Management of Implicit Requirements Data in Large SRS Documents: Taxonomy and Techniques

Dev Dave[1], Angelica Celestino[1], Aparna S. Varde[1,2], Vaibhav Anu[1]
1. Department of Computer Science, Montclair State University, NJ, USA
2. Visiting Researcher, Max Planck Institute for Informatics, Saarbrücken, Germany
(daved2 | celestinoa1 | vardea | anuv)@montclair.edu

ABSTRACT
Implicit Requirements (IMR) identification is part of the Requirements Engineering (RE) phase in Software Engineering during which data is gathered to create SRS (Software Requirements Specifications) documents. As opposed to explicit requirements clearly stated, IMRs constitute subtle data and need to be inferred. Research has shown that IMRs are crucial to the success of software development. Many software systems can encounter failures due to lack of IMR data management. SRS documents are large, often hundreds of pages, due to which manually identifying IMRs by human software engineers is not feasible. Moreover, such data is ever-growing due to the expansion of software systems. It is thus important to address the crucial issue of IMR data management. This article presents a survey on IMRs in SRS documents with the definition and overview of IMR data, detailed taxonomy of IMRs with explanation and examples, practices in managing IMR data, and tools for IMR identification. In addition to reviewing classical and state-of-the-art approaches, we highlight trends and challenges and point out open issues for future research. This survey article is interesting based on data quality, hidden information retrieval, veracity and salience, and knowledge discovery from large textual documents with complex heterogeneous data.

1. INTRODUCTION
The importance of high data quality in requirements engineering (RE) has long been recognized and well documented. Requirements elicitation is a critical RE activity that entails gathering data on system requirements from a multitude of sources including stakeholders, the existing system and its documentation, and various problem owners. Requirements explicitly expressed by the sources get recorded in requirements artifacts that are typically large Software Requirements Specifications (SRS) documents that could range to hundreds of pages with thousands of requirements. It is widely accepted that the industry is still striving to establish and apply practices to help identify crucial data on requirements that are often hidden or incomplete. Such requirements, also referred to as "Implicit requirements (IMRs)", are a known root cause of software project failure [1], [2], [3], [4].

Since SRS documents often run into 100s of pages, it is not feasible for human software engineers to detect IMR data manually. In the world of big data, SRS documents are growing as per the Vs of volume, velocity, variety, etc. For example, the volume of SRS documents can range to Terabytes, their velocity can be of the order of a huge SRS document generated per week, and their variety entails heterogeneous data including plain text, structured text, images, tables and other infographics. This makes it even more significant to dwell upon the issue of IMR management, clearly a non-trivial issue. IMR data deserves attention with respect to its classification as well as techniques, practices, and tools.

Hidden requirements data with detrimental impacts on data quality and software development was recently highlighted in seminal research, NaPiRE (Naming the Pain in Requirements Engineering) [1]. This is a family of surveys conducted by leading RE researchers to understand the state of the practice in RE data and the most critical RE problems. The NaPiRe survey, which obtained data from RE practitioners from across 10 countries and 228 companies, highlighted hidden requirements data, i.e., IMR-based data, as the most frequently cited RE problem (48% of survey respondents cited IMR data as a problem frequently affecting their project results). The NaPiRE survey also provided a probabilistic cause-effect model with major reasons on requirements data being hidden or implicit during RE. Practitioners noted that hidden requirements are caused by team members' lack of experience and knowledge about elicitation practices, inadequate use of data management techniques such as completeness checking, and not using available RE tools. Given that hidden requirements data (aka IMRs) are recognized as an important RE problem, this area calls for further study. Classical work by Spanoudakis [5] focuses on analogical reuse of SRS data, propounding a paradigm to compute similarities between SRS documents. Though modeling such analogies is a useful aspect of SRS documents, this study does not focus on the actual identification or management of IMRs.

Research on practices, tools, and techniques available for IMR data identification and management has been sporadic, being performed by a select few, e.g., [6], [7]. A tool SRElicitor [6] has been developed as a prototype

in the form of an Eclipse plugin, usable in RE to convert SRS data to plain text and then to the required format via adaptation of a Semantic of Business Vocabulary and Rules (SBVR). Other tools, NAI and ARUgen, surveyed in [7] have been developed for ambiguity resolution, helpful in addressing IMR data. A tool COTIR has been proposed in [8] and enhanced in [9] to integrate commonsense knowledge with textual data mining and ontology for early detection of IMR. While fundamental COTIR deals with plain textual data in SRS documents, its enhanced version aims to address finding IMR from images and tables via approaches such as CNN (convolutional neural networks). Likewise, there are interesting advancements that deserve attention in IMR identification.

The data science community on a broad scale has addressed the importance of data on implicit requirements and related terms such as hidden / ambiguous/vague requirements through works such as [10], [9], [11], [12]. These works bridge various perspectives, e.g., data quality, business management, machine learning, commonsense knowledge, ontology, fuzzy logic, etc. Such research has often been published in multidisciplinary venues. However, data scientists have not dwelt substantially on IMR data management. Yet, this is a significant paradigm in the world of big data today where huge complex SRS documents are generated regularly and need to be effectively harnessed for high-quality information. Big data on IMRs is ever-growing and mandates considerable attention. Moreover, IMR data management relates broadly to topics such as data cleaning [13] due to connection with missing information; data veracity [14] due to emphasis on authenticity and salience of information [15]; hidden data discovery [16], [17] due to focus on hidden requirements, complex text data analytics [18] due to heterogeneous textual data involved, and machine learning for data science [19] due to automation of knowledge discovery from large documents, all of which are of much interest to the data science community. Hence, this inspires a detailed study in the area and motivates our survey article.

To that end, this article is our attempt at presenting the state-of-the-research and state-of-the-practice in IMR identification and management through a review and analysis of the relevant literature. We focus on delineating a taxonomy of IMR in several categories accompanied by suitable examples to clarify the concepts. Additionally, we aim to inform the RE community of the techniques, practices, and tools used in IMR identification and management while also discussing the open issues in the area with pointers to future research directions.

2. IMR OVERVIEW & TAXONOMY

This section provides a brief introduction to software requirements engineering and IMRs followed by an IMR classification system that we have proposed.

