# Ontology Proposal for Support Team: A Case Study in a Software Development Company for the Financial Market

Maria Gabriela Costa Lazaretti[1][a], Vitor Augusto Cecilio e Silva[2], Nelson Nunes Tenório Junior[2,3][b], Thaise Moser Teixeira[3][c] and Steffi Becker[1][d]

[1] *Universidade Cesumar, CESUMAR University, Maringá, Paraná, Brazil*

[2] *Universidade Estadual de Maringá, UEM, Maringá, Paraná, Brazil*

[3] *Universidade Cesumar, CESUMAR University, Instituto de Ciência, Tecnologia e Inovação (ICETI), Maringá, Paraná, Brazil*

*fi*

Keywords: Knowledge Management, Search, Retrieve, Maintenance.

Abstract: Knowledge management has become a crucial activity for organizations focused on knowledge. This is particularly true for software development companies, as their knowledge has become a complex factor directly influencing the practice of developing and maintaining software products. One challenge in software maintenance is organizing knowledge effectively. While tools like bug tracking support the maintenance of software-based products, they primarily automate processes and may not address knowledge organization comprehensively. To enhance tool utilization, one approach is to incorporate ontology, which explicitly represent knowledge for efficient retrieval. This work aims to present a prototype ontology that can be further improved through a proactive and reactive knowledge management initiative in a software development company. A case study was conducted in the customer support sector of the company, utilizing data analysis from a dedicated database and engaging in conversations with a company collaborator. The prototype is presented along with its capabilities to address the problem identified in this study. It is concluded that the developed ontology could be an option for assisting the knowledge organization process in the studied company. However, further research is necessary to assess the return on investment of implementing the suggested solution.

## 1 INTRODUCTION

The world is undergoing transformation changes driven by the vast flow of information, globalization, and the rise of a knowledge-based society focused on knowledge production and dissemination (de Faria, 2003). Consequently, "companies have increasingly embraced knowledge as a means to enhance their competitiveness" (dos Santos et al., 2016). Given the socially intricate nature of knowledge, its imitation is challenging, and thus, organizations that prioritize knowledge can attain sustained competitive advantages by not only creating knowledge but also effectively leveraging it (Trierveiler et al., 2015). The paradigm of the information society has displaced the industrial society paradigm, shifting from a national economy to a global economy and from centralization to decentralization. These characteristics define the knowledge society (SOUZA, 2015). Consequently, an organization's knowledge becomes its competitive advantage, setting it apart from competitors. However, it's important to note that this knowledge does not reside solely within the organization itself, but rather within the individuals who comprise it (Lacombe, 2013).

The same holds true for software development companies. Knowledge is acknowledged as a multifaceted element that shapes the practice of software engineering (Schneider et al., 2009), leading to improved individual and organizational performance and competitive advantage (Mao et al., 2016). According to (Chen et al., 2018), the majority of software development costs arise during the production phase, when the software is ready and actively used by users, requiring ongoing enhancements and up-

[a] https://orcid.org/0000-0002-8314-8531
[b] https://orcid.org/0000-0002-7339-013X
[c] https://orcid.org/0000-0002-7115-9807
[d] https://orcid.org/0000-0002-8464-0122

Lazaretti, M., Silva, V., Tenório Junior, N., Teixeira, T. and Becker, S.
Ontology Proposal for Support Team: A Case Study in a Software Development Company for the Financial Market.
DOI: 10.5220/0012237700003598
In *Proceedings of the 15th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management (IC3K 2023) - Volume 3: KMIS*, pages 73-84
ISBN: 978-989-758-671-2; ISSN: 2184-3228

dates carried out by maintenance teams.

According to (iee, 1998), the software maintenance process encompasses several phases, including problem identification, classification, prioritization, analysis, design, implementation, testing, and delivery. To successfully execute these phases, it is essential to possess knowledge of the product domain, understand the characteristics of the organization utilizing the software, and be familiar with various software engineering techniques and tools. However, (Anquetil et al., 2007) highlight a significant challenge in software maintenance, which is the scarcity of knowledge about the product, as this knowledge often resides solely within the minds of software engineers.

The user support sector plays a pivotal role in software product maintenance, as it serves as the intermediary between user dissatisfaction and the resolution of encountered problems. While tools like Bug Trackers exist to assist in resolving issues, such as duplicate problem reports (Hindle and Onuczko, 2019), they do not guarantee a solution to the challenge of effectively organizing knowledge related to the correction process.

While not all knowledge management (KM) practices rely on information systems, such systems can play a collaborative role in KM through various means. They can facilitate efficient knowledge search within existing databases, provide means for externalizing knowledge, and help determine customer needs through transactional data analysis (Alavi and Leidner, 2001). KM systems act as facilitators by capturing knowledge, ensuring its relevance in terms of processes and content, and making it available to interested parties when needed (Damodaran and Olphert, 2000) .

Hence, the aim of this article is to utilize data mining from a software development company's customer call database to propose an initial version of a domain ontology. This study, conducted as a case study, intends to address the following research question: Can ontologies serve as a means to enhance the knowledge retrieval process of an instant payments online software product?

## 2 KNOWLEDGE MANAGEMENT AND SOFTWARE ENGINEERING

According to (Matsumoto, 2014) , Software Engineering encompasses the discipline of software development, operation, and maintenance. It is composed of 15 areas of knowledge, which include: Software requirements, Software design, Software construction, Software testing, Software maintenance, Software configuration management, Software Engineering Management, Software Engineering Process, Software Engineering Models and Methods, Software Quality, Software Engineering Professional Practice, Software Engineering Economics, Computing Fundamentals, Fundamentals of Mathematics, and Engineering Foundations.

While the development phase of a software project typically spans a few months or years, the software maintenance phase often extends over many years (Serna and Serna, 2014). During this phase, the software undergoes modifications aimed at fixing issues and enhancing performance. To accomplish this, various steps are involved, including problem identification, prioritization, analysis, design, implementation, testing, and delivery (iee, 1998). Hence, the maintenance phase encompasses the Software Engineering phases as defined by Swebok. Knowledge plays a crucial role in this context, as the maintenance team requires an understanding of the software's domain, its user base, Software Engineering practices, programming languages utilized, module interrelationships, and more (Pigoski, 1996).

