



**HAL**  
open science

# Enabling industrial maintenance knowledge sharing by using knowledge graphs

Hicham Hossayni

► **To cite this version:**

Hicham Hossayni. Enabling industrial maintenance knowledge sharing by using knowledge graphs. Networking and Internet Architecture [cs.NI]. Institut Polytechnique de Paris, 2021. English. NNT : 2021IPPAS017 . tel-03545734

**HAL Id: tel-03545734**

**<https://theses.hal.science/tel-03545734>**

Submitted on 27 Jan 2022

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.



INSTITUT  
POLYTECHNIQUE  
DE PARIS

NNT : 2021IPPAS017

Thèse de doctorat



# Enabling Industrial Maintenance Knowledge Sharing by using Knowledge Graphs

Thèse de doctorat de l'Institut Polytechnique de Paris  
préparée à Télécom SudParis

École doctorale n°626 École doctorale de l'Institut Polytechnique de Paris (EDIPP)  
Spécialité de doctorat : Informatique

Thèse présentée et soutenue à Evry, le 17/12/2021, par

**M. HICHAM HOSSAYNI**

Composition du Jury :

Yacine Ghamri-Doudane Professor, Université de La Rochelle, France	Président
Fabien Gandon Research Director, Inria Université Côte d'Azur, France	Rapporteur
Giovanna Di Marzo Serugendo Professor, University of Geneva, Switzerland	Rapporteur
Fatna Belqasmi Associate Professor, Zayed University, UAE	Examineur
Juan Sequeda Ph.D, Principal Scientist, Data.world, USA	Examineur
Noel Crespi Professor, Institut Polytechnique de Paris, France	Directeur de thèse
Imran Khan Ph.D, Senior Principal Architect, Schneider-Electric, France	Encadrant



# Dedication

*To My Family*



# Acknowledgements

I would like to express my deep and sincere gratitude to Prof. Noel Crespi, my thesis director and supervisor for giving me all the support, encouragement, guidance, and freedom one could wish for.

I would like to acknowledge and give my warmest thanks to my thesis co-supervisor Imran Khan, Ph.D., who made this work possible. He taught me the principles of research, with lots of patience, and his insight and advice helped my work go in the right direction. I am extremely grateful for what he has offered me.

I'd like to thank the reviewers of my thesis; Prof. Gandon Fabien (Université Côte d'Azur) and Prof. Giovanna Di Marzo Serugendo (University of Geneva) for their valuable feedback to improve the quality of my work.

My profound love and respect goes to my parents, my wife, my children, and my brothers for their love, understanding, and who always boosted me with encouragement to achieve the new milestones in my life.

Finally, throughout my research journey at Schneider-Electric, I have met, interacted with a number of excellent people. They, on a smaller or a larger scale influenced me and helped me to see things from different perspective. I thank them for their support and shared knowledge.

Hicham Hossayni



# Abstract

Formerly considered as part of general enterprise costs, industrial maintenance has become critical for business continuity and a real source of data. Despite the heavy investments made by companies in smart manufacturing, traditional maintenance practices still dominate the industrial landscape. In this Ph.D., we investigate maintenance knowledge sharing as a potential solution that can invert the trend and enhance the maintenance activity to comply with the Industry 4.0 spirit. We specifically consider the knowledge graphs as an enabler to share the maintenance knowledge among the different industry players. In the first contribution of this thesis, we conducted a field study through a campaign of interviews with different experts with different profiles and from different industry domains. This allowed us to test the hypothesis of improving the maintenance activity via knowledge sharing which is quite a novel concept in many industries. The results of this activity clearly show a real interest in our approach and reveal the different requirements and challenges that need to be addressed.

The second contribution is the concept, design, and prototype of "SemKoRe" which is a vendor-agnostic solution relying on Semantic Web technologies to share the maintenance knowledge. It gathers all machine failure-related data in the knowledge graph and shares it among all connected customers to easily solve future failures of the same type. A flexible architecture was proposed to cover the varied needs of the different customers. SemKoRe received approval of several Schneider clients located in several countries and from various segments.

In the third contribution, we designed and implemented a novel solution for the automatic detection of sensitive data in maintenance reports. In fact, maintenance reports may contain some confidential data that can compromise or negatively impact the company's activity if revealed. This feature came up as the make or break point for SemKoRe for the interviewed domain experts. It allows avoiding sensitive data disclosure during the knowledge-sharing activity. In this contribution, we relied on semantic web and natural language processing techniques to develop custom models for sensitive data detection. The construction and training of such models require a considerable amount of data. Therefore, we implemented several services for collaborative data collection, text annotation, and corpus construction. Also, an architecture and a simplified workflow were proposed for the generation and deployment of customizable sensitive data detection models on edge gateways.

In addition to these contributions, we worked on different peripheral features with a strong value for the SemKoRe project, and that has resulted in different patents. For instance, one of the short-term evolutions planned for the SemKoRe is to enable predictive



maintenance features. Such features rely heavily on the analysis of time-series data. Thus, we developed and patented a novel method to query time series data using semantic criteria. It combines the use of ontologies and time-series databases to offer a useful set of querying capabilities, and the evaluation shows a good efficiency even on resource-constrained edge gateways.

In another hand, SemKoRe proposes an ontology-centric approach for the maintenance of knowledge sharing. Hence, the development and industrialization of SemKoRe's services require a good understanding of semantic web technologies. However, in our entity, most IoT solutions' developers have little or even no knowledge of semantic web technologies. Therefore, we designed and implemented an easy-to-use solution to help users and developers to instantiate and query OWL-based ontologies without any knowledge of semantic web technologies. This solution has been patented and turns out to be useful for other ontology-based projects.

Finally, we continue working on other innovative features in other to enrich the SemKoRe ecosystem and to offer a better experience to SemKoRe's stakeholders including developers, domain experts, and customers.

### **Keywords**

Industry 4.0, Industrial Internet of Things, Machine Maintenance, Knowledge Graphs, Knowledge Sharing, Privacy Preservation, Semantic Web.

# Résumé

Autrefois considérée comme faisant partie des coûts généraux de l'entreprise, la maintenance industrielle est devenue critique pour la continuité de l'activité et une véritable source de données. Malgré les sommes importantes investies par les entreprises dans la fabrication intelligente, les pratiques traditionnelles en maintenance dominent toujours le paysage industriel. Dans cette thèse, nous étudions le partage des connaissances de la maintenance comme une solution potentielle qui peut inverser la tendance et améliorer l'activité de maintenance pour se conformer à l'esprit de l'industrie 4.0. Nous considérons spécifiquement les graphes de connaissances comme un outil permettant de partager les connaissances de maintenance entre les différents acteurs de l'industrie. Dans la première contribution de cette thèse, nous avons mené une étude de terrain à travers une campagne d'entretiens avec différents experts aux profils différents et issus de domaines industriels différents. Cela nous a permis de tester l'hypothèse d'une amélioration de l'activité de maintenance via le partage des connaissances, qui est un concept assez nouveau dans de nombreuses industries. Les résultats de cette activité montrent clairement un réel intérêt pour notre démarche et révèlent les différents besoins et défis à relever.

La deuxième contribution est la conception et le prototype de "SemKoRe" ; une solution s'appuyant sur les technologies du Web sémantique pour partager les connaissances de maintenance. Il collecte toutes les données liées aux défaillances de machine, les structure dans un graphe de connaissances et les partage entre tous les clients connectés pour résoudre facilement les futures défaillances du même type. Une architecture flexible a été proposée pour couvrir les besoins variés des différents clients. SemKoRe a reçu l'agrément de plusieurs clients Schneider implantés dans plusieurs pays et de différents segments.

Dans la troisième contribution, nous avons conçu et mis en oeuvre une nouvelle solution pour la détection automatique des données sensibles dans les rapports de maintenance. En effet, les rapports de maintenance peuvent contenir des données confidentielles susceptibles de compromettre ou d'avoir un impact négatif sur l'activité de l'entreprise si elles sont révélées. Cette fonctionnalité est perçue, par les experts du domaine interrogés, comme un point essentiel et critique pour SemKoRe. Elle permet d'éviter la divulgation de données sensibles lors de l'activité de partage des connaissances. Dans cette contribution, nous nous sommes appuyés sur le web sémantique et le traitement du langage naturel pour développer des modèles personnalisés pour la détection de données sensibles. La construction et l'apprentissage de tels modèles nécessitent une quantité considérable de données. Par conséquent, nous avons mis en place plusieurs services pour la collecte collaborative de données, l'annotation de texte et la construction de corpus. Aussi, une architecture et un workflow simplifié ont été proposés pour la génération et le déploiement de modèles de détection de

données sensibles personnalisables sur les passerelles de périphérie.

En plus de ces contributions, nous avons travaillé sur différentes fonctionnalités connexes à forte valeur ajoutée pour le projet SemKoRe, et qui ont abouti à différents brevets. Par exemple, l'une des évolutions à court terme prévues pour SemKoRe est de fournir des fonctionnalités de maintenance prédictive. De telles fonctionnalités reposent fortement sur l'analyse de données de séries chronologiques. Ainsi, nous avons développé et breveté une nouvelle méthode pour interagir avec les données de séries chronologiques à l'aide de critères sémantiques. Elle combine l'utilisation d'ontologies et de bases de données de séries chronologiques pour offrir un ensemble utile de requêtes, et l'évaluation montre une efficacité même sur des passerelles périphériques aux ressources limitées.

D'autre part, SemKoRe propose une approche centrée sur l'ontologie pour le partage des connaissances de maintenance. Ainsi, le développement et l'industrialisation des services de SemKoRe nécessitent une bonne maîtrise des technologies du web sémantique. Cependant, dans notre entité, la plupart des développeurs de solutions IoT ont peu voire aucune connaissance du web sémantique. Par conséquent, nous avons conçu et mis en oeuvre une solution facile à utiliser pour aider les utilisateurs et les développeurs à instancier et à interroger des graphes de connaissances sans aucune connaissance des technologies du Web sémantique. Cette solution a été brevetée et s'avère utile pour d'autres projets basés sur des ontologies.