2.1 Background on SRS and IMR Data

The software development life cycle (SDLC) typically begins with gathering data on requirements, followed by the requirements analysis phase that involves a detailed study of the needs the software is supposed to address. Often, some requirements related data is not well documented, and the burden of visualizing such data is left on the developer. Such vague requirements often result in software solutions not addressing the clients' needs completely. Most requirements not captured initially are discovered inadvertently in user acceptance stages [2]. Clients think that such requirements would be captured by software automatically. It is difficult to identify who is at fault for not capturing IMRs. To avoid such incidents, resulting in dissatisfied customers, it is important to focus on identifying IMR data.

Figure 1 adapted and redrawn from [21], gives a generic depiction of Requirements Engineering as a whole.

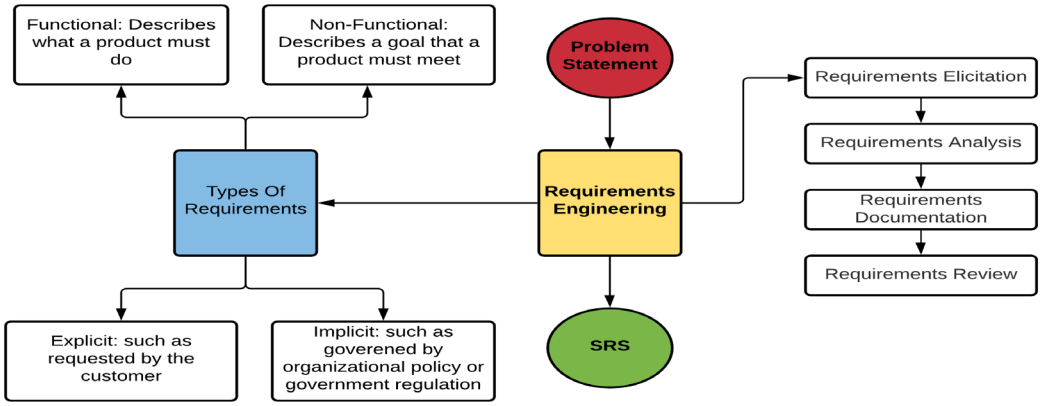

Figure 1. The requirements engineering (RE) process

Once the client presents the software problem to the developers, requirements data is collected before embarking on analysis. Many requirements are indeed explicitly captured. However, customers have other expectations: system security, availability, usability, performance, hardly defined in SRS documents. By default, and without writing, users expect that the system will always be available, be essentially secure, and perform its intended tasks well, without necessarily stating that. IMRs often look too simple due to which the customer views them as obvious, or too complicated such that the customer cannot visualize them.

Software requirements fall into 2 broad categories: functional and non-functional. Functional requirements define the functions of a system or its components, while non-functional requirements specify criteria that can be used to judge its operation, rather than specific behavior. Non-functional requirements are introduced during development [22]. Functional requirements are specified before the development begins. Functional development constitutes the earliest stage in the SDLC. The distinction between functional and non-functional requirements informs how each is handled during elicitation, documentation, and validation [22]. Often, developers ignore the assessment of non-functional requirements, which if identified can aid early detection and mitigation of risks. Speech tag parts constitute highly informative parts of the non-functional requirements. Based on this general background, we proceed with a thorough classification of IMR data.

2.2 Classification of IMR Data

In Table 1, we present a taxonomy of implicit requirements (IMR) data. This taxonomy is the main contribution of our survey and was developed based on this literature study and our expertise in the area. We are proposing that there are 5 categories of requirement data that are frequently missed (i.e., remain implicit). As shown in Table 1, our main IMR categories are: Security, Accessibility, Maintainability, Sustainability, and Usability. We describe the five categories below.

Data on security requirements can be explicitly stated but can also imply additional requirements (which are not overtly stated by the clients). Riaz et al. propose to identify security requirements data. Security categories in their study are: identification and authentication, availability, accountability, and privacy [23].

Accessibility requirements constitute another category of data often missed. This is due to a lack of proper development methods and authoring tools [24]. Web Content Accessibility Guidelines (WCAG) 2.0 and ISO 9241-171:2008 guidance on software accessibility provides a wide range of recommendations to make the web more accessible for individuals with disabilities. Fundamental categories in accessibility guidelines are: perceivable, operable, understandable and, robust [24].

Data on maintainability requirements is important during software design and implementation. Improving software processes can therefore enhance software maintainability [25]. Problems during development can lead to lower standards of maintainability. The ISO/IEC (International Organization for Standardization and International Electrotechnical Commission) 9126-1 2001 standard defines maintainability as the capability of a software to be modified. Software maintainability has 4 main categories: analyzability, changeability, stability, and traceability [25].

Sustainability requirements data is generally not supported by traditional software engineering methods. Yet it is very important, especially today with much focus on developing sustainable and environment-friendly systems. Sustainable development is defined as meeting the needs of the present without compromising the ability of future generations to meet their own needs [26]. Sustainability is mentioned as per 3 variables: time, function, and system. Time refers to the actual amount of time required to maintain or develop software; function refers to satisfying the task-based requirements, and system refers to humanity in its ecosystem. Note that the system usage aspect can be further divided into 5 sustainability dimensions, namely: economic, technical, social, individual, and environmental. The 5 dimensions serve to highlight the various potential impacts of the system.

Usability data affects the design of software and should be considered during the requirements phase. Usability is an important requirement as it improves productivity and customer satisfaction while reducing training and documentation costs as mentioned in an interesting piece of research. This research aims to embody HCI (Human Computer Interaction) principles. The author defines Functional Usability Features (FUF) based on HCI to make recommendations that the software should provide to the user. Usability requirements are related to the software's user interface and often remain hidden during RE. We found eight (8) sub-categories of usability requirements / features, namely: feedback features, undo features, cancel features, form or field validation, Wizard requirements, user expertise features, different languages and alert features.

Table 1 provides descriptions of the various categories in IMR taxonomy with their sub-categories, definitions and examples of software requirements. Hence, we have presented a detailed taxonomy of IMR data as widely accepted in the literature. We now proceed with tools and techniques used in the literature to address IMRs.