This knowledge often proves challenging to identify and locate, as it may be documented sparingly or reside solely within the minds of specialists. Consequently, a significant portion of this knowledge remains untapped and unused in day-to-day operations (Walz et al., 1993). Knowledge Management (KM) aims to address this issue by developing structured systems and processes to ensure the retention and sharing of knowledge. According to (Gopalkrishna et al., 2012), KM involves incorporating individual knowledge into business processes, enabling software maintainers to share knowledge (RODRÍGUEZ ET. AL., 2004). This sharing of knowledge offers numerous benefits, including enhancing product quality, improving software maintenance processes, reducing costs, and minimizing errors (Dingsøyr and Conradi, 2002).

Some fundamental concepts for the implementation of a GC system, regarding the basic objects that are manipulated, are the concepts of (I) data, which is any content that can be observed; (II) information, which is content that represents data that has been analyzed, that is, contextualized data; and (III) knowledge, which is the understanding of information (Statdlober, 2016). Still, as for the types of existing knowledge, two stand out: tacit knowledge, which is not formally documented anywhere, is that which belongs to a particular individual, and explicit knowledge, which is that which has been documented and can be accessed either in a physical document or in an electronic system.

## 2.1 Knowledge Centered Support

Companies have recognized the significance and crucial role of customer support quality in gaining a competitive edge (Negash et al., 2003). Despite the existence of electronic systems, (Davenport and Klahr, 1998) argue that accessing the knowledge required to solve customer problems is not straightforward. The extensive time required to search for solutions within documents means that the only readily accessible knowledge is that which resides within the minds of experts. Consequently, organizations often rely on a large number of specialists to address the diverse range of customer issues. (Davenport and Klahr, 1998) also discuss the utilization of knowledge management techniques by organizations to externalize knowledge and make it readily available to support teams.

The subarea of Knowledge-Centered Support, developed by the Consortium for Service Innovation, primarily focuses on reactive knowledge management. Reactive knowledge management involves capturing and updating knowledge at the moment it is required or has been utilized (Statdlober, 2016). This approach offers several advantages, including potentially lower knowledge management costs, integration of knowledge generation into the customer support process, and alignment of stored knowledge with customer needs. However, there are also drawbacks to this approach, such as the potential lack of necessary knowledge during initial interactions and the risk of redundancies within the knowledge base.

The primary focus of the case study conducted in this work was the Solution Loop, which is a set of practices within Knowledge-Centered Support (KCS) for capturing, structuring, reusing, and improving knowledge (Statdlober, 2016). In the first stage, knowledge is "captured" at the point when it becomes explicit. This capture should occur in the user's context to aid in knowledge retrieval. The captured knowledge is then stored to facilitate future queries. In the second stage, the knowledge is "structured," involving descriptions, technical environment details, solutions, and metadata. The third stage entails "reusing" the knowledge, wherein users are encouraged to search the knowledge base before seeking support to determine if a solution to their problem already exists. This practice helps avoid unnecessary repetition by leveraging existing articles. Finally, the last stage involves "improvement" of knowledge through reviewing and updating existing content.

## 2.2 Ontology for Storing and Retrieving Knowledge

The field of knowledge representation is described by (Lakemeyer and Nebel, 2005) as a subfield of artificial intelligence that addresses the challenge of representing, maintaining, and manipulating knowledge related to a specific application domain. (Lakemeyer and Nebel, 2005) also discuss the computational criteria involved, including the expressiveness and efficiency of the representation. They highlight the need to strike a balance between these two factors to achieve the desired objectives with the knowledge representation.

An ontology can be defined as a representational artifact that, when integrated into systems, provides a structured framework for machines (Almeida, 2020). A representational artifact refers to a representation, model, or description of an object, process, or concept created to aid in understanding, explaining, or communicating an idea. Essentially, it is an artificial construct that represents something that exists or can exist. For instance, a map is a representational artifact that visually depicts the geography of a place. Similarly, an ontology can be considered a representational artifact, as it serves as a model that represents concepts and relationships within a specific domain of knowledge (Almeida, 2020).

Furthermore, an ontology can serve as a means to facilitate the sharing of organizational knowledge, enabling interoperability among the systems within a company (Horrocks, 2008). An ontology can be defined as a "conceptual and terminological description of shared knowledge about a specific domain" (Serna and Serna, 2014), enhancing communication between different stakeholders by establishing a common nomenclature and conceptualization system (De Reuver and Haaker, 2009). This shared understanding and standardized representation foster improved communication and collaboration among various actors in an organization.

By employing ontologies, it becomes feasible to define concepts, classes, properties, and relationships between objects. These ontologies help structure and organize knowledge, establishing a shared vocabulary for researchers who require information sharing within a specific domain (Noy et al., 2001). A domain ontology serves as a formal and explicit representation of concepts, entities, and their relationships within a particular knowledge domain. Its primary objective is to enhance comprehension, communication, and interoperability among systems and individuals involved in that domain (Chandrasekaran et al., 1999).

Domain ontologies play a critical role in current research areas such as Machine Learning, Internet of Things, Robotics, and Natural Language Processing. They enable information exchange among disparate systems (McDaniel and Storey, 2019). Among their many applications, domain ontologies are utilized in Knowledge Management (KM) systems to organize and structure knowledge within organizations (Almeida and Barbosa, 2009). Several methods exist for the development of domain ontologies, aiming to provide systematic guidance for their construction and subsequent manipulation. In this article, we will focus on and follow Method 101, developed by (Noy et al., 2001).

#### 2.2.1 Method 101

Method 101 serves as a valuable resource for ontology creation. To avoid redundant mentioning of the authors who developed the method, it should be noted that this section draws inspiration entirely from their guide. While various methods exist in the literature, there is no definitive or superior approach to ontology development. Thus, the selection of a method to define an ontology is guided by three principles: (I) There is no singular correct way to model a domain; multiple viable alternatives always exist. (II) The optimal solution depends on the intended application and anticipated extensions. (III) Ontology development is an iterative process, requiring continuous refinement and iteration.

