither agree nor disagree, agree, and strongly agree). We applied descriptive statistics to analyze the responses by counting the proportion of participants who selected each option on the Likert scale for each strategy.

Table 4. Survey questions for assessing strategies to avoid undesirable attributes of requirements engineers.

ID	Follow-up Question (FQ) Description	Type
-	We identified 17 strategies that could be used to avoid less desirable attributes of requirements engineers. For each of them, please indicate how important it is for a requirements engineer to avoid less desirable attributes of requirements engineers.	-
FQ1	Improving business understanding - It is related to the ability to deepen knowledge about the organization's processes, goals and challenges, allowing the analyst to understand the context in which the project is inserted.	Closed
FQ2	Having frequent communications with stakeholders - It is related to the ability to establish communication channels between the requirements analyst and all stakeholders in the project.	Closed
(...)	(...)	(...)
FQ17	Being patient - It is related to maintaining a calm and understanding behavior throughout the project.	Closed

4.1 Demographics

In total, eleven participants answered the follow-up question. Table 5 presents the participants' characterization indicating their years of experience, the number of organizations the participant has worked, current company size (small – up to 50 employees; medium-sized – with 52 to 1000 employees; large – more than 1000 employees), and role. All participants are from different organizations.

It is evident that most of the participants are requirements engineers, with the majority having between eleven and 20 years of experience in their roles. Most participants work in medium-sized companies and most have worked in at least six different companies throughout their careers.

Although our sample consists of software practitioners from the Brazilian software industry, it includes a diverse group of professionals with varying levels of experience in project management or requirements engineering roles across companies of different sizes.

Table 5. Participant's demographics

Participant ID	Years of Experience	#Companies worked	Company Size	Role	Follow up Interviews (Y/N)
P1	15	4	Large	Project manager	Yes
P2	2	3	Small	Requirements engineer	Yes
P3	10	6	Large	Requirements engineer	Yes
P4	25	5	Large	Project manager	Yes
P5	3	4	Small	Requirements engineer	Yes
P6	2	3	Medium-sized	Requirements engineer	Yes
P7	16	7	Large	Requirements engineer	Yes
P8	5	10	Large	Project manager	Yes
P9	5	8	Medium-sized	Requirements engineer	Yes
P10	30	11	Large	Project manager	Yes
P11	10	11	Medium-sized	Project manager	Yes
P12	15	5	Large	Project manager	No
P13	1.5	1	Medium-sized	Requirements engineer	No
P14	15	3	Large	Project manager	No
P15	12	5	Large	Requirements engineer	No
P16	1	1	Medium-sized	Requirements engineer	No
P17	20	10	Large	Project manager	No
P18	1	4	Medium-sized	Requirements engineer	No

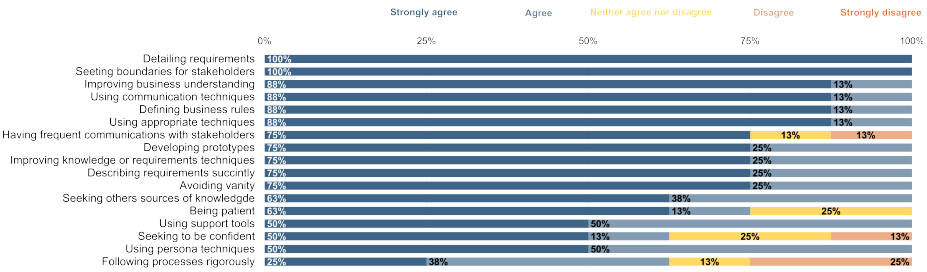

Fig. 2. Interviewees' point of view on the importance of the identified strategies for undesirable attributes (N=11).

4.2 Strategies assessment

Figure 2 presents the results from the follow-up question. In general, the **strategies to avoid undesirable attributes** were well evaluated. The strategies considered most important were: *Detailing requirements, setting boundaries for*

stakeholders, improving business understanding, using communication techniques, defining business rules, using appropriate techniques. On the other hand, the strategy *following processes rigorously* did not achieve unanimous agreement among the participants.