Table 1. Taxonomy of IMR Data: Categories, Definitions, and Examples

Category	Sub-category	Definition	Example of Software Requirement
Security	Identification and Authentication	Claimed identity of user must be valid for the user, process, or device	The system shall authenticate the user before any access to Protected Resources (e.g., PHI) is allowed, including when not connected to a network e.g., mobile devices.
	Availability	The system or component must be available to a certain degree	The system shall provide business continuity in the situation where the Electronic Health Record (EHR) system is not available by providing access to the last available clinically relevant patient data in the EHR.
	Accountability	Any action taken that affects the system can be traced back to the user responsible for the action	System shall keep track of every entry in the health record. Each entry will be identified with the author and should not be made/signed by someone other than the author.
	Privacy	The user can understand and control how their information is used in the system	The system shall allow nurses to provide legitimate care in crisis situations that may go against prior patient consent directives ("break the glass" situations).
Accessibility	Perceivable	Information and UI components must be presentable to users in ways they can easily discern the information	The system shall provide text alternatives for any non-text content so it can be transformed into other forms people need, such as large print, speech, symbols, or simpler language.
	Operable	User interface components and navigation must be usable	All functionality that is available by mouse is also available by keyboard.
	Understandable	Information and the operation of the user interface must be comprehensible	Significant changes on a web page do not happen without the consent of the user.
	Robust	Content must be adaptable enough that it can be interpreted reliably by a wide variety of user agents, including assistive technologies	The system shall reliably interpret markup by ensuring it is valid.
Maintainability	Analyzability	The software system is easy to diagnose or examine	All states, including fault conditions, are known.
	Changeability	The software system is easy to change or modify	The system shall be easily modifiable so it can be compatible with new hardware.
	Stability	The software system performs as expected and does not have any unexpected effects	During a 'sale/festival' season many people purchase commodities from web portals. The performance of a website must satisfy user expectations.
	Traceability	The ability to describe and follow the life of a requirement	The system shall have a functional audit trail by ensuring that a record is kept of all the changes made to the system.
Sustainability	Software Evolution Aspect	Sustainability of a software during its upkeep period by continuous monitoring of quality and knowledge management until it is replaced by a new system	The system shall be able to show what equipment is available, where it is located and in which state. This will avoid buying superfluous equipment and maximize the expected lifetime of equipment by doing maintenance when needed.
	System Production Aspect	Sustainability of a software system as product with respect to its use of resources for production is achieved. This can be achieved by using green IT principles.	The system shall enable the Event Manager to select Event Parts that require low energy and emit low CO2 amounts.
	System Usage Aspect	Sustainability of a software system in the usage process takes into account responsibility for the environmental impact and designing green business processes.	The system shall support the quality manager in efficiently assessing the sustainability of an event, in order to enhance firm's practices.
Usability	Feedback	The software informs the user about what is happening to the system	The system should always keep users informed about what is going on through appropriate feedback within reasonable time
	Undo	The software allows the user to Undo an action at several levels	If the user cancels, the system will go back to the first window listing what theatres there are.
	Cancel	The software allows the user to Cancel the execution of a command or an application	The user will be given the chance to cancel the operation, and the system will again display the selected theatre show times.
	Form/Field Validation	Improve data input for users and software correction as soon as possible	Form validation should be consistent and non-obtrusive in styling, location, and tone relevant to its
	Wizard	Assist users with tasks that require various steps and user input	An installer will be used to unpack the required libraries for the program via an install wizard.
	User Expertise	Allows adapting system functionality to users' expertise	The application enables users to select tutorials based on their level of expertise.
	Different Languages	Allows users to work with their language, currency, ZIP code etc.	The system requires an Internet connection and uses Google Translate to perform the text translation
	Alert	Warns users of actions with important consequences	At the end of the booking process, the system will display a window reporting whether the operation was a success or failure

3. IMR TECHNIQUES AND TOOLS

This section provides a comprehensive overview of existing IMR identification techniques and tools. Our literature selection criteria was: "tools and techniques used for IMR detection and classification".

3.1 Ontology, Semantics, and Pragmatics

Technological performances on management of IMR data are evaluated by various metrics and approaches. Various frameworks are used in detecting and managing IMR data, including case-based reasoning, ontology-based reasoning, and analogy-based reasoning, [2]. Some of these are illustrated in Figure 2 pertaining to ontology (redrawn based on [27]), and in Figure 3 about analogy (reconstructed from [28]). A reuse-based implicit requirements model (RM) is vital in facilitating the reuse of IMR data across projects when substantial similarities can be established between existing documented requirements and new ones [2].

Semantic matching is used to check for the similarity between requirements [2]. This deploys ontology to improve syntactic matches by exploring the relationship between semantics. Detecting IMR data is a collective responsibility of various team members. Analyzing risks of defects can save time and resources. Things that seem common sense can hurt system functionality, e.g., ignoring the fact that users can delete their accounts. Giving users control to delete their accounts can hurt an organization, especially by users parting with the company acrimoniously. Various tests require different tools and methods. Among the IMR data tested, an interesting method is the Implicit Priming Test (IPT) designed to determine connections between attributes and objects. The IPT further attempts to detect the strength of such connections. The speed of a response to an external stimulus is determined while factoring in the effects of priming [2]. By design, the IPT ignores explicit responses but returns values for implicit connections, e.g., emotional connection of a product itself and customers' thoughts about the given product.

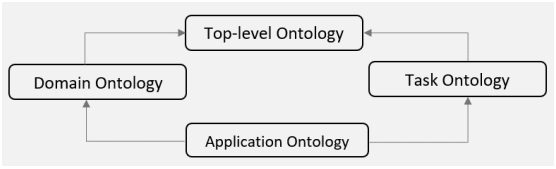

Figure 2. Types of ontology in IMR data management

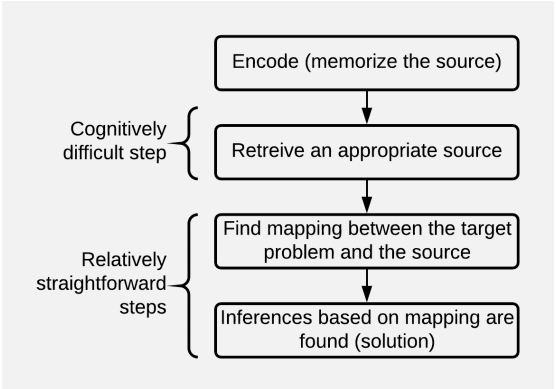

Figure 3. Data analogies map for IMR data management

The pragmatism of quality assurance (QA) teams should span beyond the benefits of the team to the entire company [29]. Challenging defects, regardless of their priorities, results in a decrease of unresolved issues. Technical knowledge is a critical requirement in analyzing and detecting IMRs, especially in complex systems. Various aspects of software system data might not be documented, mandating deep technical knowledge for testing. This includes testing for data integrity, application properties, flooding and draining queues, circuit breakers, and deadlocks. The deeper is the QA knowledge in a given domain, the more effective is the IMR data management. Automating and incorporating functional testing in the SDLC brings QA and development teams together, freeing more time for exploratory tests [22]. QA teams working closely with developers help in fault-finding and fixing issues early.