The guide outlines 7 main steps to aid in ontology modeling and prompt relevant questions during the process. These steps are as follows:

- Step 1: Begin designing the ontology by posing basic questions that help define the domain. Subsequently, develop competency questions that the knowledge base should be capable of answering.
- Step 2: Determine if the domain has already been modeled by checking publicly available databases, or consider using an expandable version of an incomplete domain.
- Step 3: Compile a list of terms that need to be explained to users. Identify which terms represent properties and which refer to classes or individuals.
- Step 4: Define the classes and class hierarchy using one of three approaches: (I) Top-Down, where the most general classes are defined first and specialists are defined later; (II) Bottom-Up, where the most specific classes are defined first, followed by their generalizations; and (III) a mixture of Bottom-Up and Top-Down.
- Step 5: Determine the characteristics that describe both the class as a whole and individual instances. These characteristics can be intrinsic or extrinsic. Define the (non-hierarchical) relationships between different classes.
- Step 6: Define the properties, which can have various types and cardinalities (e.g., string, number, boolean, enumeration).
- Step 7: Select a specific class and develop instances or individuals that represent specific details about that class.

The guide also emphasizes other ”good practices,” but for the purposes of this article, the discussion of the methodology steps is sufficient.

## 2.3 Related Works

The works of different authors related here provide perspectives on how ontologies can be applied to address issues relevant to user support, the software industry, and information retrieval in the context of knowledge management.

Starting the literature review, we have the research by (Oliveira et al., 2022), where the authors conducted an investigation in the context of software development support. The researchers modeled an ontology using Protégé software, based on stages of planning, specification, knowledge acquisition, conceptualization, and ontology validation. The purpose of the developed ontology was to provide an artifact that could be used to translate poorly structured and uninformative reports from the support sector into more meaningful texts.

In the context of ontologies used to collaborate during the software engineering process, specifically in the subfield of requirements engineering, (Nardi and de Almeida Falbo, 2006) present an ontology that is intended to serve as a foundation for the development of tools in this domain. The authors emphasize the importance of a clear understanding of requirements concepts, as well as their relationship with other elements of the software engineering process. Furthermore, they continue to elaborate that with an understanding of these concepts, it is possible to build tools that support the requirements engineering process and are more useful than traditional tools.

(Isotani and Bittencourt, 2015) provide a review of the main challenges encountered when it comes to ontology-based software engineering. They discuss the adoption of ontologies in software engineering to create tools that help prevent communication errors, requirements issues, and information sharing problems. They also explore how ontologies can assist in software management, ensuring quality and integrity throughout the software development phases. The authors identify best practices in areas related to software engineering with the intention of creating a semantic software development environment. An interesting point related to this work is the difficulty in managing information for developers due to the complex landscape of different interdisciplinary and distributed systems, and the fact that information flow between agents, clients, and end-users does not always occur adequately.

To conclude, in the context of ontologies as artifacts for information retrieval, (Rezgui, 2006) proposes and validates an ontology in the construction industry sector. Information retrieval techniques are used to support the needs of users involved in the central system of the case study discussed in the article. The paper also discusses how ontologies can be used to semantically index knowledge present in documents, even though more conventional techniques like keyword-based indexing yield satisfactory results, they simplify important elements such as the hierarchical relationship between terms, aggregation between different terms, and the distinction between specialization of different terms.

## 3 RESEARCH METHOD

The completion of this article involved four distinct stages. The first stage involved conducting a literature review on ontologies, knowledge management, software maintenance, and the selected ontology engineering methodology, namely the ”one hundred and one” method. Additionally, a survey of related works was conducted during the theoretical foundation phase, with an exploratory and non-exhaustive approach. This survey encompassed works that applied ontologies to aid in the software development process, customer support process, and as information retrieval artifacts.

The research tool employed for this study was Google Scholar, chosen for its indexing capabilities across various types of works, thereby facilitating the search process. The key terms used in Portuguese to search for references on the mentioned topics included: “Ontologias”; “Desenvolvimento de Ontologias”; “Ontologias em desenvolvimento de Software”; “Ontologia no suporte ao consumidor”; “Método 101”; “Recuperação da informação”; “Ontologia para recuperação da informação”; “Ontologia para manutenção de Software”. Additionally, the following English keywords were utilized: “Ontology”; “Ontology Development”; “Ontology in software development”; “Ontology for customer support”; “Method 101”; “Information Retrieval”; “Ontology for information retrieval”; “Ontology for Software Maintenance”. In the second stage of this work, a case study was conducted involving the development of a domain ontology for the support sector of a software development company. The case study followed the steps outlined in the Method 101 guide proposed by (Noy et al., 2001). The dataset used for the case study consisted of structured (numeric) and unstructured (text) data, provided by a software company specializing in the financial market. The dataset was collected between January 2021 and April 29, 2022, and comprised 4,591 call occurrences. The data was initially organized in a tabular spreadsheet format, consisting of 56 fields. However, for the purpose of developing the ontology and conducting the case study, only 5 fields were deemed relevant: Subject, Reason/Functionality, Type, Creation time, and Time of resolution.

For data analysis, the Orange software was utilized, originally developed as a library for the Python programming language, offering a wide range of machine learning algorithms (Demšar et al., 2013). Since its inception in 1997, it has garnered an active user base that contributes to the library’s ongoing development and maturity. The visual interface version of Orange includes pipelines for data visualization, which aim to simplify the data exploration process by concealing complex implementation details (Demšar et al., 2013). One drawback of using Orange is the absence of certain features found in other similar software, such as KNIME (Tougui et al., 2020). Nevertheless, the graphical interface of Orange proved to be indispensable for this article as it offers unique components called Widgets that can be interconnected to create a data processing and visualization flow.

In order to visualize and clean the data, a preprocessing step was performed to remove elements with null values in the columns of interest. Additionally, unnecessary columns were eliminated for both data analysis and ontology development, resulting in a reduced total of 4,381 occurrences. To achieve this, two workflows were created using the Orange Software. For analyzing the attributes ”Motive/Functionality,” ”Type,” ”Time of creation,” and ”Time of resolution,” an exploratory data analysis was conducted utilizing a collection of simple but robust techniques (Lopes et al., 2019). The first workflow employed a combination of Widgets including ”data import,” ”Column selection,” ”Column remover by parameter,” ”Unique per category,” ”Bar chart,” and ”Statistics.”