In our previous study [3], we reported the assessment performed for **strategies to obtain desirable attributes**. The participants are the same who assessed the strategies to avoid undesirable attributes. Figure 3 presents the results, showing that *Training*, *having a domain knowledge*, *know how to listen*, *know how to ask*, *know how to write*, *patience*, and *have knowledge on requirements analyses* were considered the main strategies to improve the ability of requirements engineers.

In general, the strategies (to avoid undesirable attributes and obtain desirable attributes) were well evaluated by the participants.

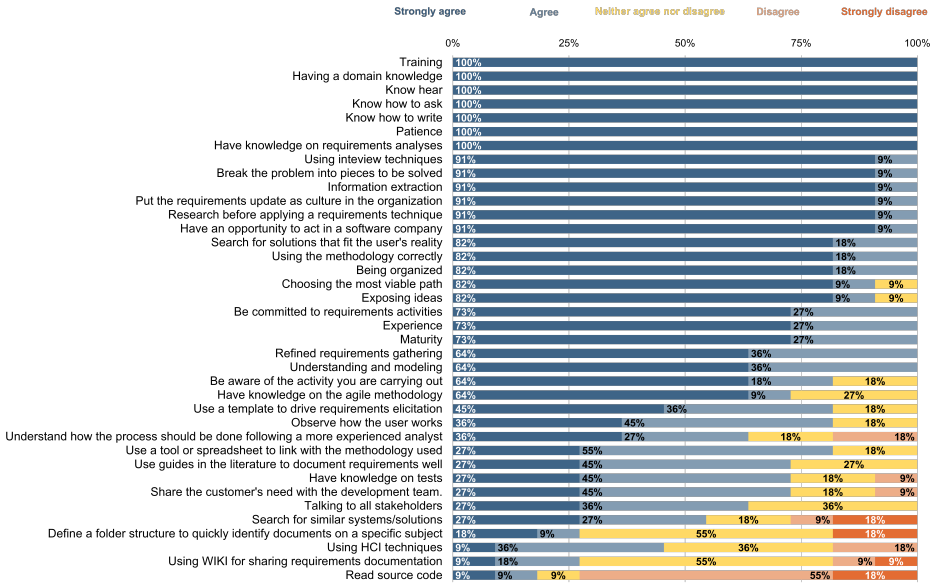

Fig. 3. Interviewees' point of view on the importance of the identified strategies for desirable attributes (N=11) [3].

5 Threats to validity

As with any empirical investigation, certain threats and limitations must be considered to properly assess the validity and reliability of the findings [16]. We discuss these threats and their mitigation actions below.

External validity. Our analysis is based entirely on responses from 18 software practitioners in the software development industry in a single country, as reported in [2, 3]. To mitigate this threat, a variety of participants in terms of years of experience, company size, and number of companies worked was considered, as described in Section 2.1.

Internal validity. Threats to internal validity could arise from the questionnaire itself. Since it was administered remotely, participants might have misinterpreted some questions. To minimize this risk, the questionnaire was subjected to two internal validations and one external validation, as well as a pilot study before data collection.

Conclusion validity. One threat to conclusion validity stems from the qualitative analyses performed, as they inherently involve subjective interpretation. To address this threat, two researchers conducted the analyses independently, and disagreements were resolved through consensus discussions with an experienced third researcher. Additionally, the researchers employed the constant comparison technique when coding the interview transcripts, systematically comparing new findings with previous ones as they emerged during the analysis process. Another threat relates to the participants' project roles. Although requirements engineers are central to the context of this study, not all participants were exclusively performing requirements engineering tasks.

6 Conclusion

In this paper, we introduce RPD as a human-centered perspective on RD, conceptualizing how human-related factors contribute to observable deficiencies in requirements artifacts and activities. We identified a set of RPD items, their causes, and preventive practices, organizing these elements into a conceptual map. The map may support software teams in managing RPD and guide researchers in future problem-driven research efforts. Researchers may also leverage insights from the current state of practice on RPD to develop new methods, strategies, and tools aimed at improving RE and TD management.

As future work, we intend to investigate the state of the practice of RPD, revealing its effects used for repayment and validate the proposed conceptual map in real-world projects and exploring its adaptability across different software development contexts.

Acknowledgments

We used ChatGPT for language refinement. All analysis and interpretation are the authors' sole responsibility. The authors would like to thank all software practitioners who participated in the experimental studies.