3.2 IMR Architectural Framework

Researchers address IMR data identification in different ways. While some prefer an unorthodox means of obtaining real-life views from the user perspective,

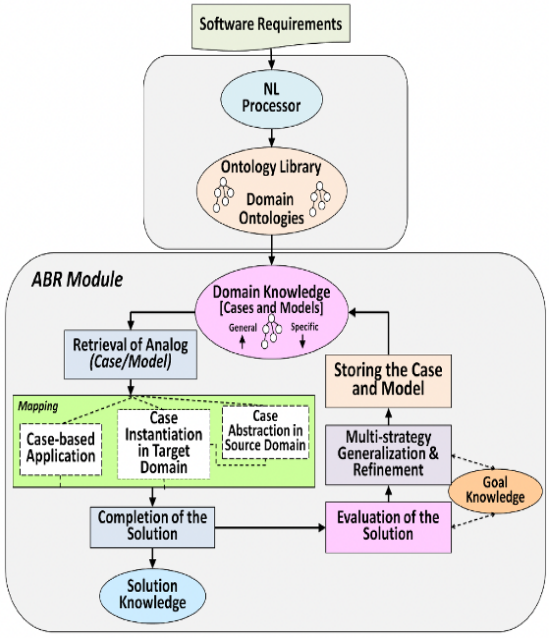

Figure 4. Excerpt of IMR architectural framework

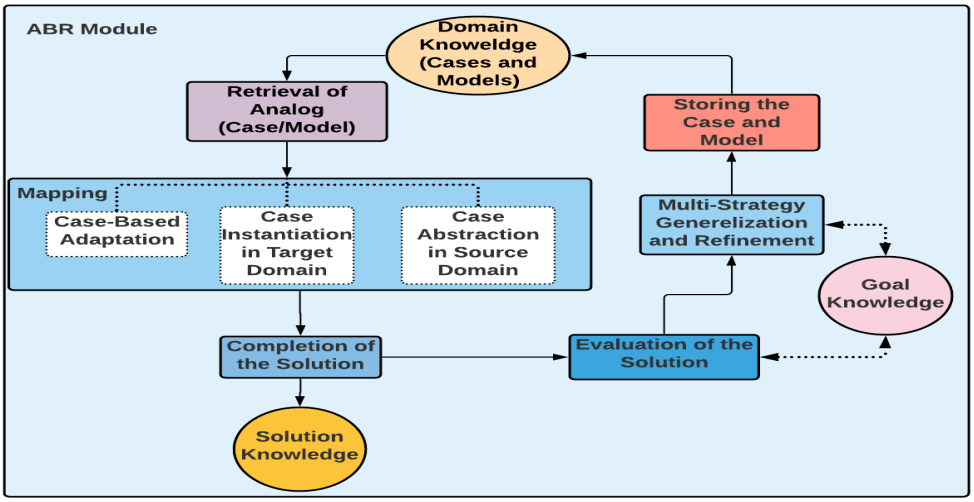

Figure 5. The PROMIRAR Tool

others opt for utilizing theoretical and conceptual frameworks [30]. An IMR architectural framework is shown in Figure 4 [31]. This provides support for managing IMR data as follows: (1) IMRs already documented, can be reused in new projects; (2) New IMRs previously overlooked can be discovered and stored thereby mitigating additional costs; (3) IMR data can be ranked based on established organizational standards, assigning the right priority level and scope to specific requirements.

This framework uses analogy-based reasoning for the reuse of previous requirements specifications, ontology to represent relevant domain knowledge crucial for managing IMRs, and natural language processing (NLP) to facilitate the analysis of textual data. This research aims to evolve a process framework for managing IMR within an organization. However, the system lacks human reasoning. Likewise, the ranking deployed also presents the scope for further research, interesting in data science. Ranking is a problem that is of interest to data scientists. Conducting ranking in this context would help to prioritize the requirements, so as to guide the software development. Including human reasoning in this process (potentially through machine learning), would help simulate the manner in which human software engineers can identify IMRs. This can be beneficial in software development processes.

3.3 The PROMIRAR Tool

PROMIRAR is a tool that facilitates the reuse of previously documented specifications to establish new requirements via analogy-based reasoning [27]. To identify the basis for analogy, understand similarities, and discover IMRs, NLP is used to analyze and extract important information, as shown in Figure 5 [27].

PROMIRAR takes preprocessed SRS documents as input data, uses NLP that empowers its feature extractor, and has an ontology library for knowledge representation of domain ontologies (specific purposes / general business rules). The Java Protege 4.1 ontology API is used to build the ontology library. Its feature extractor provides essential rules for classifying possible sources of IMR data in SRS documents. It has a heuristic classifier responsible for classifying the actual requirements based on intermediate outputs. Its analogy-based reasoner comprises 3 types of knowledge: domain, solution, and goal. Steps to use the PROMIRAR tool based on its architecture are: preprocess, import, analyze, identify and manage.

This tool addresses IMR data management by adapting analogy-based reasoning and provides good results. Yet, there is the potential for augmentation via research on advanced data management techniques in areas combining NLP, image mining, and data extraction from infographics, embodying domain knowledge and software engineering concepts. Such research can help improve IMR data management because many SRS documents contain complex data such as images, tables and other infographics. Extracting valuable information from these is a non-trivial process. While human software engineers can do this on a small scale, it is hard to achieve for huge volumes of complex data in SRS documents. Hence, discovering knowledge on IMRs from these documents in a seamless manner using data science approaches mentioned here would contribute to the RE phase, thus enhancing software development.

3.4 Using Templates for IMR Data Detection

An interesting practice used for detecting IMR data is the concept of "templates" to elicit implied security requirements [23]. The goal of this work is to determine whether automatically suggested security requirements templates help in efficient and effective requirements elicitation compared to a manual approach based on personal expertise. This tool aids the visualization of IMR data by providing a template as a checklist for developers to reference.