The second workflow was developed for analyzing the ”Subject” attribute, utilizing text mining concepts, which involve the preparation of text data. The commonly adopted approach in text mining is to employ the simplest technique that yields satisfactory results, such as the TF-IDF (Term Frequency-Inverse Document Frequency) algorithm (Provost and Fawcett, 2016). The TF-IDF algorithm is widely recognized for word weighting, utilizing two key components: the term frequency (TF), which measures the frequency of a term within a document, and the inverse document frequency (IDF), which quantifies how many documents contain the term ((Hakim et al., 2014). The TF-IDF value is calculated using the following equation, where t represents the term, d denotes a document, and D represents the collection of documents:

$$TF - IDF(t,d,D) = TF(t,d)xIDF(t,D) \quad (1)$$

For this second workflow, the following Widgets were utilized: ”Data import,” ”Column selection,” ”Removal of lines by parameter,” ”Corpus,” ”Text pre-processing,” ”Concordance,” and ”Extraction of keywords.”

Furthermore, it is worth mentioning that alongside the automated data analysis, a manual examination of the database was conducted on a case-by-case basis to gain a deeper understanding of the context for each keyword. Concurrently with the second stage, the third stage involved conducting an interview with an employee from the company to gain clarity on problem definitions, competency issues, and important vocabulary terms, such as initial classes and individuals for the ontology.

The fourth step involved ontology modeling using the Protégé ontology modeling software, which was developed by Stanford University. Protégé enables the editing of ontologies and knowledge bases through a graphical interface with Java API (Sivakumar and Arivoli, 2011). This tool was selected due to its widespread usage in ontology development and its numerous functionalities, including ontology creation, modification, querying, and visualization (Schekotihin et al., 2018).

With the case study defined, and the ontology developed, a critical discussion is conducted, drawing upon concepts presented in the theoretical foundation, to evaluate the usefulness of the prototype and how it can integrate into the knowledge-centered support process. The methodology is illustrated in Figure 1.

The next chapter will explain the results of the methodological procedures carried out here.

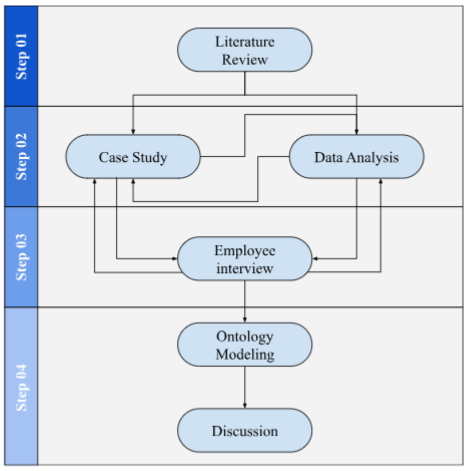


Figure 1: Flowchart of the summarized methodology.

## 4 RESULTS AND DISCUSSION

### 4.1 Context of the Study

The focus of the study is a company operating in the technology sector, specifically involved in developing solutions for the financial market. The company in question is based in the city of Campinas (SP) and employs over 900 individuals, as indicated by information obtained from its official website. The specific area of investigation is centered around providing support for a software product line aimed at facilitating instant payments.

As a result of high employee turnover, it is often the case that support staff lack the necessary knowledge to provide a satisfactory solution to customers, necessitating a search process to find another team member with the required expertise. Therefore, the outcome of this case study, which is the initial version of the ontology, aims to describe the ontological classes and individuals related to the most prevalent issues found in the provided dataset. The objective is to establish an efficient knowledge base that can be accessed through the ontology. The following section will delve into the information obtained through the exploratory data analysis conducted on the given dataset.

### 4.2 Quantitative Data Analysis

Prior to commencing the ontology development process, it is recommended to conduct a quantitative analysis of the data. This analysis serves two purposes: firstly, to assess the validity of the claim regarding support delays in resolving tickets, and secondly, to gather additional information from the dataset that can be utilized in subsequent stages.

Figure 2 illustrates the distribution of calls based on their classification. It is evident that the majority of calls fell into the ”Service Request” category, totaling 2116 occurrences. The second most frequent classification was ”Incident” with 1457 occurrences, followed by ”Doubt” with 806 occurrences. Conversely, the ”Problem” classification had a negligible presence in the dataset, comprising only 2 occurrences, which accounts for a mere 0.04 percent of the total. Consequently, this class was disregarded for future analyses.

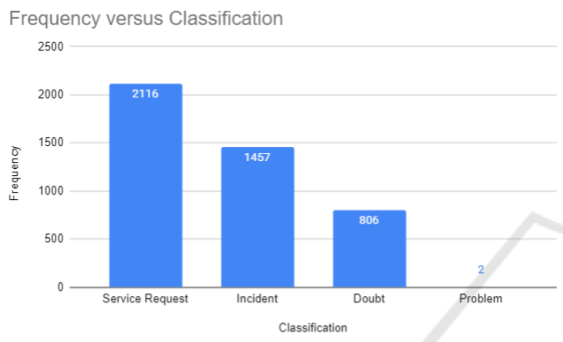


Figure 2: Bar chart for distribution of reasons.

Considering the company’s primary motivation to implement knowledge management practices, specifically the prevalence of undocumented knowledge in the support process, Table 1 presents the essential insights to comprehend this requirement.

Table 1: Summary of data statistics on Call Type.

| Type | average hours | median hours | maximum hours |
|---|---|---|---|
| Service Request | 111.06 | 47 | 2,963 |
| Incident | 102.17 | 40 | 2,701 |
| Doubt | 65.47 | 26 | 1,306 |

## 4.3 Domain Definition

In accordance with the recommendations outlined in Method 101, the definition of the ontology began by engaging in interactions with a collaborative member from the company and conducting data analysis. These initial steps were taken to gain an understanding of the ontology’s scope and domain. Several key questions were addressed during this process, including: ”What is the intended purpose of the ontology?”, ”What specific inquiries should the ontology be able to address?”, ”Who will be involved in the ongoing development of the ontology?”, ”What software services does the company offer that can be referenced in support tickets?”, ”What types of support tickets can be submitted via email and the portal?”, and ”What are the different topics (reasons/functionality as they appear in the dataset) that have been mentioned in the previously reported issues?”.