Enfin, nous continuons à travailler sur d'autres fonctionnalités innovantes pour enrichir l'écosystème SemKoRe et offrir une meilleure expérience aux parties prenantes de SemKoRe, notamment les développeurs, les experts du domaine et les clients.

### **Mots-clés**

Industrie 4.0, Internet des objets industriel, Maintenance industrielle, Graphes de connaissances, Partage de connaissances, Protection de données sensibles, Web Sémantique.

# Table of contents

<b>List of figures</b>	<b>15</b>
<b>List of tables</b>	<b>17</b>
<b>1 Introduction</b>	<b>19</b>
1.1 Motivation and Research Problems . . . . .	22
1.2 Research Methodology . . . . .	24
1.3 Contributions of the Thesis . . . . .	25
1.4 Overview of my publications . . . . .	26
1.5 Thesis organization . . . . .	27
<b>2 Background and state-of-the-art</b>	<b>29</b>
2.1 Introduction . . . . .	30
2.2 Industrial Maintenance - Normative definition . . . . .	30
2.3 Motivating Scenario . . . . .	31
2.4 Interviews for Maintenance Knowledge Sharing . . . . .	32
2.4.1 Setup . . . . .	32
2.4.2 Results . . . . .	33
2.4.2.1 End-to-end story for corrective maintenance . . . . .	33
2.4.2.2 Life-cycle of maintenance reports . . . . .	34
2.4.2.3 The current practices in terms of reusing or sharing the avail- able maintenance knowledge . . . . .	34
2.4.3 Maintenance Knowledge Sharing Challenges . . . . .	35
2.5 Existing Maintenance Knowledge Standards and Initiatives . . . . .	37
2.5.1 Existing Standards and Initiatives Limitations . . . . .	38
2.6 Maintenance Knowledge Sharing Requirements . . . . .	38
2.7 Industrial Maintenance Knowledge Management & Sharing - State-of-The-Art	39
2.7.1 Importance of knowledge collection and management for industrial maintenance . . . . .	39
2.7.2 Maintenance Data Management Systems . . . . .	40
2.7.3 Current practices in industrial maintenance knowledge management	41

2.7.4	Maintenance knowledge capturing and capitalization . . . . .	42
2.7.5	Maintenance knowledge sharing . . . . .	43
2.8	Lessons learned . . . . .	44
2.9	Summary . . . . .	45
<b>3</b>	<b>Industrial Maintenance Knowledge Sharing With Semantic Knowledge Graphs</b>	<b>47</b>
3.1	Introduction . . . . .	48
3.2	Ontology data models for maintenance knowledge capturing . . . . .	48
3.2.1	Machine Failure Ontology Model . . . . .	49
3.2.2	Machine Domain Ontology Model . . . . .	50
3.2.3	Upper-ontologies adoption . . . . .	56
3.3	Proposed Architecture . . . . .	57
3.3.1	High-Level Architecture . . . . .	57
3.3.2	Detailed Architecture . . . . .	58
3.4	SemKoRe Workflow & Process . . . . .	59
3.4.1	Case-Based Reasoning formalization . . . . .	61
3.4.1.1	Case description . . . . .	61
3.4.1.2	Case retrieval . . . . .	62
3.4.1.3	Case reuse . . . . .	63
3.4.1.4	Case revise . . . . .	63
3.4.1.5	Case retain . . . . .	63
3.5	Proof-of-Concept Prototype . . . . .	63
3.5.1	Implemented services . . . . .	63
3.5.1.1	Startup Commissioning . . . . .	63
3.5.1.2	Failure Data Collection . . . . .	64
3.5.1.3	Anonymization Service . . . . .	65
3.5.2	Failure Data Aggregation . . . . .	65
3.5.2.1	Failure Data Sharing . . . . .	66
3.5.3	Implementation Details . . . . .	66
3.5.4	Evaluation setup . . . . .	68
3.5.5	Performance measurements . . . . .	68
3.6	Learned Lessons . . . . .	70
3.7	Summary . . . . .	71
<b>4</b>	<b>Privacy-preserving Sharing of Industrial Maintenance Reports</b>	<b>73</b>
4.1	Sensitive Data in Maintenance Reports . . . . .	75
4.2	Sensitive Data Detection Approach in Maintenance Reports . . . . .	76
4.2.1	Overview . . . . .	76
4.2.2	Sensitive data detection flow in maintenance reports . . . . .	78
4.3	Data corpus collection & preparation . . . . .	79
4.3.1	IMDP - Industrial Machine Data Pool . . . . .	79
4.3.2	Data Sources and Data Structure . . . . .	81
4.3.3	Texts Annotation . . . . .	83

4.3.3.1	Automatic Annotation . . . . .	83
4.3.3.2	Manual Annotation . . . . .	84
4.3.4	Summary . . . . .	84
4.4	Machine ontology definition . . . . .	85
4.5	Proof-of-Concept Prototype . . . . .	86
4.5.1	Training Named Entity Recognition models for machine components detection . . . . .	86
4.5.1.1	Data input . . . . .	88
4.5.1.2	Data output . . . . .	90
4.5.1.3	Data preprocessing . . . . .	90
4.5.1.4	Training setup . . . . .	91
4.5.1.5	Evaluation setup . . . . .	92
4.5.2	Implementation details . . . . .	92
4.5.3	Evaluation . . . . .	93
4.6	Limitations and Improvement Areas . . . . .	94
4.7	Conclusion . . . . .	95
<b>5</b>	<b>Embedded Semantic Querying Tool for Distributed Time Series Data</b>	<b>97</b>
5.1	Proposed Solution . . . . .	99
5.1.1	Overview . . . . .	99
5.1.2	Past Event Detection Features . . . . .	100
5.2	Implementation details . . . . .	101
5.2.1	Overview . . . . .	101
5.2.2	DTSE Abstract layers . . . . .	102
5.2.2.1	Data Model Interface . . . . .	102
5.2.2.2	Time Series Interface . . . . .	102
5.2.3	DTSE Query Language . . . . .	102
5.2.4	DTSE query decoding . . . . .	104
5.2.5	Distributed Times Series Engine’s query processing . . . . .	105
5.3	Proof-of-concept Prototype & Evaluation . . . . .	106
5.4	Conclusion . . . . .	107
<b>6</b>	<b>Consistent Knowledge Graphs Manipulation for Semantic Applications</b>	<b>111</b>
6.1	Runtime API for consistent Ontology Instantiation . . . . .	112
6.1.1	Overview . . . . .	112
6.1.2	Data Manipulation Features . . . . .	114
6.1.2.1	Data Querying Feature . . . . .	114
6.1.2.2	Data Update Feature . . . . .	114
6.1.2.3	Parametrized SPARQL Queries . . . . .	114
6.2	Technical details . . . . .	115
6.2.1	Data Querying interfaces . . . . .	116
6.2.1.1	Get the list of classes . . . . .	116
6.2.1.2	Get the structure of a class . . . . .	116
6.2.1.3	Get the instances of a class . . . . .	117

6.2.1.4	Get the details of an instance . . . . .	118
6.2.2	Data update interface . . . . .	118
6.2.3	Consistent KG manipulation . . . . .	118
6.2.3.1	Class Description Compliance . . . . .	119
6.2.3.2	Value Restrictions Checking . . . . .	119
6.2.3.3	Cardinality Restrictions Checking . . . . .	120
6.2.4	Consistency Checking Alternative . . . . .	120
6.3	Conclusion . . . . .	120
<b>7</b>	<b>Conclusion and Future Work</b>	<b>121</b>
7.1	Summary . . . . .	122
7.2	Future Work . . . . .	123
<b>A</b>	<b>Paper I</b>	<b>137</b>
<b>B</b>	<b>Paper II</b>	<b>157</b>
<b>C</b>	<b>Paper III</b>	<b>167</b>
<b>D</b>	<b>Interviews Campaign Questions</b>	<b>177</b>

# List of figures

1.1	Machine Maintenance steps after a failure occurrence . . . . .	21
2.1	Normative Maintenance Types - NF EN 13306. . . . .	31
3.1	Failure ontology. . . . .	54
3.2	SemKoRe ontology design. . . . .	55
3.3	Ontology modeling approach . . . . .	56
3.4	SemKoRe's flexible architecture's configurations . . . . .	57
3.5	SemKoRe detailed architecture. . . . .	58
3.6	SemKoRe Workflow. . . . .	59
3.7	SemKoRe process. . . . .	60
3.8	CBR steps for machine failure solving . . . . .	61
3.9	SemKoRe implementation setup. Simulated virtual edge gateways with SemKoRe Agents managed by a SemKoRe server instance hosted on the Microsoft Azure cloud. . . . .	67
3.10	Execution time evaluation for the creation and querying of Failure instances	69
3.11	Failure ontology. . . . .	70
4.1	Detailed overview of sensitive data detection in maintenance reports . . . . .	77
4.2	Sensitive data detection flow for maintenance data sharing . . . . .	78
4.3	Example of Wikipedia page with highlighted links . . . . .	80
4.4	IMDP's architecture . . . . .	81
4.5	Screenshot of the data collection Popup . . . . .	82
4.6	Screenshot of the MDPI's collected taxonomy view . . . . .	83
4.7	MDPI's Annotation interface . . . . .	85
4.8	Machine ontology design . . . . .	85
4.9	T-Box & Machine ontology specifications . . . . .	86
4.10	Example of automatic detection of Machine Components and Mechanical Equipments in the text. . . . .	90
5.1	Overview of the Semantic Time Series Engine deployed on a gateway . . . . .	99
5.2	Distributed times series engines configuration . . . . .	100