Dealing specifically with security, the process takes requirements-related artifacts as inputs. These include requirement specifications, feature requests, use case scenarios, etc. Based on these, it generates security requirements. This tool is a good visual aid and can be useful in conjunction with other IMR-related research, especially dealing with data security and privacy issues. It can propel further research in these areas and can be subjected to enhancement based on research outcomes.

3.5 InfoVis: IMR Data Visualization

A tool-supported approach is proposed to identify IMR data based on ambiguity and incompleteness [4]. This uses NLP techniques combined with visualization techniques to extract and interpret IMR data. It involves taking in user story requirements processed by a novel algorithm that deploys the Semantic Folding Theory (SFT) to calculate the semantic distance between 2 words to produce a similarity score. The algorithm also calculates the ambiguity score computed as a linear combination of term and context similarity. In this research, a visualization approach called InfoVis is developed that enables analysts to explore multiple viewpoints and extract IMR data. It harnesses Venn diagrams to simplify highlighting IMR data.

In their evaluation, term pairs in each category (low ambiguity, medium ambiguity, high ambiguity scores) are used from the WebCompany dataset with 98 user story requirements [4]. Students are presented with selected term pairs and asked to rate ambiguity. Results indicate a strong correlation between the score calculated by the algorithm and that given by students, the latter being the source of ground truth. Hence, this tool serves as an effective means for managing IMR data. This can serve as a benchmark for comparison further studies on IMR data management.

3.6 Machine Learning for IMR data

Binkhonain and Zhao [32] present a review of various machine learning approaches to identify and classify non-functional requirements. They present an overview of 16 machine learning approaches that utilize 5 Supervised, 5 Semi-Supervised, and 4 Unsupervised machine learning algorithms as shown in Table 2.

Table 2. List of Machine Learning Algorithms for Identification and Classification of NFRs

Supervised	Semi-Supervised	Unsupervised
•Support Vector Machines (SVMs) •Naïve Bayes (NB) •Decision Tree (DT) •K-Nearest Neighbors (K-NN) •Multinomial Naïve Bayes (MNB)	•Expectation-Maximization (EM) •Self-training •Active learning •Random Subspace Method for Co-training(RAS-CO) •Relevant Random Subspace Method for Co-training (Rel-RASCO)	•Latent Dirichlet Allocation (LDA) •K-means •Hierarchical Agglomerative •Biterm Topic Modelling (BTM)

All the machine learning approaches followed the same general process consisting of text preparation, learning, and evaluation. The text preparation phase consisted of text preparation and feature selection [32]. Text preparation involved NLP techniques such as Stemming, Stop word removal, tokenization, etc. For feature selection, the text document is converted into a numeric matrix using methods such as Bag of Words (BoW) and Term Frequency-Inverse Document Frequency (TF-IDF). The important features are selected from the matrix using methods such as information gain and Chi-square [32]. The learning phase consists of training and testing the machine learning algorithms on the preprocessed text dataset. The evaluation phase consisted of measuring the performance of the machine learning algorithms by using various metrics such as precision, recall, F-Score [32]. The key findings of this research were that ML-approaches generally perform well and achieve an accuracy of more than 70% when identifying and classifying NFRS, Supervised algorithms perform better than Semi-Supervised and Unsupervised algorithms with SVM and NB having the best overall performance, ML algorithms produce better results when individual words are used rather than phrases and without text preprocessing such as stemming and lemmatization [32]. Some of the challenges in this research area highlighted by Binkhonain and Zhao are the lack of shared datasets to train machine learning algorithms, lack of a standard definition of NFRs, and feature identification and selection [32].

3.7 Other Approaches in IMR Research

An interesting approach in IMR data management is COTIR, i.e., Common Sense Knowledge, Ontology, and Text Mining for Implicit Requirements. It consists of 6 steps: (1) Preprocess source documents to get requirements into text file format devoid of graphics,

images, and tables; (2) Select existing CSKB (Common Sense Knowledge Base) to be used for identification of IMR data; (3) Import SRS documents and domain ontology into the COTIR environment; (4) Click on the "analyze" function to allow the feature extractor to identify potential sources of IMR data in SRS; (5) See the potential IMRs identified and their recommended possible explicit requirements; (6) Seek expert opinion on IMR data, experts could approve/disapprove recommendations by adding/removing using the tool.

Emebo et al. envisage an enhancement of COTIR where a convolutional autoencoder can be used. This enables detecting IMRs in complex data (as opposed to plain text only in fundamental COTIR), e.g., deciphering text within images [9]. Facets of the enhancement are as follows: (1) SRS documents supply requirements data from which IMRs need to be identified. Data cleaning removes noise in the RE data. This step is performed in its NLP component. (2) The requirements author selects relevant knowledge from the CSKB and relevant domain ontology from the ontology library. Previously cleaned RE data with the selected knowledge and domain ontology are transferred to the CNN-based autoencoder component. (3) The autoencoder's input construction transforms the data into vectors for deep learning models. In an autoencoder, parameters are trained by minimizing differences between input and output layers in an unsupervised manner. (4) The trained model is applied to solve new IMR problems. This model is capable of encoding the word frequency vector of a new IMR feature into the hidden states. In this enhanced version, COTIR has fewer steps than its predecessor and takes in more input, while catering to more complex data.

Further, Emebo et al. revisit COTIR via a demo paper. They focus on explaining how commonsense concepts, ontological aspects, and mining of textual data help identify areas of explicit requirements where relevant IMRs may be hiding [33] therefore, making it an IMR-source localization tool. As per their claims, this is the first tool embodying commonsense knowledge for the detection of IMR data by aiming to simulate human reasoning. In the demo, the researchers use a course management system (CMS) example where the SRS document is based on explicit requirements available for the CMS. Possible IMR sources are analyzed by the feature extractor, with suitable recommendations being prompted for the CMS. The demo concludes that the COTIR tool reduces software defects by around 10% and enhances overall software quality by around 20% on an average [33].