With this information, an understanding of the domain has been established, wherein the ontology will serve as a support tool for crafting responses to customer inquiries. Furthermore, it will assist in identifying the most frequent types of issues encountered in the customer journey, enabling prompt responses and definitive solutions, thereby mitigating the likelihood of future recurrences. Additionally, the support team will be tasked with maintaining an up-to-date knowledge base and ensuring continuous evolution of the ontology.

Lastly, the software services provided to customers encompass pre-configured environments tailored to fulfill technical and legal requirements mandated by regulatory entities. Additionally, an API service is available, enabling these environments to make calls and execute functions associated with the Instant Payment System. Customers can seek support for incidents, problems, questions, and services through email and the support portal. The classification of these requests can vary as per the customers’ specifications, which occasionally leads to misclassifications. Following pre-processing, a total of 301 distinct types were identified.

## 4.4 Definition of Terms

As discussed, an ontology can be viewed as a specialized vocabulary pertaining to a specific domain. Following the recommendations outlined in Method 101, the second stage of the method was omitted in this case, as the initial proposal for the ontology possesses an ad hoc nature. Consequently, the third step of Method 101, utilizing automated procedures with the Orange Data Mining tool, was employed to conduct an initial exploration of the most pertinent terms. Subsequently, a manual review was conducted, sifting through the terms present in the raw dataset and comparing them with the terms generated automatically, as illustrated in Table 2.

Afterwards, several straightforward steps were undertaken during the text mining process to extract keywords. These steps involved eliminating null elements in the relevant columns and performing text pre-processing tasks such as removing stop words, converting words to lowercase, and eliminating special characters.

Upon extracting the terms, it becomes evident from the summarized Table 3 that the calls align with the previously discussed domain and address the com-

Table 2: Example of available data in the call database.

| ID | Type | Subject |
|---|---|---|
| 452317 | Incident | problem with pix key registration |
| 452417 | Service | dynamic qr code expiration |
| 452435 | Service | messaging - production - conectivity test |
| 452486 | Service | synchronization Scheduling - access denied |
| 452489 | Service | reprocessing of pix transactions |
| 452497 | Service | pix not sent - mip does not notify transaction |
| 452501 | Incident | transactions are getting backlogged in the queue |
| 452512 | Service | account pi balance discrepancy |
| 452562 | Incident | errors generated on the occurrence reprocessing screen - production |
| 452642 | Service | implementation of dynamic qr code in mip |
| 452668 | Service | pix receipts with central bank rejection |
| 452720 | Service | payment rejection |
| 452844 | Incident | authentication failure in keycloak - testing environment - high priority |
| 453678 | Incident | intermittence in the operation of pix addressing |

petency questions. The terms that substantiate this assertion include ”environment,” ”failure,” ”unavailability,” ”pix,” ”spi” (Instant Payment System), ”version,” and ”key.” It is worth noting that these terms were selected based on their relevance, disregarding miscellaneous terms encountered during manual analysis, which will be further elaborated upon in the subsequent section explaining the classes.

## 4.5 Classes, Relations and Individuals

With the key terms identified for prototype development, the process of understanding commenced to determine the classes offered by the dataset, as well as the individuals within each class and the relationships that interconnect them to form a cohesive whole. The Protégé ontology editing software was utilized to iterate between ontology modeling and reviewing the data. This step posed significant challenges due to the limited availability of information and the inadequate classification of call types and reasons. However, the following classes were successfully defined:

- CustomerSPI: Refers to the direct customers of the company's services. These customers are small, medium, or large-sized establishments that have opted to utilize the Banking as a Service solutions provided by the company.
- PIAccount: Refers to the account used by end users when conducting any form of instant payment transaction, such as pix.
- Agent: Designates an employee of the company who is responsible for addressing incoming calls, irrespective of the call type.
- Environment: This class represents the type of environment utilized during the implementation process of the provided solutions, once the implementation is completed and the system is fully operational. Two instances have been identified: ”Environment for Approval” where tests are conducted to ensure the proper functioning of the

Table 3: Example of terms raised automatically.

| Words | TF-IDF |
|---|---|
| pix | 0.4961291716912596 |
| slowness | 0.4131082316814325 |
| production | 0.20207860874503155 |
| client | 0.017304412146895203 |
| homologation | 0.016168418252144694 |
| keys | 0.016139044668015756 |
| qrcode | 0.01607894193583626 |
| dict | 0.01497739958596505 |
| account | 0.01398328586318701 |
| key | 0.013683693406186373 |
| dispute | 0.013272857555930528 |
| problem | 0.013170211949826253 |
| notification | 0.013158701148155087 |
| bacen | 0.012206098834082134 |
| spi | 0.01114051326557928 |
| version | 0.010967236521929333 |
| addressing | 0.010783646007998277 |
| environment | 0.010713365428183354 |
| synchronization | 0.010121971026313355 |
| failure | 0.009995997632641442 |
| unavainability | 0.009878362015947875 |
| return | 0.009509278184916889 |
| portability | 0.009137496411406064 |

environment components and adherence to regulations set by the Central Bank, and ”Production Environment” where the system is actively used by customers for their daily operations as per the contracted services. This class exhibits a strong relationship with the ”Environment Elements” class.