5.3	Components view of the DTSE module and interactions with other services	101
5.4	Structures of the DTSE queries . . . . .	103
5.5	Example highlighting the differences between the Values Queries and the Time ranges queries . . . . .	104
5.6	Example of Abstract Syntax Tree (AST) generated for a query example . .	105
5.7	Distributed Times Series Engine's query processing diagram . . . . .	108
5.8	Execution time comparison of DTSE and Machbase . . . . .	109
6.1	Runtime API for Consistent KG Manipulation . . . . .	113
6.2	KG Manipulation Flow . . . . .	115
6.3	Parametrized SPARQL Query flowchart . . . . .	115
6.4	Node-Red flow implementation for the data querying feature . . . . .	116
6.5	Example of a class description . . . . .	117
6.6	Node-Red flow implementation for the data update features . . . . .	118
6.7	Diagram flow of consistent creation of new class instances . . . . .	119

# List of tables

3.1	Failure ontology classes description . . . . .	51
3.2	Failure ontology object properties description . . . . .	52
3.3	Failure ontology data properties description . . . . .	53
4.1	Classes description of the Anonymization rules, Machine model, and Machine taxonomy ontologies. . . . .	87
4.2	Object properties description of the Anonymization rules, Machine model, and Machine taxonomy ontologies. . . . .	88
4.3	Data properties description of the Anonymization rules, Machine model, and Machine taxonomy ontologies. . . . .	89
4.4	The dataset volumes used for the training of the NER models . . . . .	89
4.5	Number of instances of each class used in the dataset . . . . .	90
4.6	Training parameters for Spacy, CRF, and BERT . . . . .	91
4.7	Cross-validation evaluation of Spacy, CRF, and BERT NER models (P: Precision, R: Recall, F1: F1 score) . . . . .	94
4.8	Execution time of the NER models on laptop and Raspberry pi (ms/sample of 1000 words) . . . . .	94
5.1	Description of the DTSE queries fields . . . . .	103
5.2	DSTE evaluation queries . . . . .	107



Chapter **1**

# Introduction

## Contents

---

1.1	Motivation and Research Problems . . . . .	22
1.2	Research Methodology . . . . .	24
1.3	Contributions of the Thesis . . . . .	25
1.4	Overview of my publications . . . . .	26
1.5	Thesis organization . . . . .	27

---

During the past decade, Internet of Things (IoT) has emerged as a major paradigm shift in the technology world. With its promise to bridge the connection between digital and physical worlds, IoT has seen tremendous interests and growth from all sectors. Advances in cloud computing, Big Data, cheaper & ubiquitous connectivity, sensor technologies, and availability of different platforms have catapulted the IoT as a viable means for improved performance, efficiency, and better user experience in multiple segments of society. We have seen several estimates for IoT deployment and benefits like from CISCO [1] and Forbes [2].

The IoT movement entered into the industrial world as the Industrial Internet of Things (IIoT) amid the same promises of improved performance, efficiency, and better user experience. As an enabling technology for seamless connectivity, intelligent automation, data processing to gain insights and optimize the end-to-end processes to reduce costs, IIoT has become one of the key strategic items of Digital Transformation initiatives of many industries around the world [3].

Factories and industrial plants are key parts of the industries involved in manufacturing processes. Their impact on the modern world and its functioning is massive not just because of the goods that come out of them but also as a source of jobs and income to millions of people. Like other parts, factories and plants have also benefited from the IIoT revolution and are a major targets for digital transformation initiatives. Recent events like COVID-19 [4] have not only accelerated these initiatives but also has given us good visibility on the gains we can get on the productivity, efficiency, lower costs and safety by using the appropriate technology [5, 6].

Due to this, many industries are making heavy investments in smart manufacturing and production systems [7]. In return, they expect optimal and sustainable production with minimum maintenance efforts. This makes maintenance a real source of benefits rather than an enterprise cost, and one of the most important industrial activities.

Systems such as Computerized Maintenance Management System (CMMS), Manufacturing Execution System (MES), and Enterprise Resource Planning (ERP) are used to perform maintenance activities in several industries [8]. These systems provide many important features to do the routine maintenance activity or when failures occur. However, these systems are used to schedule, perform and optimize the maintenance process at a given location (within a factory or a site). There is no easy way, using these systems, to allow two different factories (or sites) to share the maintenance knowledge and details of a specific machine's maintenance operations without a human expert in the loop. This leads to some factories or sites being better equipped to deal with the failures thanks to the experienced workforce while others have to deal with longer downtime periods understanding how to solve the failures. Therefore, the ability to share maintenance knowledge in an open and standard manner is key for overall improvements and efficiency in the maintenance process.



Figure 1.1: Machine Maintenance steps after a failure occurrence

Another aspect is that today when a new machine is installed in a factory, there is no existing knowledge of its failures unless there is a human expert who has dealt with similar machines before. In any given factory, each failure is only discovered at its first occurrence and requires usually a long and costly curative maintenance operation. If several sites have the same machine (which is not uncommon) then chances are that each failure will be dealt with in isolation without taking benefit of the previous maintenance experiences performed on the same machine elsewhere. The time taken to determine the cause of the failure and find the right solution has a direct impact on the production capacity of the factory therefore sharing maintenance knowledge can bring huge benefits.

The maintenance process includes, as shown in Figure 1.1, diagnostics to determine the reasons for a failure, its impact, and to define and apply the correct repair procedures.

Considering the possibilities brought by IIoT and associated technologies, we can use this maintenance knowledge to provide improved services, predict failures and design better machines after learning their failures.

The work in this thesis aims to propose improvements in the maintenance activity by sharing the maintenance knowledge among multiple actors. We propose an open solution using Semantic Web and Knowledge Graphs to share the maintenance knowledge and experiences between the factories that own and operate the same types of machines. Our goal is to democratize the maintenance knowledge sharing approach for all industries from various domains and of various sizes. Additionally, we also addressed concerns of our internal

stakeholders on the data sensitivity topic and the ease of use of Semantic Web technologies for a typical developer when using the proposed solutions.

Concretely, the thesis makes five contributions. The *first* contribution is a field study through a campaign of interviews with domain experts from different domains to understand their needs and the challenges of sharing maintenance knowledge on a large scale. The *second* contribution is the concept, design, and prototype of the knowledge repository called "SemKoRe" which is a vendor-agnostic solution relying on the Semantic Web technologies to collect and share the maintenance knowledge. In the *third* contribution, we designed a novel solution for automatic detection of sensitive data in maintenance reports which we had to address as a priority topic based on the early feedback. The *fourth* contribution is a novel method to query time series data using semantic criteria and that supports distributed queries execution. A *fifth* contribution is an ontology tool to easily manipulate and instantiate ontologies without any knowledge of semantic web technologies.

## 1.1 Motivation and Research Problems

In the last decades, knowledge sharing in the technology world has become more democratized and more prevalent for individuals and professionals. It is a go-to approach in many domains like in computer software development through open-source software and tools, open datasets, crowd-sourcing, and the collective universal encyclopedia (like Wikipedia). Tools like GitHub and GitLab have demonstrated that Knowledge sharing is an effective medium for the exchange of ideas, validation of concepts, innovation, and collaboration on large scale.

Knowledge sharing has been adopted by several organizations to increase operational efficiency and staff productivity [9]. It also makes the expertise of specialists accessible to a large community which helps for individual growth and development. However, this trend has not gained traction in the industry yet, where despite the existence of digital solutions for maintenance knowledge management (e.g. CMMS, MES, ERP), traditional practices are still in force, such as paper-based maintenance reports or maintenance reports that are never reused. Also, human factors as well as lack of established policies make it hard to share the maintenance knowledge with other stakeholders.

Multiple standards and international consortia have promoted the sharing of industrial maintenance knowledge [21–27]. These initiatives have proven that sharing maintenance knowledge between multiple stakeholders can significantly optimize maintenance activity and improve overall efficiency. In practice, the shared knowledge could be used by the machine operators as a guide to reduce the diagnostic and repair times, target the failures' root cause(s), and improve the machines' efficiency. Another important effect of maintenance

knowledge sharing is that it will facilitate the transfer of maintenance expertise between experts and novices. This will lead to much faster handling of machine failures, training of new staff, and reducing the dependency on external maintenance companies. Even more importantly, it will reduce the number of accidents related to maintenance activities. However, to the best of our knowledge, there is still no solution to facilitate the maintenance knowledge sharing among multiple stakeholders.

The starting point of this thesis was an internal project at Schneider-Electric led by the Machine Solution Line of Business (MSol LoB) to provide new services and solutions to its customers like Original Equipment Manufacturer (OEM) machine builders. These OEM machine builders provide machines to some of the largest companies in the world. As part of their digital transformation journey, these companies want to avoid downtime as much as possible as well as the reduction in maintenance costs and better visibility about the performance of their machine during their operation. For OEMs these needs translated into benefits like new services to offer by gathering data from different life cycle phases of the machines like design, engineering, commissioning, operation, maintenance, and disposal or recycling. Another benefit is that OEMs can use the insights from the operation and maintenance phases to improve their own machine design process and identify why some parts of the machine fail more often than others under certain conditions. Schneider Electric, being the service provider itself, saw the business opportunities to provide these new services as well as predictive analytics to the OEMs and the companies who have service contracts with it.

These concrete business needs have fuelled our activity during this thesis and our approach of using Knowledge Graphs for maintenance knowledge sharing proved suitable for the MSol LoB needs. In fact, our work received the approval of several important clients located in USA, UK, France, Germany, Italy, and China, working in various domains such as Pharmaceutical, Automotive, Heating, Ventilation, and Air Conditioning (HVAC) and Food & Beverage.