Other approaches exist in the overall paradigm of IMR data management [34] that could refer to IMRs by different names such as hidden, vague, missing, ambiguous, incomplete, derived, or assumed requirements instead of the specific term\implicit requirements used herewith. Regardless of terminology and nomenclature, the detection and management of IMR data is of the utmost importance for good software

Table 3. Comparison of IMR Tools and Techniques

Techniques/Tools	Taxonomy	Data Model	Strengths	Weaknesses
Implicit Priming Test (IPT) [2]	General	Ontology, Semantics and Pragmatics	• Analyzing risks of defects • Determining connections between attributes and objects	• Relies heavily on retrieving previous IMR classifications results to classify future IMRs
IMR Architectural Framework [31]	General	Analogy-based reasoning	• Reuse of previous requirements, ontology and domain knowledge and NLP for textual analysis	• The system lacks human reasoning, new IMRs may be overlooked
The PROMIRAR Tool [27]	General	Analogy-based reasoning	• Feature extractor using NLP • Heuristic classifier to classify IMRs	• Can be improved by augmentation using techniques in NLP, image mining and data extraction
Using Templates for IMR Data Detection [23]	Security	Templates	• Aids visualization of IMR by providing a template as a checklist for developers	• Good visual aid but works better when used with other IMR tools
InfoVis: IMR Data Visualization & NLP [4]	Maintainability – Defect Detection	NLP combined with semantic similarity techniques	• Enables analysts to explore multiple viewpoints and extract IMR Data • Uses Venn diagrams to simplify highlighting IMR Data	• Effectiveness of tool needs to be tested at a larger scale • Algorithm for detecting ambiguity can be improved and tuned while avoiding over-fitting
Machine Learning Classification techniques [32]	General	• Supervised ML • Semi-Supervised ML • Unsupervised ML	• Average accuracy of above 70% when identifying and classifying NFRs • Supervised ML algorithms, specifically, SVM and NB perform the best on average	• Lack of shared datasets to train ML algorithms • Lack of standard definition for NFRs • Feature identification and selection

development with user satisfaction and is a good practice that should be emphasized in data science, software engineering, and related areas.

3.8 Comparison of IMR Tools, Techniques

Table 3 above presents a comparison of the various IMR tools and techniques discussed in the previous sections. The table highlights the data model used for each tool and technique along with advantages and disadvantages.

4. TRENDS IN IMR MANAGEMENT

Existing gaps in IMR data management are likely to propel further research trends in this area. Some works such as [12] observe gaps in the literature regarding the specification of data-flow requirements. IMR data can be represented as either structured, semi-structured, or unstructured data as requirements can be encoded without a standard format having text, figures, and infographics [35]. Future research in IMR management can focus on developing intelligent, interactive, user-friendly tools to identify, analyze, and specify IMRs. It can also focus on more automation in the RE phase, and on improving data verification to ensure identification of vital system features.

Trends in IMR management include several research issues, among which we can potentially address the following avenues that would be of interest to data scientists:

- Relationships between data on defects in the RE phase and actual sources errors causing those defects are often ignored. Tracing precise causes of defects can result in SRS documents with enhanced data quality which would lead to better IMR data management. This can possibly be explored with data science techniques such as association rule mining. It would help to find relationships of the type "X implies Y", where X can be the actual source error causing defect Y in the RE phase, hence rules of the type "Error implies Defect" can be discovered. Such knowledge discovery can help fix the root cause of problems leading to IMR related issues in SRS documents. Hence, this is a justifiable piece of research since it would augment the RE phase in software engineering, leading to better outcomes in software development as a whole.

- Thorough studies can be conducted with tools such as COTIR, NAI, SR-Elicitor, and ARUgen, used in contexts related to IMRs [33] addressing efficiency, accuracy, complexity etc. with respect to usefulness in facets such as ontology, knowledge bases, and other relevant data science concepts. This can be justified as follows. Considering ontology, it would be interesting to explore whether standards such as RDF (Resource Description Format) and OWL (Web Ontology Language) can be useful in IMR specification tasks, since that would create a streamlined manner of data exchange usable globally by software engineers and data scientists across industry and academia. Currently, there are no such ontological standards. Nor are there such existing studies relating IMR with data science paradigms. Likewise, creating knowledge bases, e.g. domain-specific KBs (finance, law, healthcare) with respect to IMR data can guide the RE phase of software development processes. Such research would help to use data science concepts within software development to resolve issues on IMRs.

- IMR practices should be integrated into education and training for students and working professionals, e.g., in courses such as Human-Computer Interaction, Database Management Systems, Machine Learning, and special topics courses in areas such as Data Quality, and Requirements Engineering. We should strive towards making this commonplace in academia and industry, as potentially paradigm-shifting best practices [36], [37]. The justification for this entails the constant growth of education and training to keep abreast with the latest technology. This is required by academic boards such as ABET (Accreditation Board for Engineering and Technology) and CAC (Computing Accreditation Commission). Such growth and advancement is also preferred by the corporate world when they hire fresh graduates, e.g. as software engineers, data scientists, full-stack developers. Hence, embedding IMR research within relevant courses in the realms of data science and software engineering is beneficial to education.

Some further research in the avenues mentioned here could be orthogonal to efforts of interest to the data science community. A few appealing works in line with such insights include: [38] that deals with data quality monitoring for constantly evolving big data focusing particularly on data veracity (in addition to volume, variety, etc.); [39] that addresses ontology compliance for query processing with enrichment; [16], [17] that entail hidden information discovery; [40] that intends to conduct query optimization with robustness and reduced complexity with emphasis on real-life workloads; [13] that encompass various perspectives of data cleaning; [14], [15] that address topics such as veracity and salience of information; [19] that addresses machine learning within database management, and [41] that propels database education in conjunction with natural language aiming to make this widespread.

Other interesting works include those in commonsense knowledge (CSK), e.g. [42] that presents a short survey on the usefulness of CSK, incorporating its derivation, knowledge base construction, as well as benefits in data

management and machine learning; and [43] that presents a tutorial on CSK extraction, compilation, and evaluation, explaining where CSK is significant and how it can be supplementary/complementary to deep learning. Since we aim to incorporate both deep learning and CSK in some of our future work on IMR data management, many studies surveyed in these articles [42], [43] provide useful references.

There is the potential for future research in IMR data management and related avenues that have heretofore remained areas of less substantial focus. We have outlined trends as envisaged by us upon a survey of related topics. Our work on enhanced automation in early IMR detection with usage in suitable applications [33], as well as further investigation with active practice in studies and training [36], [37] would contribute the two cents to the paradigm of IMR data management. This would help augment R&D and education in data science and software engineering.

