Research Question:	RQ1.2 How do software practitioners conceptualize TD?
Goal:	Investigate how software practitioners define TD and if this definition is close to the one commonly disseminated in the technical literature
Expected Result:	- Conceptual map of identified elements in TD definitions
Necessary Files:	- RQ1.2_Analysis_COUNTRY-NAME_Execution_TDPerception.xlsx

Analysis Procedure

For answering this research question, we consider the participants' answers for Q10, which asks the participants how they define TD.

In order to analyze the reported definitions, a researcher needs to carry out a coding process to identify the main TD "elements" described by the participants. By "elements" we mean any word that the participant uses to characterize TD. For example, in the statement below, the elements are highlighted in blue bold:

> "Technical debt refers to **activities** that, during the software development, **are not performed** due to **unfamiliarity** and **time constraints**, that in the future can become **harmful to the good evolution of the system**."

At the end, all identified elements are organized in a conceptual map that represents the point of view of the participants of survey about the concept of TD.

To perform this analysis, we suggest the following steps:

1. Each definition should be read and elements identified should be extracted as described by the participants;
2. The identified elements should be grouped using as criterion the meaning of each element. When elements have the same meaning but different nomenclature, we need to standardize their names considering the most commonly cited term;
3. At this moment, we obtain the final list of elements associated with the TD definition;
4. Considering the list obtained in the step 3, we try to identify how the elements are related to each other. To perform this step, we need to analyze each TD definition reported in search of explicitly described relationships among the elements;
5. Finally, we can build the conceptual map containing the elements and their relationships identified in the TD definitions reported by the participants;
6. All these steps need to be reviewed by a second researcher.

Example

Table 1 exemplifies the steps 1 and 3 using two TD definitions extracted from the set of answers (InsighTD BR):

Table 1. Examples of the TD definition analyses procedure

Step 1 – Extraction of elements	**Step 2 - Standardization**

Technical debt refers to activities that, during the software development, are not performed due to unfamiliarity and time constraints, that in the future can become harmful to the good evolution of the system.	Not performed activity Lack of knowledge Lack of time Low maintainability
TD can be understood as a strategy to stop doing some activity/artifact in order to achieve a short-term goal. On the other hand, TD can also be understood as the activities / artifacts that were not developed by negligence or even lack of knowledge. When not properly managed, TD can impact the maintenance and evolution of the software.	Team Choices Not performed activity Short-term goal Negligence Lack of knowledge Low maintainability

Considering the same examples presented in Table 1, Table 2 illustrates how **Step 4** can be performed. The underlined texts refer to the identified elements. The bold and colored fragments refer to the terms that can allow the identification of the relationship between the elements.

Table 2. Examples of the identification of relationships between the elements

Step 1 – Extraction of elements - TD definition	**Step 3 - Identified relationships**
Technical debt refers to activities that, during the software development, are not performed **due to** unfamiliarity and time constraints, that in the future **can become** harmful to the good evolution of the system.	• Lack of time → Not performed activity → Low maintainability • Lack of knowledge → Not performed activity → Low maintainability
TD can be understood as a strategy **to** stop doing some activity/artifact **in order to** achieve a short-term goal. On the other hand, TD can also be understood as the activities / artifacts that were not developed by negligence **or even** lack of knowledge. When not properly managed, TD **can impact the** maintenance and evolution of the software.	• Short-term goal → Team choices → Not performed activity → Low maintainability • Negligence → Not performed activity → Low maintainability • Lack of knowledge → Not performed activity → Low maintainability

Finally, Figure 1 presents an example of the conceptual map containing the elements and their relationships identified in the TD definitions reported by the study participants in the InsighTD BR execution. Whenever there is a relationship between two elements, we need to put a link between them on the conceptual map. According to the participants point of view, TD can be inserted intentionally or unintentionally. Lack of knowledge leads to unconscious decisions that end up leading to the presence of debt items. On the other side, short-term goals and lack of time impact team choices about how development activities will be carried out. These choices are sometimes characterized by issues such as planning failure, inappropriate prioritization of activities, and negligence in carrying out activities. Finally, TD was mainly characterized by unrealized activities or poorly performed activities. Their presence can bring several consequences for the project as low quality, negative impact on the project, rework, among others. The numbers represent the number of times the elements were quoted in Q10 responses.

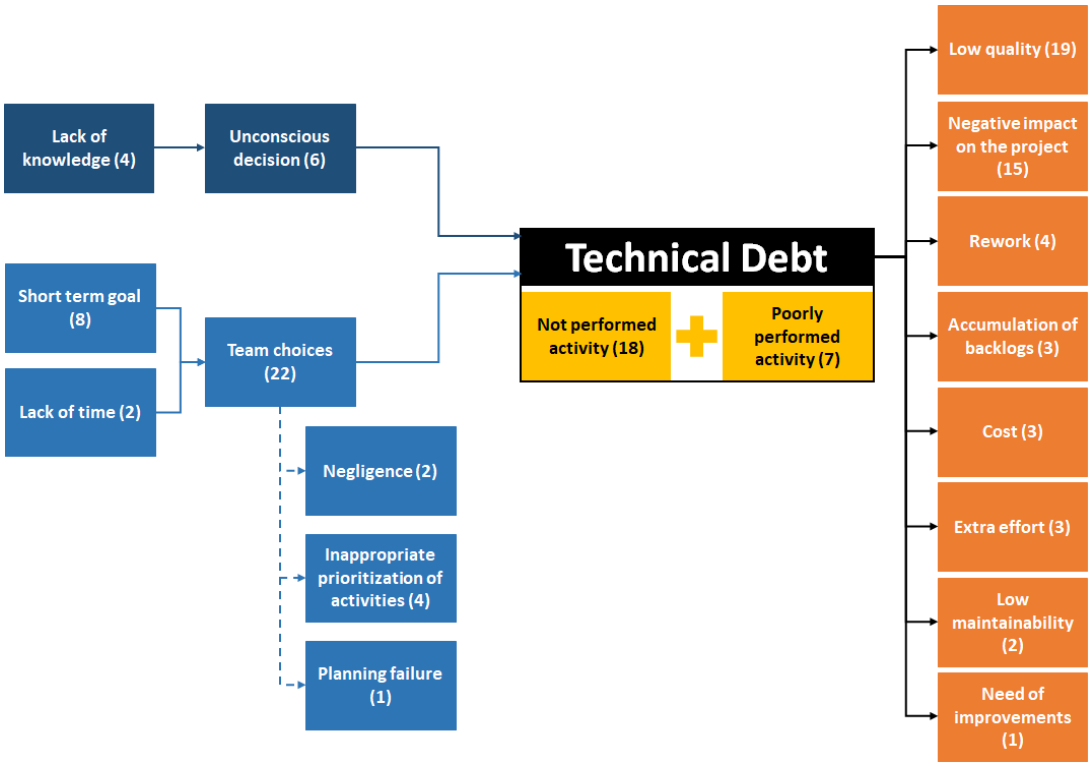

Figure 1. Conceptual map of identified elements in TD definitions reported by study participants