With this motivation, used as basis, this thesis addresses the following questions:

- *Q1- What is the current state of the art dealing with the sharing of maintenance knowledge? What is the taxonomy and research works? What are the challenges and roadblocks of the knowledge sharing approach? Do industrial maintenance experts adhere to this approach? What are the benefits of the adoption of such an approach?*
- *Q2- How maintenance knowledge can be shared between different stakeholders? What will be an efficient architectural solution to accomplish this? Especially, how to manage the maintenance knowledge collection and sharing on the three IoT layers: Edge, Fog, and Cloud since not everyone is connected to Cloud? What features or services should*



*be part of the proposed solution?*

- *Q3- How to design an open and extensible solution that is applicable to multiple domains and works across silos? Particularly, how to capture and share the maintenance knowledge independent of any domain constraints?*
- *Q4- What is the nature of sensitive data in maintenance reports? How to avoid sensitive data disclosure during the sharing of maintenance knowledge? What techniques could be applied for the privacy-preserving needs and how?*
- *Q5- How Semantic Web technologies can be used to enrich and to pull out insights from IIoT historical data? More particularly, how to deal with the diversity of data sources in IIoT? How to have an efficient mechanism for querying historical data stored on multiple IIoT low-end gateways?*
- *Q6- How can Knowledge Graphs (KG) consuming and manipulation by application developers be facilitated? How to guarantee the consistency of the manipulated KGs? How can ontology engineers be involved to enhance the developers' experience?*

## 1.2 Research Methodology

In this thesis, the following research methodology is used to solve the identified research questions.

The first step of our work was to understand the current practices and issues with respect to the maintenance activity and data. We conducted a field study through an interview campaign with the domain experts from multinational companies working in different segments. These interviews helped us to understand the challenges faced by the maintenance teams and how technology solutions can help resolve their challenges. In parallel, we also conducted a literature study to analyze the existing solutions around industrial maintenance data sharing, and to identify their limitations and their applicability for general purpose usage.

The second step consisted of proposing the technical solution for maintenance knowledge sharing. During this step, we were working with the MSol LoB team to identify and satisfy the various needs and requirements of their customers. We proposed architectural solutions for different IoT layers: Edge, Fog, and Cloud. We also proposed a base data model, based on Semantic Web technologies, to capture the different aspects of machines and machines failures.

The proposed architectures and data models are implemented and evaluated using a proof-of-concept in which we implemented the identified services for the different IoT layers.

The technical solutions were well received but MSol LoB customers raised the issue of potentially sharing the confidential data during maintenance knowledge sharing. To the best of our knowledge, we did not find any existing solution for data anonymization for the maintenance data. Therefore, in the third step, we worked on a novel solution for sensitive data detection in maintenance reports. We also developed custom tools for sensitive data detection that can be easily adapted to cover the various needs of each customer.

In this thesis, we propose an ontology-based approach for maintenance knowledge sharing. Hence, the industrialization and extension of our solution require skilled IoT solutions developers with a good understanding of the Semantic Web technologies. However, in MSol Team, the majority of developers have little to no understanding of these technologies, therefore we implemented and patented a couple of novel services that aim to help the developers easily create their applications using our proposed solution.

### 1.3 Contributions of the Thesis

This thesis makes five contributions in total which are described here briefly and linked to the original research papers addressing them. The detailed contribution of each original research paper is presented in subsequent chapters. However, to avoid invention disclosure, we will include only the URLs of the patents that are publicly available.

In the first contribution of this thesis, to set the basis, we conducted a field study through a campaign of interviews with domain experts from multinational companies working in different segments. This survey is described in the background section and in the papers "Data Anonymization for Maintenance Knowledge Sharing" and "Privacy-preserving Sharing of Industrial Maintenance Reports in Industry 4.0". The survey was limited to manufacturing industries for which equipment maintenance is a major activity. On one hand, the survey highlighted several issues faced by these experts in their daily routine. We found a real interest in a maintenance knowledge sharing solution and these actors believed that it will be beneficial in the short term as well as in the long term. On the other hand, the survey allowed us to identify the various challenges that could impede the development of generic maintenance knowledge sharing solutions.

The second contribution of the thesis is "SemKoRe", a technical solution for maintenance knowledge sharing. We addressed most of the challenges identified through the interviews campaign. The overall concept of SemKoRe was presented in the paper "SemKoRe: Improving Machine Maintenance in Industrial IoT with Semantic Knowledge Graphs", along with possible research avenues. In this contribution, we present the architectural choices and the adequate data models to efficiently capture and share the machine failures. We also provide details of the services running in the Cloud, Fog, and Edge layers for sharing

the maintenance knowledge. During this work, we were worked with the MSol LoB team to identify and satisfy the various needs and requirements of their customers.

In the third contribution, we designed and implemented a novel solution for the automatic detection of sensitive data in maintenance reports. This contribution is described in detail in the paper "Privacy-preserving Sharing of Industrial Maintenance Reports in Industry 4.0" and is subject of the Patent "Data Sanitization Method for Industrial Maintenance Reports". In this work, we analyze the structures and the nature of sensitive data in maintenance reports and present the methodology to construct the needed data corpus in order to compensate for the lack of real maintenance reports. We also define a semantic-based approach combined with Natural Language Processing and Rule-Based techniques to build custom sensitive data detection tools that fit the needs of different stakeholders.

In the fourth contribution, we designed and implemented a semantic engine for distributed time-series data. This work is the subject of the Patent "Semantic Search Method for a Distributed Data System with Numerical Time Series Data". It simplifies the interaction with the time-series database in the Semantic-based SemKoRe's ecosystem. In this work, we defined a simple data querying language for non-experts to allow them to use semantic concepts and time-series data to get the needed information. The evaluation shows good efficiency even on resource-constrained edge gateways.

The final and fifth contribution is a REST API for consistent ontology instantiation. In this work, we address the needs of typical software developers who want to build new applications and services over the Semantic Web stack without deep expertise in it. The API allows to manipulate Knowledge Graphs with respect to the defined ontology models' constraints and to guarantee the consistency of the Knowledge Graphs without needing extra ontology tools like a reasoner. This work is the subject of the Patent "Runtime API for Consistent Ontology Instantiation", and turns out to be useful for other ontology-based projects at Schneider Electric.

## 1.4 Overview of my publications

**Paper I** is "*SemKoRe: Improving Machine Maintenance in Industrial IoT with Semantic Knowledge Graphs*". It presents the proposed approach for industrial maintenance knowledge sharing along with implementation details. This paper is available on: <https://www.mdpi.com/2076-3417/10/18/6325>.

**Paper II** is "*Data Anonymization for Maintenance Knowledge Sharing*". It provides the background and motivations of the need to avoid sensitive data disclosure during the maintenance knowledge sharing activity. It also describes the state-of-the-art techniques for data anonymization that can be applied for sensitive data detection purposes. This paper

is available on: <https://ieeexplore.ieee.org/document/9568260>.

**Paper III** is "*Privacy-preserving Sharing of Industrial Maintenance Reports in Industry 4.0*". It presents a practical approach for sensitive data detection in maintenance reports. Details of the prototype and performance evaluations are also presented. This paper is accepted and presented at the AIKE<sup>1</sup> (Artificial Intelligence & Knowledge Engineering 2021) conference.

**Patent I** is "*Semantic Search Method for a Distributed Data System with Numerical Time Series Data*", Patent Number WO/2019/104077. It introduces a new method to query times series data using semantic criteria. It is described in Chapter 5 of this thesis. This patent is available publicly on the URL: <https://patentscope.wipo.int/search/en/detail.jsf?docId=W02019104077>.

**Patent II** is "*Runtime API for Consistent Ontology Instantiation*", Patent US Filing Reference Number 17/360,923. It proposes a tool for manipulating Knowledge Graphs in a consistent manner for the users having little or no knowledge about the Semantic Web technologies. This patent is not publicly available.

**Patent III** is "*Data Sanitization Method for Industrial Maintenance Reports*", Patent filed on 07-10-2021, Reference Number 63/251,362.. This patent covers the same idea as paper III.

The author is the lead contributor in all the papers and patents and has led the implementation, prototyping, compilation of results, and writing of the papers. During the preparation of the papers and patents, the author discussed the progress in the meetings and incorporated the suggestions/inputs given by other co-authors and supervisors.

## 1.5 Thesis organization

The following chapters summarize the main contribution of the thesis and provide discussions and ideas for future work. Chapter 2 discusses the background, motivation, and summary of the state-of-the-art, and includes the global conclusions learned from the interviews campaign. Chapter 3 describes the SemKoRe approach for industrial maintenance sharing, it includes the design and implementation details and performance evaluation results. This chapter is the summary of paper I. Chapter 4 presents our approach for sensitive data detection in maintenance reports. It summarizes the contributions in Papers II and III and the main idea of patent III. Chapter 5 is devoted to the semantic engine for distributed time-series data. It is the summary of the Patent I. Chapter 6 describes the design and implementation of the REST API for consistent ontology instantiation work. It summarizes the contribution in Patent II. Chapter 7 presents the items for future work and a summary

---

<sup>1</sup><https://www.ieee-aike.org/>

of the thesis. Finally, the research papers are attached with this thesis in the following order. Paper I is in annex A, paper II in annex B, and paper III in annex C. The patents files are not included in this thesis, we include only the internet URLs of the patents publicly available.