5. CONCLUSION

Summary: Missed user requirements, also known as implicit requirements (IMRs), are a major contributor to software project failures. These can have other nomenclature such as hidden / vague / ambiguous / derived requirements. In this survey article, we have presented a background of IMR data along with a detailed taxonomy of terms, definitions, and examples associated with this paradigm, mainly encompassing the categories of security, accessibility, maintainability, sustainability, and usability.

IMR data is primarily available for software development projects that use the Waterfall model as SRS documents are not created for projects using the Agile model. Existing research has not focused on Agile requirements engineering and more specifically, on identifying hidden requirements from user stories. Even though many software development projects use the Agile model, not much research has been done on identifying hidden requirements. While this can be an issue of concern and calls for further research, we have not focused on that in this study. Our survey in this paper caters much more to the Waterfall model, and the IMR data management there.

We have provided an insight into classical as well as state-of-the-art tools, techniques, and practices in the overall context of IMR data management. We have outlined projects such as PROMIRAR, InfoVis, and COTIR. Suitable explanations with illustrations and discussions have been provided. The works surveyed in this article indicate that there is much emphasis on IMR data with its importance being realized in successful software development from user standpoints, yet there is scope for further research. We have listed open issues for future work, in our survey of the respective tools as well as in our section on trends in IMR management. These would contribute further to enhancing IMR detection and improving software development.

In short, this survey article provides novel insights for the data science community with respect to the challenging and non-trivial issue of IMR data management from huge SRS documents. It is interesting from the angles of data quality, hidden information retrieval, knowledge discovery, textual heterogeneous data, ontology, and semantics, all of which are avenues of interest to data scientists.

Roadmap: In our own future work, we aim to overcome a few limitations of the current survey and address some further challenges. First, we intend to extend this study by performing a more systematic literature review to help address more research studies (possibly not outlined in the current survey) that focus on various facets of IMR data.

Second, we intend to investigate in detail specific works of the literature in areas such as data quality, veracity, and hidden information retrieval for IMR management to outline specific research sub-problems that would be of joint interest to the data science and software engineering communities. These would provide the potential for MS Theses and Ph.D. Dissertations in the common areas, along with the scope for implementing useful software tools that cater to the interests of both communities, as well as publications in both venues.

Third, we aim to further automate the early detection of IMR data by leveraging deep learning techniques with the potential use of commonsense knowledge to localize the source of IMRs and assist their management. An important goal of our work here is to develop a large-scale tool for the automation of early IMR identification. We envisage that such a tool will be useful to software developers and will also help to train students and practitioners on adequate IMR detection in the requirements engineering phase.

Finally, we envisage making early IMR detection a common practice via dissemination and usage of our research and development efforts as well as active deployment of various IMR data management tools in real-world applications. We would reach out to our collaborators in academia to include IMR-related concepts and practices in their course syllabus, and to our industry collaborators to help spread the awareness of IMR data management in software developmental efforts. We anticipate that such a practice will enhance work in data science and software engineering.

6. ACKNOWLEDGMENTS

Dev Dave has been funded by a Graduate Assistantship in Computer Science at Montclair State University. Aparna Varde acknowledges the NSF grants MRI: Acquisition of a High-Performance GPU Cluster for Research and Education Award # 2018575, and MRI: Acquisition of a Multimodal Collaborative Robot System (MCROS) to Support Cross-Disciplinary Human-Centered Research and Education at Montclair State University, Award # 2117308. She is a visiting researcher at Max Planck Institute for Informatics, Saarbrücken, Germany in the research group of Gerhard Weikum, during her sabbatical. Vaibhav Anu hereby acknowledges the NSF grant, REU Site: Enhancing Undergraduate Research Experiences in Cyber-security and Privacy-Enhanced Technologies (NSF Award # 2050548).

7. REFERENCES

[1] D. M. Fernandez, S. Wagner, M. Kalinowski, M. Felderer, P. Mafra, A. Vetro, T. Conte, M.-T. Christiansson, D. Greer, C. Lassenius, et al. Naming the pain in requirements engineering. *Empirical software engineering*, 22(5):2298-2338, 2017.

[2] O. Daramola, T. Moser, G. Sindre, and S. Bi. Managing implicit requirements using semantic case-based reasoning. In *REFSQ, Springer LNCS*, pages 7915:172-178, 03 2012.

[3] O. Emebo and A. Varde. Early identification of implicit requirements with the cotir approach using common sense, ontology and text mining. Technical report, Fulbright Scholarship Program, Dept. of Comp. Sc., Montclair State Univ., NJ, 2016.

[4] L. Dalpiaz. Pinpointing ambiguity and incompleteness in requirements engineering via information visualization and NLP. In *REFSQ*, pages 119-135, 2018.

[5] G. Spanoudakis. Analogical reuse of requirements specifications: A computational model. *Applied Artificial Intelligence*, 10(4):281-305, 1996.

[6] A. Umber, I. S. Bajwa, and M. A. Naeem. NL-based automated software requirements elicitation and specification. In *Adv. in Comp.and Communications ACC*, Springer, pages 191: 30-39, 2011.

[7] U. S. Shah and D. C. Jinwala. Resolving ambiguities in natural language software requirements: A comprehensive survey. *ACM SIGSOFT Software Engineering Notes*, 40(5):1-7, 2015.

[8] O. Emebo and A. Varde. Common sense knowledge, ontology and text mining for implicit requirements. In *CSREA Press Intl. Conf. on Data Minin*g, pages 146-152, 2016.

[9] O. Emebo, V. K. Anu, and A. S. Varde. Identifying implicit requirements in SRS big data. In *IEEE Intl. Conf. on Big Data*, pages 6169-6171, 2019.

[10] F. Provost and T. Fawcett. Data Science for Business: What you need to know about data mining and data-analytic thinking. O'Reilly Media, Inc., 2013.

[11] M. Bhuiyan and M. A. Hasan. Interactive knowledge discovery from hidden data through sampling of frequent patterns. Statistical Analysis and Data Mining: The *ASA Data Science Journal*, 9(4):205-229, 2016.

[12] A. Marconi, M. Pistore, and P. Traverso. Implicit vs. explicit data-flow requirements in web service composition goals. In Intl. Conf. on Service Oriented Computing, *ICSOC*, pages 459-464, 2006.

[13] A. Arasu, S. Chaudhuri, Z. Chen, K. Ganjam, R. Kaushik, and V. R. Narasayya. Towards a domain independent platform for data cleaning. *IEEE Data Eng. Bulletin*, 34(3):43-50, 2011.