- API: Refers to the API (Application Programming Interface) service that acts as a bridge between a customer's environment and the services managed and administered directly by the company. This API service enables seamless communication and interaction between the customer's environment and the company's services. The API class is closely linked to the ”API Elements” class, which encompasses the specific elements and components associated with the API service.
- Invocation: This is the central class in the case study, representing the tickets that are submitted to the support team. It is further divided into three subclasses: ”Environment Ticket”, which pertains to calls directly related to an environment; ”Functionality Ticket”, which relates to calls directly associated with API functionality; and ”Regulatory Ticket”, which involves a regulatory communication or requirement from the Central Bank or another governing body.
- ”Environment Element”: This class represents the components or characteristics that constitute an environment. Generally, an environment element can be a software module specific to the company's solutions or more general ones, denoted by the ”Environment Module” subclass. It can also encompass infrastructure-as-a-service or proprietary infrastructure providers, indicated by the ”Environment Provider” subclass. Additionally, hardware or virtual resources that form the infrastructure of the environment are represented by the ”Environment Resource” subclass.
- ”API Element”: This class represents the elements that constitute the API solution. An API element is further divided into two subclasses. ”Endpoint” refers to the electronic address and, upon ontology expansion, it should have attributes such as Method and fields present in the request body. The second subclass, ”API Functionality” pertains to high-level functionalities that can be accessed through requests to endpoints. It specializes in three types of functionality:
  - ”Account Functionality”: which operates on Instant Payment Accounts.
  - ”QRCode Functionality”: which operates on QR codes.
  - ”Transaction Functionality”: which handles money movement and payments.
- ”Cause”: This class holds significant importance for the subsequent retrieval section. It encompasses the various problems or general themes for which a ticket was opened to seek resolution. The database used revealed diverse reasons, but the most common and notable ones identified include: ”Missing Parameter”, ”Duplicate key”, ”Key Creation Failure”, ”Key Deletion Failure”, ”Security Failure”, ”Functionality Implementation”, ”Module Implementation”, ”Environment Unavailability”, ”Functionality Unavailability”, ”Regulatory Violation”, ”Environment Module Initialization”, ”Functionality Slowdown”, ”Documentation Request”, ”Pending After Deletion”, ”Key Portability”, ”Queue Problem”, ”Receipt Not Found”, ”Sufficient/Insufficient Balance”, ”Key Synchronization”, ”Interbank Transaction”, ”Environment Resource Exchange”, ”Environment File Upload”, ”Environment Versioning”, and ”Functionality Versioning”.
- ”Regulation”: This class pertains to current legislation and other guidelines established by regulatory institutions, such as the Central Bank. It represents the regulations and norms that govern the operations and practices within the domain.
- ”Type”: This class represents the type assigned to a ticket by the creator when reporting the situation that needs to be addressed. Based on the data, three types have been identified: Doubt, Incidents, and Services. These types categorize the nature of the reported issue or request.
- ”Solution”: This class serves as an aggregator for solutions pertaining to different calls. It is anticipated that after a call, individuals of this class will be created or updated to document the resolution or response provided.

With all the classes and individuals defined, the process of discovering and defining relationships between the components begins. The identified relationships, along with their constraints, are as follows:

- hasSolution: The ”Ticket” class has a domain, and the ”Solution” class has an image.
- apiIsComposedOf: Controls the ”API” class and targets the ”API Element” class.
- isComposedOf: The ”Environment” class has a domain, and the ”Environment element” class has an image.
- respondsToATicket: The ”Ticket” class has the domain, and the ”Agent” class is the target.

- ticketHandledBy: The inverse relation of ”respondsToATicket”. Therefore, the ”Agent” class has the domain, and the ”Ticket” class is the target.
- ticketCreatedBy: The ”Ticket” class has the domain, and the ”SPIClient” class is the target.
- createATicket: The inverse relation of ”ticketCreatedBy”. Therefore, the ”SPIClient” class has the domain, and the ”Ticket” class is the target.
- isAbout: It has four specializations, all with the ”Ticket” class as the domain and different targets. They are ”isAboutEnvironment” targeting the ”Environment” class, ”aboutEnvironmentElement” targeting the ”Environment Element” class, ”aboutAPIFunctionality” targeting the ”API Functionality” class, and ”aboutRegulation” targeting the ”Regulation” class.
- thereIsACause: The ”Ticket” class has the domain, and the ”Cause” class is the target.
- hasType: The ”Ticket” class has the domain, and the ”Type” class is the target.

A visual representation of the classes and their relationships can be seen in the diagram shown in Figure 3.

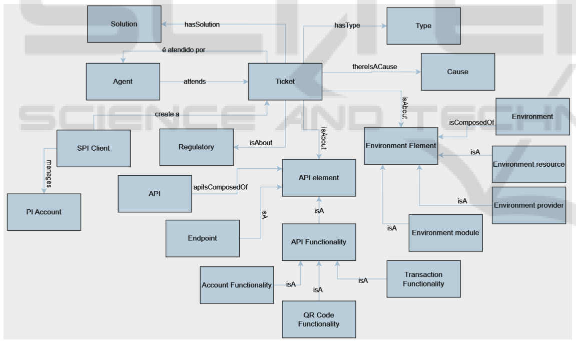
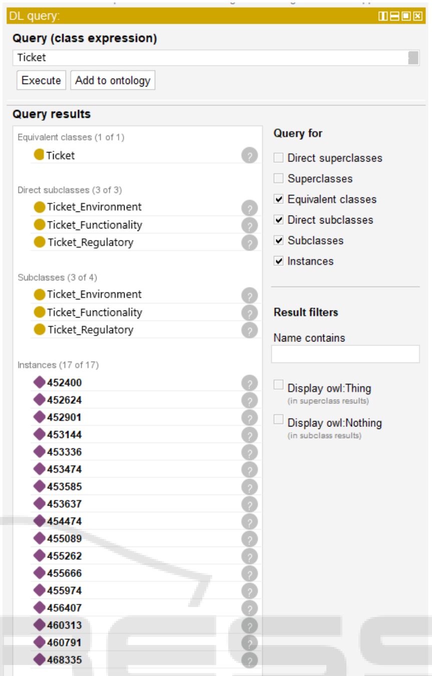


Figure 3: Ontology Prototype - Classes and their Relationships.

## 4.6 Information Retrieval

With the demonstration of the ontology prototype in the preceding section and considering the previous discussion, the problem that the company aims to address with a comprehensive ontology is to provide support agents with a means to retrieve solutions from previous tickets that exhibit similar characteristics to the current tickets. In the following section, we will present examples of utilizing the ontology for information retrieval. Despite the Protegé tool allowing a method to define instance titles, the ticket numbers were retained since it will make it easier for support analysts to use this numbering for retrieving the history of the same and, consequently, the existing knowledge in the history of these tickets.

The 17 distinct examples of calls depicted in Figure 4 have been generated based on the example provided in Table 2.