# Chapter 2

## Background and state-of-the-art

### Contents

---

<b>2.1</b>	<b>Introduction</b>	<b>30</b>
<b>2.2</b>	<b>Industrial Maintenance - Normative definition</b>	<b>30</b>
<b>2.3</b>	<b>Motivating Scenario</b>	<b>31</b>
<b>2.4</b>	<b>Interviews for Maintenance Knowledge Sharing</b>	<b>32</b>
2.4.1	Setup	32
2.4.2	Results	33
2.4.3	Maintenance Knowledge Sharing Challenges	35
<b>2.5</b>	<b>Existing Maintenance Knowledge Standards and Initiatives</b>	<b>37</b>
2.5.1	Existing Standards and Initiatives Limitations	38
<b>2.6</b>	<b>Maintenance Knowledge Sharing Requirements</b>	<b>38</b>
<b>2.7</b>	<b>Industrial Maintenance Knowledge Management &amp; Sharing - State-of-The-Art</b>	<b>39</b>
2.7.1	Importance of knowledge collection and management for industrial maintenance	39
2.7.2	Maintenance Data Management Systems	40
2.7.3	Current practices in industrial maintenance knowledge management	41
2.7.4	Maintenance knowledge capturing and capitalization	42
2.7.5	Maintenance knowledge sharing	43
<b>2.8</b>	<b>Lessons learned</b>	<b>44</b>
<b>2.9</b>	<b>Summary</b>	<b>45</b>

---

## 2.1 Introduction

This chapter discusses the background, motivation, and state-of-the-art in the area of maintenance knowledge sharing. It addresses the following research questions:

*What is the current state of the art dealing with knowledge sharing for industrial maintenance data? What are the challenges and roadblocks of the knowledge-sharing approach? Do industrial maintenance experts adhere to this approach? What are the current practices in terms of: maintenance knowledge collection, reusing and sharing?*

At the beginning of this thesis work, it was observed that there is a lack of studies about the maintenance knowledge sharing on a large scale. Most of the works are focused on optimizing the maintenance operations or knowledge management and reuse for specific machines or for specific organizations or domains. And to the best of our knowledge, there is no existing study or research work that proposes a generic solution for the maintenance data sharing, and that can be adopted by industries from any domain and of any size.

In order to address this, we start with the normative definition of the the industrial maintenance. Then, we present some key motivating scenarios, with practical examples to help us envision the core idea of the problem domain. After, we present the details of the interview campaign we conducted to identify the needs and pain points of the domain experts and to discover the roadblocks that hinder maintenance knowledge sharing in the real-world. We also present a state-of-the-art study about the existing solutions that already adopted the maintenance knowledge sharing approach along with their application limitations, and the the different facets of maintenance knowledge sharing, such as the current practices in industry, the maintenance knowledge collection, reuse or capitalization and sharing.

## 2.2 Industrial Maintenance - Normative definition

To understand the context of this thesis, let's first start with the definition of maintenance. The French standard *NFX 60-010* [10] defines the maintenance as "*all the actions needed to maintain or restore an asset in a specified state or able to provide a specific service*". This standard was replaced by the European standard *NF EN 13306* [11] that extends the definition of maintenance to: "*All the technical, administrative and management actions during the life cycle of an asset, intended to maintain or restore it in a state in which it can perform the required function.*" According to the *NF EN 13306* standard, as shown in Figure 2.1, industrial maintenance is divided in two main categories: corrective maintenance and preventive maintenance. On the one hand, the corrective maintenance is performed after a breakdown or a failure occurrence. It can be performed immediately or deferred in time

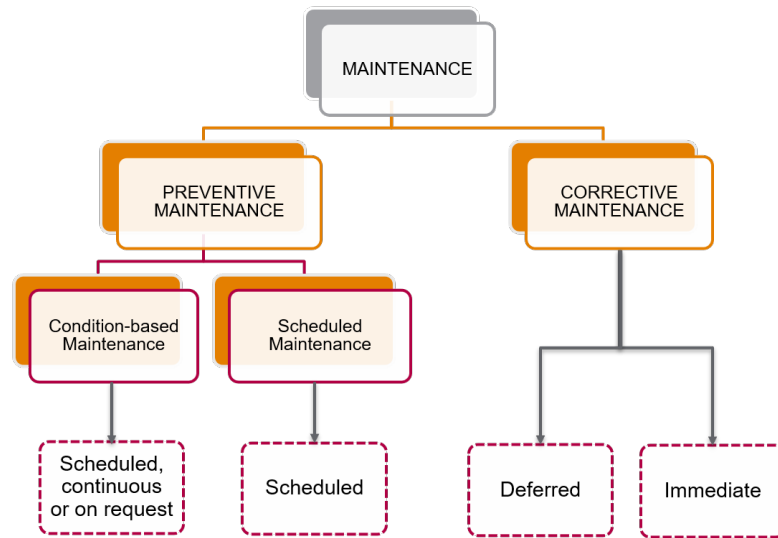


Figure 2.1: Normative Maintenance Types - NF EN 13306.

depending on the function of the asset or on the severity of its failure. On the other hand, the preventive maintenance is performed at predetermined intervals (i.e. Scheduled maintenance) or according to a prescribed criteria (i.e. Condition-based Maintenance) to reduce the probability of failures or degradation of the functioning of an asset [13].

NF X60-000 Standard [13] defines five levels of maintenance tasks. Level 1 consists of the simplest maintenance tasks that can be usually done by a machine operator whereas level 5 corresponds to the complex maintenance tasks that require significant resources usually related to the machine builder. The maintenance tasks of levels 2, 3 and 4 have increasing complexity and are done by maintenance technician or by external maintenance companies.

For each of these maintenance task, the machine operator or the maintenance technician prepares a maintenance report in which he describes the context and the details of the performed actions. The maintenance report can be on paper or digital such as a maintenance data management system, as expected from Industry 4.0 [14].

### 2.3 Motivating Scenario

The following scenario is based on our interactions with real customers who want to improve their existing maintenance process. Let us consider three actors: Bob the machine operator, Alice the maintenance technician, and Joe the OEM machine builder. On a given day, Bob is working on the factory floor operating several machines when suddenly one machine stops working. Bob spends some time fixing the issue himself but is unable to do so since Bob's main job is to operate the machine. He might be able to fix small issues due to his experience



but he is supposed to call a qualified technician for anything major. He then calls Alice to come to the factory floor to check on the machine. When Alice checks the machine she finds that she is also not able to solve the issue so she calls the OEM or Schneider Electric service bureau, where a machine expert guides her through the repair process. Finally, Alice is able to fix the issue and the machine starts working.

The whole process took a long time and while Bob is now able to operate his machine, if the same issue occurs in a similar type of machine located in a different city the same process would likely be repeated because only Alice knows how to quickly solve this particular issue. However, if Alice can describe what she learned from the service bureau and share her experience with the technicians in other sites by using some appropriate mechanism, they could all benefit from this common knowledge.

Another beneficiary of this common knowledge is Joe. Today, when Joe gets reports about the issues with his machines from different customers, he has no easy way to get the finer details that can only come from the technicians like Alice. These details could be useful and help him to understand why some of his machines are facing particular issues. This can help him to improve the design and engineering process of his machines, especially in the case of hundreds or thousands of machines being used worldwide, the scale of the problem and timely action in resolving the issue becomes hugely difficult. Another benefit is that using the insights from customer A, Joe can help customer B to quickly respond to machine issues while respecting privacy and the sensitive nature of the information, if both customers have the same type of machines.

## 2.4 Interviews for Maintenance Knowledge Sharing

### 2.4.1 Setup

To understand the perspectives of different actors involved in industrial maintenance operations, we conducted a survey through a set of direct interviews. We interviewed personas like plant managers, production manager, machine operators, maintenance technicians, maintenance engineers, service bureau managers, and innovation VP. The MSol LoB team identified 30 companies from various domains (Engineering, pharmaceuticals, automotive, HVAC, and Food & Beverage) and from 6 countries (UK, Spain, France, Switzerland, China, and the US). The survey was limited to manufacturing industries for which equipment maintenance is a considerable activity.

The author of the thesis has personally led the interviews with 9 domain experts lasting between 60 to 120 minutes. Each interview was recorded and transcribed.

Through this investigation, we tried to understand various aspects of industrial maintenance such as:

- The end-to-end story from the occurrence of a failure until the problem resolution.
- The life cycle of maintenance reports and the different actors generating or consuming them.
- The current practices in terms of reusing or sharing the available knowledge.
- The sensitivity of the maintenance knowledge.

The main questions that were asked during these interviews are presented in Annex A. These do not include the business analysis and the impromptu questions asked during the interviews.

The interviews allowed us to discover the added value of the research topic directly from domain experts and potential customers, and to learn about the various facets of maintenance activity. On the one hand, the survey brought forward several issues that are faced by these actors in their daily routine. We found a real interest in a maintenance knowledge sharing solution and these actors believed that it will be beneficial in the short term as well as in the long term. On the other hand, the survey allowed us to identify the various challenges that could impede the development of generic maintenance knowledge sharing solutions. Hereafter, we summarize the answers of the different domain experts to the most meaningful questions for the background analysis.

## 2.4.2 Results

### 2.4.2.1 End-to-end story for corrective maintenance

When a machine failure occurs, it is signaled, either by the system, in case of automatic surveillance of critical components or by the machine operator. The machine is then repaired by the machine operator in case of simple problems, or by the maintenance technician or external companies in case of complex problems. Large groups invest in service bureau entities, with qualified maintenance technicians and engineers, to manage the maintenance activity and to reduce the dependency on external companies.

In large companies, the service bureau provides the maintenance procedures to repair the known problems and defines the different parameters to measure before and after the intervention. In case of unknown failures, the service bureau remotely assists the maintenance technician using a digital camera. This approach is typical in most of the other companies. Usually, the machine operator or the maintenance technician improvises or, when available, follows the machine builder's instructions which are not necessarily applicable in all situations. When an external company intervenes, the solving procedure is almost never shared with the customer.

Once a machine gets repaired, a maintenance report is written describing the problem, the context, and the repairing procedure. The report is later extended by the maintenance manager or the technician with additional details such as the maintenance cost and the appropriate KPIs of the machine.

#### **2.4.2.2 Life-cycle of maintenance reports**

The maintenance report is most of the time written on paper by the maintenance technician before being transcribed later on CMMS or equivalent tools by the maintenance manager or the service bureau. However, according to the domain experts, several industries keep maintenance reports on paper or use document scanning as a digitization approach without any ability to track the maintenance tasks.