[14] K. Popat, S. Mukherjee, J. Strotgen, and G. Weikum. Credeye: A credibility lens for analyzing and explaining misinformation. In The Web Conference *WWW Comp*. Vol., pages 155-158, 2018.

[15] M. Ponza, L. D. Corro, and G. Weikum. Facts that matter. In Conference on Empirical Methods in Natural Language Processing, *EMNLP*, 2018.

[16] S. Ghosh, S. Razniewski, and G. Weikum. Uncovering hidden semantics of set information in knowledge bases. *Journal of Web Semantics*, 64:100588, 2020.

[17] F. M. Suchanek, A. S. Varde, R. Nayak, and P. Senellart. The hidden web, XML and the semantic web: scientific data management perspectives. In *EDBT Intl. Conf. on Extending Database Technology*, pages 534-537. ACM, 2011.

[18] S. Qiu, B. Xu, J. Zhang, Y. Wang, X. Shen, G. de Melo, C. Long, and X. Li. Easyaug: An automatic textual data augmentation platform for classification tasks. In The Web Conference, *WWW Comp.*, 2020.

[19] T. Kraska, U. F. Minhas, T. Neumann, O. Papaemmanouil, J. M. Patel, C. Re, and M. Stonebraker. Ml-in-databases: Assessment and prognosis. *IEEE Data Engineering Bulletin*, 44(1):3, 2021.

[20] A. Yague, P. Rodrguez, and J. Garbajosa. Optimizing agile processes by early identification of hidden requirements. In International Conference on Agile Processes and Extreme Programming in Software Engineering, *XP*. Springer, 2009.

[21] M. Adu. Inadequate requirements engineering process: A key factor for poor software development in developing nations: A case study. International Journal of Computer and Information Engineering, *IJCIT*, 01 2014.

[22] Z. Kurtanovic and W. Maalej. Automatically classifying functional and non-functional requirements using supervised machine learning. In Intl.

Requirements Engineering Conf., *RE*, pages 490-495, 2017.

[23] M. Riaz, J. Slankas, J. T. King, and L. A. Williams. Using templates to elicit implied security requirements from functional requirements - a controlled experiment. In ACM-IEEE Intl. Symp. on Empirical Software Engineering & Measurement, *ESEM*, page.

[24] P. M. J. A. Raul Minon, Lourdes Moreno. An approach to the integration of accessibility requirements into a user interface development method. *Science of Computer Programming*, Elsevier, 86(16):58-73, 2014.

[25] S.-J. H. Jie-Cherng Chen. An empirical analysis of the impact of software development problem factors on software maintainability. The Journal of Systems and Software, *JSS*, 2019.

[26] B. Penzenstadler. Towards a definition of sustainability in and for software engineering. In ACM Symp. on Applied Computing, *SAC*, pages 28:1183-1185, 2013.

[27] O. Emebo, O. Daramola, and C. K. Ayo. Promirar: Tool for identifying and managing implicit requirements in SRS documents. In *WCECS Conference*, 2018.

[28] J. Hey, J. Linsey, A. Agogino, and K. Wood. Analogies and metaphors in creative design. *International Journ. of Engineering Education*, 2008.

[29] M. Claypool, P. Le, M. Wased, and D. Brown. Implicit interest indicators. In Intl. Conf. on Intelligent User Interfaces, *IUI*, 2001.

[30] T. Arts, M. Dorigatti, and S. Tonetta. Making implicit safety requirements explicit - an autosar safety case. In *SAFECOMP*, 2014.

[31] E. Onyeka. A process framework for managing implicit requirements using analogy-based reasoning. *IEEE RCIS*, pages 1-5, 2013.

[32] Binkhonain, Manal, and Liping Zhao. "A Review of Machine Learning Algorithms for Identification and Classification of Non-Functional Requirements." *Expert Systems with Applications*: X, vol. 1, 12 Mar. 2019, p. 100001., doi:10.1016/j.eswax.2019.10000.

[33] O. Emebo, A. Varde, V. Anu, N. Tandon, and O. Daramola. Using commonsense knowledge and text mining for implicit requirements localization. In IEEE Intl. Conf. on Tools with Artificial Intelligence, *ICTAI*, pages 935-940, 2020.

[34] R. Ellis-Braithwaite. Analysing the assumed benefits of software requirements. *arXiv*:1305.3853, 2013.

[35] "Asif, Muhammad, et al. "Annotation of Software Requirements Specification (SRS), Extractions of Nonfunctional Requirements, and Measurement of Their Tradeoff." *IEEE Access*, vol. 7, 2019, pp. 36164–36176., doi:10.1109/access.2019.2903133.".

[36] V. Anu and A. Varde. Using commonsense knowledge and deep-learning based text mining to identify implicit software requirements. Technical report, Dept. of Comp. Sc., Montclair State Univ., NJ, July 2020.

[37] A. Varde and V. Anu. An integrated architecture of deep learning based approaches to identify implicit requirements during software engineering. Technical report, Department of Computer Science, Montclair State University, April 2021.

[38] D. Srivastava. Towards high-quality big data: Lessons from FIT. In IEEE *Intl. Conf. on Big Data*, 2020.

[39] J. Ao, Z. Cheng, R. Chirkova, and P. G. Kolaitis. Temporal enrichment and querying of ontology-compliant data. In New Trends in Databases & Information Systems *ADBIS*, volume 1259, 2020.

[40] R. Borovica-Gajic, S. Idreos, A. Ailamaki, M. Zukowski, and C. Fraser. Smooth scan: robust access path selection without cardinality estimation. *VLDB Journal*, 27(4):521-545, 2018.

[41] W. Wang, S. S. Bhowmick, H. Li, S. R. Joty, S. Liu, and P. Chen. Towards enhancing database education: Natural language generation meets query execution plans. In *ACM SIGMOD Intl. Conf. on Management of Data*, 2021.

[42] N. Tandon, A. S. Varde, and G. de Melo. Commonsense knowledge in machine intelligence. *ACM SIGMOD Record*, 46(4):49-52, 2017.

[43] S. Razniewski, N. Tandon, and A. S. Varde. Information to wisdom: Commonsense knowledge extraction and compilation. In *ACM WSDM* Intl. Conf. on Web Search and Data Mining, 2021.