DL query:

Query (class expression)

Ticket

Execute | Add to ontology

Query results

Equivalent classes (1 of 1)
- Ticket

Direct subclasses (3 of 3)
- Ticket_Environment
- Ticket_Functionality
- Ticket_Regulatory

Subclasses (3 of 4)
- Ticket_Environment
- Ticket_Functionality
- Ticket_Regulatory

Instances (17 of 17)
- 452400
- 452624
- 452901
- 453144
- 453336
- 453474
- 453585
- 453637
- 454474
- 455089
- 455262
- 455666
- 455974
- 456407
- 460313
- 460791
- 468335

Query for
- [ ] Direct superclasses
- [ ] Superclasses
- [✓] Equivalent classes
- [✓] Direct subclasses
- [✓] Subclasses
- [✓] Instances

Result filters

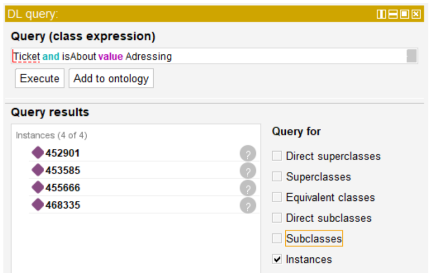
Name contains

- [ ] Display owl:Thing (in superclass results)
- [ ] Display owl:Nothing (in subclass results)

Figure 4: Example of Calls in the database.

The creation of these tickets was adjusted based on the meaning conveyed by each message, rather than solely relying on the information provided in the report. This adjustment was necessary due to the aforementioned misclassifications made by users. Taking this into consideration, ten of the tickets are categorized as Incidents, five as Services, and one as a Doubt.

The query scenario presented in figure 5 revolves around the requirement of finding tickets related to the Addressing module.


DL query:

Query (class expression)

Ticket and isAbout value Adressing

Execute | Add to ontology

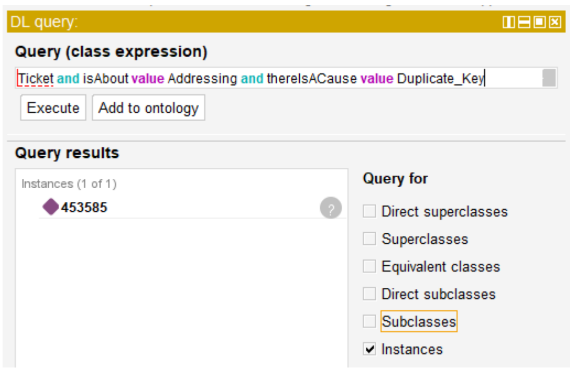
Query results

Instances (4 of 4)
- 452901
- 453585
- 455666
- 468335

Query for
- [ ] Direct superclasses
- [ ] Superclasses
- [ ] Equivalent classes
- [ ] Direct subclasses
- [ ] Subclasses
- [✓] Instances

Figure 5: Search for Addressing-type Tickets.

However, the search depicted in Figure 5 is not sufficiently specific, as in a database with thousands of calls related to the same module, it may not yield significant results. Therefore, a more specific query within the same module can be demonstrated in Figure 6.


DL query:

Query (class expression)

Ticket and isAbout value Addressing and thereIsACause value Duplicate_Key

Query results

Instances (1 of 1)

453585

Query for

- [ ] Direct superclasses
- [ ] Superclasses
- [ ] Equivalent classes
- [ ] Direct subclasses
- [ ] Subclasses
- [✓] Instances

Figure 6: Search for Tickets of Type ”Addressing” and with the Reason of ”Duplicate Key”.

Indeed, the query illustrated in Figure 6 is more specific and, as a result, more insightful compared to a generic query, as depicted in Figure 5.

## 5 CONCLUSION

After completing the initial stages of the case study, such as defining the domain, it has become evident that the company is facing more significant challenges beyond the need for a software artifact solely for information retrieval. One profound issue uncovered is the lack of a knowledge-oriented process.

During the process of modeling the ontology prototype, numerous difficulties were encountered in extracting relevant information from the provided data. Despite collaborating with a member of the support team, it was challenging to delve into specific domain details, such as the attributes comprising an environment, the elements of an Instant Payments account, the necessary training for the ontology maintenance team, and the cardinality of relationships. Another significant point addressed is that, depending on the company's level of commitment, there may be more comprehensive solutions available in the market that better aligns with their needs.

Considering the limited scope, this work successfully applied Method 101 and addressed the necessary aspects during the case study to answer the question posed in the introduction. It also proposed a plausible application for the developed ontology and may serve as inspiration for other researchers conducting interdisciplinary applied research in the fields of Knowledge Management and Computer Science.

As a suggestion for future work, it is recommended to enhance the ontology prototype and apply it in accordance with the discussions presented. A limitation of the case study was the absence of managerial considerations, such as controls and personnel, which could be addressed in the future using the KCS Evolution Loop. Additionally, applying ontologies in other areas of the company's software development, similar to the work of (Nardi and de Almeida Falbo, 2006) and (Isotani and Bittencourt, 2015), is suggested.

## ACKNOWLEDGMENTS

CAPES (Coordination for the Improvement of Higher Education Personnel) and the Postgraduate Program in Knowledge Management in Organizations at Cesumar University (Unicesumar) in the city of Maringá, PR, are acknowledged for providing the resources and means to carry out this research.

## REFERENCES

(1998). Ieee standard for software maintenance. *IEEE Std 1219-1998*, pages 1–56.

Alavi, M. and Leidner, D. E. (2001). Knowledge management and knowledge management systems: Conceptual foundations and research issues. *MIS quarterly*, pages 107–136.

Almeida, M. B. (2020). Ontologia em ciência da informação. teoria e método.

Almeida, M. B. and Barbosa, R. R. (2009). Ontologies in knowledge management support: A case study. *Journal of the American Society for Information Science and Technology*, 60(10):2032–2047.

Anquetil, N., de Oliveira, K. M., de Sousa, K. D., and Dias, M. G. B. (2007). Software maintenance seen as a knowledge management issue. *Information and Software Technology*, 49(5):515–529.

Chandrasekaran, B., Josephson, J. R., and Benjamins, V. R. (1999). What are ontologies, and why do we need them? *IEEE Intelligent Systems and their applications*, 14(1):20–26.