In the best-case scenarios, the service bureau takes care of structuring the details mentioned in the report and defining the step-by-step procedures that might be reused to solve future problems of a similar nature. However, in many factories, only a few details can be stored in the CMMS-like tools. For example, the tools used in two Schneider-Electric plants provide text fields limited to 40 characters only to capture the details. This forces the maintenance person to provide details in Word or Excel documents as an option. However, the overall process is designed in such a manner that only maintenance experts, working in the same plant or factory, are able to consume the details and reuse them in the future.

For legal reasons, the maintenance reports are stored for a specific duration, e.g. at least two years in Europe.

#### **2.4.2.3 The current practices in terms of reusing or sharing the available maintenance knowledge**

Most of the interviewed customers admitted that they have a lot of stored maintenance reports but do not know how to follow, analyze and capitalize on them. There are several limitations to this. The *first* limitation is that such reports are often indexed by the date, making a search of similar failures or frequent root causes of failures extremely tedious.

The *second* limitation is that none of the operators or maintenance technicians have access to the recorded maintenance reports. This is mainly due to the access limitations, restricted only to a service bureau or plant managers. Another aspect is that even with the right access, the complexity of using the maintenance tools is a major issue for people not proficient with the IT tools.

The *third* limitation is that due to the conciseness of the maintenance reports they either have incomplete details or no details at all about the repairing procedures. This conciseness is not only due to software limitations. But more to the lack of awareness about the importance of the quality of the maintenance report. Two important facts came into

the limelight during our interviews. *First*, the maintenance reports are never reviewed by anyone in the company, so spending an effort to prepare detailed reports seems unjustifiable to the maintenance technicians. *Second*, the maintenance technicians that are not trained nor motivated to provide detailed maintenance reports, prefer to keep their knowledge private as a guarantee of job stability.

For all these reasons, maintenance reports are almost never reused to solve similar problems.

It happens sometimes that the repairing procedures of some machine failures are only known to a particular maintenance technician in the company. And when that failure occurs, the production stops until the maintenance technician fixes the problem. This makes the company extremely dependent on individuals, and the high turnover rate becomes harmful for businesses.

### 2.4.3 Maintenance Knowledge Sharing Challenges

Hereafter, we compiled a list of the most important challenges of maintenance knowledge sharing according to the interviewed experts:

1. **Business culture:** Sharing knowledge is not a common activity in most organizations. It has been demonstrated that individuals are rewarded mostly for what they know, and not what they share [18]. The competitive environment that encourages individual instead of collective productivity has taught employees to consider their knowledge as their own property, and that to deepen and defend their knowledge is the main way to keep their jobs within the organization. This can be translated as a lack of trust between the employers and the employees, which is a mandatory ingredient for successful knowledge sharing in organizations [19]. Also, today, when a maintenance operation is done by an external company, the produced maintenance report is usually not reusable. In fact, most of the time the maintenance details are missing or encoded depending on the rules of the external company. This lack of clarity is, most of the time, intentional to avoid transferring useful knowledge and to maintain the dependability of the customers on the external company's services.
2. **Maintenance data collection:** The collection of good quality maintenance data<sup>1</sup> is a mandatory step before extracting and sharing useful maintenance knowledge<sup>2</sup> from it. It is a long-term activity that requires competent and well-trained personnel to

---

<sup>1</sup>Maintenance data refers to maintenance reports created to document a maintenance operation. These may contain data about the machine's components, description of the failure, and details of the maintenance procedure.

<sup>2</sup>By maintenance knowledge we refer to the result of an aggregation of maintenance data collected by many entities that is all relative to the same machine type.

guarantee the quality and usefulness of the collected data. However, this is usually seen as a marginal and costly process instead of being considered a solid investment for the future [20].

3. **Human factors:** Many people may be reluctant to write a detailed report when they are not confident in their own expertise and want to avoid being judged. Also, when failure frequency is high, many maintenance operators believe there is no time to write a detailed report and thus may produce minimal reports with little or no transferable knowledge.
4. **Common maintenance taxonomy:** Collecting good quality maintenance data is not enough to make it shareable. Entities exchanging their maintenance data must have the same understanding of the shared data and hence, use the same vocabulary or taxonomy. For instance, it is common that one maintenance task might be described differently by two (or more) maintenance technicians even within the same factory. In fact, due to the employee turnover and age structure, technicians usually use the vocabulary with acronyms or abbreviations learned through previous experiences in different manufacturing domains, and often there is a lot of heterogeneity involved.
5. **Legal aspects:** Every maintenance report is a potential legal liability. In fact, the entity producing the maintenance report is legally responsible for the material damages that may be caused by the application of the report's instructions. The responsibility could be penal in the case of human damages. Due to this issue of responsibility, some large companies destroy their maintenance reports beyond a certain legal period (e.g., 2 years in Europe) to avoid any future problems associated with their reports.
6. **Sensitive data disclosure:** Maintenance data may contain sensitive information that could compromise or negatively impact a company's activity. Thus, companies often choose to keep all their data secret to ensure business stability. Therefore, judging a piece of information to be sensitive remains a totally subjective decision based on the interviewed experts. Some believe that their maintenance data are not sensitive as they are using standard machines that can be found in several factories around the world. Others believe that even on standard machines, the machine configuration can be sensitive and be part of the competitive know-how of the company. However, most of the interviewed experts agree on the need to preserve data privacy during the maintenance knowledge sharing activity.

It is important to note that in this thesis, we only focus on challenges 4, 5, and 6. We determined that challenges 1, 2, and 3 are more relative to the business culture and are out of the scope of our research work.

## 2.5 Existing Maintenance Knowledge Standards and Initiatives

There are many international standards and consortium agreements that promote and adopt the maintenance knowledge sharing approach. Some of the most well-known ones are:

- The OREDA project [21]: Offshore and Onshore Reliability Data for oil and gas industries, which led to the ISO 14224 international standard [22]. It is a project organization with 7 to 11 oil and gas companies as members, managing data of 292 installations and 18000 equipment units. Running for more than 35 years, OREDA has led to significant cost savings in the development and operation of platforms<sup>3</sup>, and has helped the participating oil companies to save \$70M.<sup>4</sup>
- SPARTA [23]: System Performance, Availability and Reliability Trend Analysis adopted by 9 wind energy companies in the UK. It manages and exchanges data of 19 wind farms and 1256 turbines in 2020. It covers more than 60% of all offshore wind power generation in the UK. Thanks to SPARTA, the average number of crew transfers per turbine in the UK fell by 50% between 2014 and 2018, to around six trips per year<sup>5</sup>.
- WInDPool [24]: stands for Windenergy-InformationData-Pool, for wind energy industries in Germany. Adopted by 7 members to exchange the maintenance data for a fleet of 640 wind turbines onshore and 297 wind turbines offshore with an expectation of more than 1M euros of costs savings on the maintenance activity<sup>6</sup>.
- The Configuration Data Exchange [25]: launched by General Electric Aviation for the world-wide aviation industry. \$4 billion savings are anticipated across the sector thanks to the configuration data exchange solution<sup>7</sup>.
- OPDE [26]: The International Pipe Failure Data Exchange project; and
- ISO 6527 [27]: A Reliability Data Sharing standard for nuclear energy producers.

Through these standards and initiatives, the maintenance knowledge sharing concept has proven its efficiency in the improvement of equipment reliability, in the enhancement

---

<sup>3</sup><http://www.datsi.fi.upm.es/~rail/new/WP2/OREDA-history.htm> (accessed 17 Feb, 2021)

<sup>4</sup><https://www.oreda.com/join-us> (accessed 17 Feb, 2021)

<sup>5</sup><https://ore.catapult.org.uk/wp-content/uploads/2018/11/SPARTA-Portfolio-Review-201718-1.pdf> (accessed 17 Feb, 2021)

<sup>6</sup>[https://wind-pool.iee.fraunhofer.de/opencms/export/sites/WInD-Pool/img/WInD-Pool-Business-Case\\_ENG.pdf](https://wind-pool.iee.fraunhofer.de/opencms/export/sites/WInD-Pool/img/WInD-Pool-Business-Case_ENG.pdf) (accessed 17 Feb, 2021)

<sup>7</sup><https://www.businesswire.com/news/home/20161115005705/en/GE-Aviation-Launches-Configuration-Data-Exchange-Reduce> (accessed 17 Feb, 2021)

of maintenance processes, and in the reduction of production costs, leading to savings over a machine's life cycle. However, each of these application cases targets a specific domain and has been adopted by only a handful of participants who exchange their data under multilateral agreements.

### 2.5.1 Existing Standards and Initiatives Limitations

All these standards and initiatives try to address aspects of the challenges presented above. The existence of international standards that rule the data sharing activity may, itself, has a positive impact on challenges 1,2, and 3, since the existing successful experiences are sufficient to influence the business culture and processes and indirectly reduce the human factors risk. They recommend the training of all the personnel involved in the data collection or reporting activities. They also encourage to propose incentives to the personnel as a way to improve the quality of the collected or reported data.

Challenge 4 is explicitly handled by the definition of standard exhaustive taxonomies that describe the equipment components, failure details, and the maintenance procedures of the maintenance key performance indicators (KPIs). This is already possible where the standards are defined for specific industry fields (e.g., oil & gas) with well-known types of equipment. However, this solution cannot be adopted to cover all industries. Consortia agreements are a solution for the legal aspect (challenge 5), as data is contractually agreed to be shared by declining all responsibility relative to its use. Nevertheless, this approach is not scalable and not efficient when targeting a large set of companies.

Finally, the sixth challenge is being addressed by the cited standards by their recommendations to apply data anonymization. This process aims to remove all sensitive data from the maintenance knowledge before it is shared. In practice, the anonymization of maintenance data is usually done manually, as no automated approach has been adopted or promoted by the standards and initiatives.