References

1. Barbosa, L., Freire, S., Rios, N., Ramač, R., Taušan, N., Pérez, B., Castellanos, C., Correal, D., Pacheco, A., López, G., Mandić, V., Maciel, R.S.P., Mendonça,

M., Falessi, D., Izurieta, C., Seaman, C., Spínola, R.: Organizing the td management landscape for requirements and requirements documentation debt. In: 25th Workshop on Requirements Engineering (WER) (2022)
2. Barbosa, L., Freire, S., Kalinowski, M., Codabux, Z., Spínola, R., Mendonça, M., Maciel, R.S.P.: Understanding undesirable attributes of requirements engineers: Insights from practitioners (2026). https://doi.org/10.13140/RG.2.2.20424.07683
3. Barbosa, L., Freire, S., Maciel, R.S., Mendonça, M., Kalinowski, M., Codabux, Z., Spínola, R.: Attributes of a great requirements engineer. Journal of Systems and Software **219**, 112200 (2025)
4. Cunningham, W.: The wycash portfolio management system. In: Addendum to the Proceedings on Object-Oriented Programming Systems, Languages, and Applications (Addendum). p. 29–30. OOPSLA '92, ACM, New York, NY, USA (1992)
5. Frattini, J., Fucci, D., Mendez, D., Spínola, R., Mandić, V., Taušan, N., Ahmad, M.O., Gonzalez-Huerta, J.: An initial theory to understand and manage requirements engineering debt in practice. Information and Software Technology **159**, 107201 (2023)
6. Freire, S., Maciel, R.S.P., Mendonça, M., Leite, J.C.: Eliciting public discourse of se tool providers in a study on requirements process debt - a different shade of gray. In: Proceedings of the XXIII Brazilian Symposium on Software Quality. p. 189–198. SBQS '24, ACM, New York, NY, USA (2024)
7. Freire, S., Mendonça, M., Do Prado Leite, J.C.S.: What does a public discourse state about requirements process debt causes? In: 2025 IEEE 33rd International Requirements Engineering Conference (RE). pp. 192–204 (2025)
8. Izurieta, C., Vetrò, A., Zazworka, N., Cai, Y., Seaman, C., Shull, F.: Organizing the technical debt landscape. In: 2012 Third International Workshop on Managing Technical Debt (MTD). pp. 23–26 (2012)
9. Lenarduzzi, V., Fucci, D.: Towards a holistic definition of requirements debt. In: 2019 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM). pp. 1–5 (2019)
10. Li, Z., Avgeriou, P., Liang, P.: A systematic mapping study on technical debt and its management. J. Syst. Softw. **101**(C), 193–220 (mar 2015)
11. Melo, A., Fagundes, R., Lenarduzzi, V., Santos, W.B.: Identification and measurement of requirements technical debt in software development: A systematic literature review. Journal of Systems and Software **194**, 111483 (2022)
12. Mendes, T.S., de F. Farias, M.A., Mendonça, M., Soares, H.F., Kalinowski, M., Spínola, R.O.: Impacts of agile requirements documentation debt on software projects: a retrospective study. In: Proceedings of the 31st Annual ACM Symposium on Applied Computing. p. 1290–1295. SAC '16, ACM, New York, NY, USA (2016)
13. Perera, J., Tempero, E., Tu, Y.C., Blincoe, K.: Quantifying requirements technical debt: A systematic mapping study and a conceptual model. In: 2023 IEEE 31st International Requirements Engineering Conference (RE). pp. 123–133 (2023)
14. Rios, N., de Mendonça, M.G., Spinola, R.O.: A tertiary study on technical debt: Types, management strategies, research trends, and base information for practitioners. Information and Software Technology **102**, 117–145 (2018)
15. Spinola, R.O., Zazworka, N., Vetrò, A., Shull, F., Seaman, C.: Understanding automated and human-based technical debt identification approaches-a two-phase study. Journal of the Brazilian Computer Society **25**(1), 5 (Jun 2019)
16. Wohlin, C., Runeson, P., Hst, M., Ohlsson, M.C., Regnell, B., Wessln, A.: Experimentation in Software Engineering. Springer Publishing Company (2012)