Chen, C., Lin, S., Shoga, M., Wang, Q., and Boehm, B. (2018). How do defects hurt qualities? an empirical study on characterizing a software maintainability ontology in open source software. In *2018 IEEE International Conference on Software Quality, Reliability and Security (QRS)*, pages 226–237. IEEE.

Damodaran, L. and Olphert, W. (2000). Barriers and facilitators to the use of knowledge management systems. *Behaviour & Information Technology*, 19(6):405–413.

Davenport, T. H. and Klahr, P. (1998). Managing customer support knowledge. *California management review*, 40(3):195–208.

de Faria, J. H. (2003). Economia política do poder: os fundamentos da teoria crítica nos estudos organizacionais. *Cadernos da Escola de Negócios*, 1(1).

De Reuver, M. and Haaker, T. (2009). Designing viable business models for context-aware mobile services. *Telematics and Informatics*, 26(3):240–248.

Demšar, J., Curk, T., Erjavec, A., Gorup, Č., Hočevar, T., Milutinovič, M., Možina, M., Polajnar, M., Toplak, M., Starič, A., et al. (2013). Orange: data mining toolbox in python. *the Journal of machine Learning research*, 14(1):2349–2353.

Dingsøyr, T. and Conradi, R. (2002). A survey of case studies of the use of knowledge management in software engineering. *International journal of software engineering and knowledge engineering*, 12(04):391–414.

dos Santos, G. S., Vieira, A. C. P., Pieri, R., Guimarães, M. L. F., Fabris, T. R., and Madeira, V. (2016). Análise das atividades de gestão do conhecimento entre extensionistas e empresas incubadas: estudo de caso da incubadora da unesc. *Revista de Extensão*, 1(1):90–107.

Gopalkrishna, B., Rodrigues, L. L., Poornima, P., and Manchanda, S. (2012). Knowledge management in software companies–an appraisal. *International Journal of Innovation, Management and Technology*, 3(5):608–613.

Hakim, A. A., Erwin, A., Eng, K. I., Galinium, M., and Muliady, W. (2014). Automated document classification for news article in bahasa indonesia based on term frequency inverse document frequency (tf-idf) approach. In *2014 6th international conference on information technology and electrical engineering (ICITEE)*, pages 1–4. IEEE.

Hindle, A. and Onuczko, C. (2019). Preventing duplicate bug reports by continuously querying bug reports. *Empirical Software Engineering*, 24(2):902–936.

Horrocks, I. (2008). Ontologies and the semantic web. *Communications of the ACM*, 51(12):58–67.

Isotani, S. and Bittencourt, I. I. (2015). *Dados abertos conectados: em busca da web do conhecimento*. Novatec Editora.

Lacombe, Francisco & Heilborn, G. (2013). *Administração:principios e tendências*. Saraiva Educação SA.

Lakemeyer, G. and Nebel, B. (2005). Foundations of knowledge representation and reasoning: A guide to this volume. *Foundations of knowledge representation and reasoning*, pages 1–12.

Lopes, G. R., Almeida, A. W. S., Delbem, A. C., and Toledo, C. F. M. (2019). Introdução à análise exploratória de dados com python. *Minicursos ERCAS ENUCMPI*, 2019:160–176.

Mao, H., Liu, S., Zhang, J., and Deng, Z. (2016). Information technology resource, knowledge management capability, and competitive advantage: The moderating role of resource commitment. *International Journal of Information Management*, 36(6):1062–1074.

Matsumoto, Y. (2014). *Software Engineering Basic Knowledge Body-SWEBOK V3. 0.* Ohmsha Co., Ltd.

McDaniel, M. and Storey, V. C. (2019). Evaluating domain ontologies: clarification, classification, and challenges. *ACM Computing Surveys (CSUR)*, 52(4):1–44.

Nardi, J. C. and de Almeida Falbo, R. (2006). Uma ontologia de requisitos de software. In *CIbSE*, pages 111–124.

Negash, S., Ryan, T., and Igbaria, M. (2003). Quality and effectiveness in web-based customer support systems. *Information & management*, 40(8):757–768.

Noy, N. F., McGuinness, D. L., et al. (2001). Ontology development 101: A guide to creating your first ontology.

Oliveira, M., Tenório, N., and Bortolozzi, F. (2022). A compreensão de reporte de bugs no desenvolvimento e uso de software: uma representação do conhecimento por meio de ontologia. *Revista Tecnologia e Sociedade*, 18(51):244–259.

Pigoski, T. M. (1996). *Practical software maintenance: best practices for managing your software investment*. Wiley Publishing.

Provost, F. and Fawcett, T. (2016). *Data Science para negócios*. Alta Books.

Rezgui, Y. (2006). Ontology-centered knowledge management using information retrieval techniques. *Journal of Computing in Civil Engineering*, 20(4):261–270.

Schekotihin, K., Rodler, P., Schmid, W., Horridge, M., and Tudorache, T. (2018). Test-driven ontology development in protégé. In *ICBO*.

Schneider, K. et al. (2009). *Experience and knowledge management in software engineering*, volume 235. Springer.

Serna, E. and Serna, A. (2014). Ontology for knowledge management in software maintenance. *International Journal of Information Management*, 34(5):704–710.

Sivakumar, R. and Arivoli, P. (2011). Ontology visualization protégé tools–a review. *International Journal of Advanced Information Technology (IJAIT) Vol*, 1.

SOUZA, H. A. (2015). Teoria geral da administração. *Rio de Janeiro-RJ: Seses*.

Statdlober, J. (2016). *Gestão do Conhecimento em Serviços de TI: Guia Prático*. Brasport.

Tougui, I., Jilbab, A., and El Mhamdi, J. (2020). Heart disease classification using data mining tools and machine learning techniques. *Health and Technology*, 10:1137–1144.

Trierveiler, H. J., Sell, D., and dos Santos Pacheco, R. C. (2015). A importância do conhecimento organizacional para o processo de inovação no modelo de negócio. *Navus: Revista de Gestão e Tecnologia*, 5(1):113–126.

Walz, D. B., Elam, J. J., and Curtis, B. (1993). Inside a software design team: knowledge acquisition, sharing, and integration. *Communications of the ACM*, 36(10):63–77.