## 2.6 Maintenance Knowledge Sharing Requirements

Based on the motivating scenario described above, and with respect to the interviews campaign results, we now present the following set of requirements.

The *first* requirement is that the proposed solution should make it easy to capture and share knowledge among various actors.

The *second* requirement is that the proposed solution should be usable both on the cloud (public or private) and on-premise systems. Indeed many customers are willing to connect their machines and factories to the cloud, while others choose to fully isolate their factories in order to protect their industrial property and to keep their private data locally.

The *third* requirement is that the solution should have a built-in mechanism to protect sensitive information about the processes and the business. During our interactions with customers, this requirement came up as the make or break point for them.

The *fourth* requirement is that the solution should support root cause analysis and make it easy to identify the component(s) that cause the failures.

The *fifth* requirement is that the solution should be platform-independent and thus should not depend on any particular hardware or software platform.

The *sixth* and last requirement is that the proposed solution should be open and extensible to cover the current as well as future needs. In fact, the interviewed customers are expecting a wide-opened solution that is opened to third-party equipment.

## 2.7 Industrial Maintenance Knowledge Management & Sharing - State-of-The-Art

In this section, we present the state-of-the-art of maintenance knowledge sharing. We first distinguish the different types of maintenance knowledge, and we highlight the importance of maintenance knowledge management. Then, we present the most commonly used technologies for maintenance knowledge management, and we detail the current practices in industries for maintenance knowledge management based on field investigations. Afterward, we describe the main approaches used to capture and capitalize the maintenance knowledge. Last, we describe the used approaches for the maintenance knowledge sharing in industrial contexts.

### 2.7.1 Importance of knowledge collection and management for industrial maintenance

In the literature, maintenance knowledge is defined as the combination of two complementary types of knowledge: tacit knowledge and explicit knowledge. This classification of knowledge is based on the degree whether knowledge could be easily shared with others [30]. Tacit knowledge refers to the knowledge that is difficult to express or extract. Polanyi stated tacit knowledge, that "we can know more than we can tell" [31]. In contrast, explicit knowledge could be easily expressed by words or documents.

Several studies like [28, 29] agree on the importance of knowledge management in maintenance activity to make the most of explicit and particularly tacit maintenance knowledge. More and more organizations try to use their internal knowledge more efficiently [33], allowing them to collect a great mass of data every day [32].

Nevertheless, the authors of [37] conducted a survey in order to obtain the perception of the maintenance technicians regarding maintenance knowledge management. It was



estimated that only 50% of the maintenance knowledge is actually recorded (explicit) in the company, while the remaining 50% is maintenance knowledge that maintenance technicians have. This significant gap between the tacit and explicit maintenance knowledge can affect the companies, as part of the strategic knowledge is lost when a maintenance technician leaves the company.

The reason for this knowledge retention is not necessarily linked to the tacit knowledge nature, as being difficult to explain, but more to the lack of means and adequate resources to help the maintenance technicians to express easily their tacit knowledge.

Also, it was proven that maintenance knowledge contributes to the safety and reliability of industrial plants [34]. The authors of [35] studied the human maintenance errors in power generation plants, they found that many of human errors factors are directly or indirectly linked to the lack of efficient maintenance knowledge. Also, Faulty procedures, poor training and work practices are usually the result of inefficient or nonexistent maintenance knowledge management strategy. Which generates significant costs and important losses to the manufacturing companies [36].

### 2.7.2 Maintenance Data Management Systems

Maintenance data management became critical with the advent of Industry 4.0, and several software editors propose various solutions to manage maintenance data. These solutions can be classified into three main categories:

1. *Computerized Maintenance Management System or CMMS*: is a software dedicated to tracking and maintaining assets and resources in an organization's maintenance unit. It allows to collect maintenance details and keeps a history of all maintenance operations for every managed asset [15].
2. *Manufacturing Execution System or MES*: is a software that collects in real-time the production data of a factory or a workshop, including quality control, production monitoring, scheduling, and preventive and curative maintenance.
3. *Enterprise Resource Planning or ERP*: is a software package composed of several modules and applications that manage every aspect of an organization's business processes (e.g. financials, manufacturing, inventory, procurement, project management, and so on). Maintenance modules are proposed as extensions and offer usually the same features as a CMMS.

These systems provide features such as predictive and preventive maintenance, maintenance planning, scheduling, execution, monitoring, and traceability. However, these systems

have three inherent drawbacks. First, they are intended to optimize the maintenance process at given location (a factory or a site). This means that two different factories (or sites) cannot share the details of a specific machine's maintenance operations without a human expert in the loop.

The second drawback of these systems is that they are not interoperable at the semantic level [16, 17]. For instance, Schneider Electric works with several Original Equipment Manufacturers (OEMs) who design, build, and ship machines for their customers around the world. In practice, there is no easy way to align the data coming out of these maintenance systems and to thereby get a uniform understanding. This issue is complicated by the heterogeneity of these systems and the associated silos since each business segment and customer is unique and operates under different regulatory and geographic constraints.

The third drawback is that these systems are difficult to use by the maintenance technicians [38]. This was also confirmed during our interview campaign by the testimonies of a maintenance technician and two service bureau managers. In fact, these tools support several advanced features but are far from being intuitive for field operators and technicians.

### 2.7.3 Current practices in industrial maintenance knowledge management

Despite the existence of several tools and resources for efficient management of maintenance knowledge. The current practices in many industries are far to be compliant with the Industry 4.0 spirit.

As an example, the authors of [38] conducted a field study on two Swedish companies, they came up to the conclusion that the maintenance information collection and sharing practices are to date at a pre-industry 4.0 maturity stage. Despite the implementation of digitization capabilities and resources such as ERP systems, word of mouth, papers, and emails remain the main communication channels in the shop floors. The authors highlight the necessity to create a supportive organizational structure and culture and to continue to develop capabilities of information systems in order to enhance knowledge sharing in industry 4.0.

In another example, the authors of [39] conducted a survey about maintenance practices on industrial companies in Manaus in Brazil. They found that most companies have a basic level of maintenance management and they spend much effort and time to attack the consequences instead of focusing on causes. They strongly believe that the main reason for this situation is the poor maintenance knowledge management, and they consider that solutions such as CMMS might be adequate for maintenance knowledge management.

The authors of [40] also found in their investigations that the maintenance knowledge is usually stored in an inconsistent format and that every maintenance operator uses different

methods to describe their experiences and tacit knowledge. Also, the different actors in industries, including managers, supervisors, and operators consider that the lack of knowledge on the plant, equipment, and process is the main limitation for implementing effective maintenance procedures.

#### 2.7.4 Maintenance knowledge capturing and capitalization

Several knowledge-based systems, usually based on expert knowledge, have been developed to support maintenance decisions, with objectives including design of strategies, scheduling of tasks, or diagnosis of machines [30].

Computerized Maintenance Management Systems (CMMS) are generalized in large companies to manage large amounts of information provided by technicians after each maintenance intervention [41]. However, most of these enterprises do not use such data properly so as to transform them into knowledge to be employed in their own activities [42]. Thus various solutions are proposed in the literature to enhance the maintenance knowledge capturing and capitalization.

in [29], the authors propose a framework allowing to manage and generate knowledge from information on past experiences in order to improve the decisions related to the maintenance activity. Historical data is selected from the existing CMMS and structured using conceptual graphs. Then, association rules are defined in order to infer additional knowledge from past experiences. The inferred knowledge is then validated by the domain expert and shared back with the CMMS tools. However, this data mining approach requires the involvement of a human expert in the different phases of knowledge extraction, formalization, and inference, which means that every solution is specific to the context of use and cannot be applicable for large-scale cases.

It is important to note that a large pool of work on industrial maintenance and maintenance knowledge capturing and capitalization use ontologies as a way to structure and formalize the maintenance knowledge. Ontologies are becoming a trend in various domains [43], research communities commonly assume that they are the appropriate modeling structure for representing knowledge [44]. In this thesis, we focus on ontology-based approaches to capture maintenance knowledge because we believe that other approaches will not fit the heterogeneity and diversity of the domains and segments of Schneider customers.

For instance, the authors of [45] created a taxonomy of the Prognostics and Health Management in manufacturing. They propose a formal ontology for failure prognostics based on industrial ISO standards for failure mode analysis, failure diagnostics, and prognostics (e.g., ISO 13372, ISO 13379, ISO 13381, and others). Failure knowledge is described in ontologies from ISO standards. Semantic Web Rule Language (SWRL) [46] is used to define rules in order to generate warning messages in case of abnormal states. However, no approach is

described to share the acquired failure knowledge among different users.

In [47], authors used ontologies for fault diagnosis for industrial control applications. They utilized reasoning capability to check the model consistency over time and to raise early alarms for critical failures.

[48] propose ontology-based support for fault diagnosis for aircraft maintenance operations. An aircraft maintenance ontology is modeled and fed by the (manual) alignment of several existing ontologies related to the avionics domain to discover the relations between the causes and failure symptoms, explain the failures and any unscheduled maintenance requirements as well as the possible procedures that can be applied to each situation.

An ontology-based approach is adopted [49] for the fault diagnostics of conveyors. The knowledge about fault symptoms, fault causes, and fault solutions was modeled with multiple ontologies. The resulting ontologies were mapped together based on a mathematical formulation of conveyor fault diagnostics. Some reasoning rules were defined in order to infer additional relations between faults, symptoms, and potential causes. In [50], R. Chen et al. used ontologies to model the knowledge of fault diagnosis for rotating machines. Their proposed ontology model describes fault diagnosis knowledge considering the vibration characteristics as the main fault factor. The model's reasoning capability is considered by defining some SWRL rules for fault diagnostics.

[51] also relied on ontologies to design a loader fault diagnosis system. It aims to help users find the fault causes, locations, and fault maintenance measures of loaders in a reasonable amount of time. Ontology is used to model the loader information and describe the relative failures. This work uses the Condition Based Reasoning (CBR) method to diagnose loader faults by finding similar corresponding situations in the past. When no corresponding case is found, CBR fails and the (SWRL-based) Rule-Based Reasoning (RBR) approach is proposed for fault diagnosis.

All the works mentioned above use ontologies to model machine data models and failure knowledge. However, none of these works satisfy all of the requirements that we identified from our motivating scenario. These works developed various ontology models and some exhaustively described potential failures and their characteristics. However, to the best of our knowledge, no machine failure ontology is available for reuse or for extension. Furthermore, neither of our two major requirements, i.e., knowledge sharing and data confidentiality have been considered.

### 2.7.5 Maintenance knowledge sharing

efficient Knowledge communication between the different manufacturing actors is a key element for successful maintenance [52]. Maintenance knowledge collection and modeling and capitalization have been intensively investigated in the literature. However, very few

papers explored the maintenance knowledge sharing field.

In fact, most of the works either do not consider the maintenance knowledge sharing aspect such as [53], or admit that the existing CMMSs, or other maintenance information systems, are sufficient for that purpose. As an example, the authors of [42] proposed a conceptual framework for sustainable performance in maintenance, they adopted a three-phases approach, consisting of 1) structuring the maintenance knowledge through knowledge graphs, 2) creation of rules to capitalize the structured knowledge, and then share the collected knowledge using CMSS-like tools. similar approaches are adopted in other works like [29].

On the other hand, the authors of [28] studied the communication practices in the industry and defined a model for gathering and sharing knowledge in maintenance work. Their model considers the maintenance technician as the central entity in the knowledge collection and sharing flows. Thus, in addition to adequate software tools and resources, the training and encouragement of maintenance operators are success keys for efficient knowledge collection and sharing.

Based on the principle: "the transfer of know-how requires a process of show-how" [54], various works propose mainly three technologies to support communication and maintenance knowledge sharing: 1) augmented reality guidance [55–57], 2) remote assistance [59–61], 3) data collection and reporting with the wearable devices [62,63]. However, despite the efficiency of such technologies, they remain, for the moment, expensive and too much sophisticated to be adopted in all industries especially the small and medium ones.

In [64] and [65], S. Wan et al. developed a Collaborative Maintenance System Planning that allows many stakeholders to collaborate to ensure maintenance process quality. An ontology-based approach is adopted to model a large field of knowledge: the machine domain model, failure knowledge, and stakeholders knowledge are modeled together to ensure the interoperability between their systems, the maintenance planning, and Resources and Constraints knowledge. However, this centralized solution focuses mainly on preventive maintenance planning. In addition, the managed failure knowledge is relatively basic and does not consider root causes or symptoms.

## 2.8 Lessons learned

Conducting this study helped us to learn several lessons. The first lesson is that there are several cultural and technical challenges to sharing the maintenance knowledge. Much of this has to do with the prevailing practices and the human nature to solve the issue at hand first with our own ideas than to find what has been done previously by others.

In terms of technical challenges, the use of semantic web technologies to solve complex

industrial problems seems promising, however, it is still not adopted in mature maintenance knowledge management solutions. Capturing the maintenance knowledge within a given context (domain, machine, segment) is not trivial. The tools have to be user-friendly as well as simple to use.

Furthermore, capitalization or reuse of maintenance knowledge seems exclusive to a few large groups, while most companies handle machines failures, as they come, without benefiting from previous experiences. Case-Based Reasoning (CBR) turns out to be an efficient approach for maintenance knowledge reuse as it aims, by definition, to solve new problems based on the solutions of similar past problems.

Maintenance knowledge sharing reveals to be the adequate solution to enhance the maintenance activity however, we felt that it will take time to be used on a large scale. Adopting the right culture and using the right incentives can speed up the adoption of maintenance knowledge sharing.

## 2.9 Summary

This chapter provides a comprehensive background about maintenance knowledge collection and sharing in industrial companies. The results of the interviews campaign with Schneider customers and domain experts were detailed showing the various challenges impeding the development of maintenance knowledge sharing. Existing standards and initiatives of maintenance knowledge sharing have been detailed and critically reviewed. And the existing state-of-the-art works about the different aspects of maintenance knowledge management were reviewed.



# Industrial Maintenance Knowledge Sharing With Semantic Knowledge Graphs

## Contents

---

<b>3.1</b>	<b>Introduction</b>	<b>48</b>
<b>3.2</b>	<b>Ontology data models for maintenance knowledge capturing</b>	<b>48</b>
3.2.1	Machine Failure Ontology Model	49
3.2.2	Machine Domain Ontology Model	50
3.2.3	Upper-ontologies adoption	56
<b>3.3</b>	<b>Proposed Architecture</b>	<b>57</b>
3.3.1	High-Level Architecture	57
3.3.2	Detailed Architecture	58
<b>3.4</b>	<b>SemKoRe Workflow &amp; Process</b>	<b>59</b>
3.4.1	Case-Based Reasoning formalization	61
<b>3.5</b>	<b>Proof-of-Concept Prototype</b>	<b>63</b>
3.5.1	Implemented services	63
3.5.2	Failure Data Aggregation	65
3.5.3	Implementation Details	66
3.5.4	Evaluation setup	68
3.5.5	Performance measurements	68
<b>3.6</b>	<b>Learned Lessons</b>	<b>70</b>
<b>3.7</b>	<b>Summary</b>	<b>71</b>

---



### 3.1 Introduction

This chapter describes the design and proof-of-concept prototype of the SemKoRe, our proposed solution for maintenance knowledge sharing. It discusses the architectural choices and the ontology design to capture efficiently the different aspects of maintenance operations, along with early performance measurements. It is based on Paper I and addresses the following questions:

- *Q2- How maintenance knowledge can be shared between different stakeholders? What will be an efficient architectural solution to accomplish this? Especially, how to manage the maintenance knowledge collection and sharing on the three IoT layers: Edge, Fog, and Cloud since not everyone is connected to Cloud? What features or services should be part of the proposed solution?*
- *Q3- How to design an open and extensible solution that is applicable to multiple domains and works across silos? Particularly, how to capture and share the maintenance knowledge independent of any domain constraints?*

In the following, we describe the ontology data model design along with the key questions used to gather the various maintenance tasks details. In order to share the collected knowledge between different industry actors around the world, only one choice is available; using the cloud as a central bridge to transfer knowledge from one actor to the others. However, the interviews campaign showed that industrial companies adopt different policies in terms of connectivity to the cloud. So, we proposed a flexible architecture in order to fit the various needs and configurations of Schneider's customers. Collecting and sharing the maintenance knowledge are the first steps of our journey. The final step is the maintenance knowledge reuse and capitalization. So, we adopted a Case-Based Reasoning (CBR) [66] approach for efficient reuse of maintenance knowledge.

We implemented a proof-of-concept to show the validity of our approach for maintenance knowledge sharing. The technical details and the implemented services are briefly described along with performance evaluation.

### 3.2 Ontology data models for maintenance knowledge capturing

The first task of our approach is to capture information about the different maintenance experience aspects, and especially the machine failures characteristics. In fact, we consider machine failure as a central concept in our data model, and we need to describe the failure symptoms, impact, root causes, in addition to the repairing or prevention procedures. We

need also to identify the machine components that are impacted by the failure or that need to be repaired or changed.

For this purpose, we defined our data model using Semantic Web standards because of the schema-less nature of RDF (Resource Description Framework), RDFS (RDF Schema), OWL (Web Ontology Language), and the explicit formalism supported by these languages. This choice allowed us to propose a generic and extensible data model that can fit different use cases and situations.

We adopted the Seven-step method, developed by the Medical Information Center of Stanford University [69] for the development of our ontology models. Also, for a sake of genericity, we divided the data model into two parts: a machine domain ontology and a machine failure ontology.

### 3.2.1 Machine Failure Ontology Model

To capture the different details of a machine failure, the designed data model should be able to answer the following competency questions:

- What are the failure symptoms? Symptoms reflect the perceptible aspects of failures whether they are visual, sonic, odor, or heat-related.
- What is the impact of the failure? This may or may not be detected easily. Each failure impact is relative to a machine or to one of its components.
- What are the root causes of a particular failure? This question is difficult to answer since it assumes prior knowledge about the cause-effect relations specific to each machine type. Answering this question requires the knowledge of a machine domain expert.
- After knowing the failure type, how can we repair the machine?
- After knowing the root causes of a specific failure, is there a preventive maintenance procedure that can help us to avoid that failure?

To answer all of these questions, we adopted the following steps in regards to the Seven-steps method for the development of the machine failure ontology:

1. **Determine domain and scope:** This work focuses on industrial machine failures.
2. **Consider reusing existing ontologies:** No existing machine domain or machine failure ontology was found for reuse, therefore we developed both for this work. Regarding upper-level ontologies, the discussion in the section 3.2.3 clarifies our point of view.

3. **List important terms in the ontology:** After interactions with the machine domain experts, the following important terms were identified Failure, Symptom, Impact, Root cause, Solving Procedure, among others.
4. **Define classes and class hierarchy:** Several classes were created including the important terms listed above. However, since no specialization concept will be introduced, the ontology is flat. Only the classes relative to types (e.g., Failure Type, Symptom Type) are grouped as sub-classes of the Types class.
5. **Define object properties of classes:** We defined a set of object properties that link all the defined classes together. For example, the property *hasSymptoms* links a Symptom to a specific Failure. The complete list is illustrated in Figure 3.2b
6. **Define data properties of classes:** We also defined several of the data properties of classes as shown in Figure 3.2c.
7. **Create instances and check exceptions:** Instances of SemKoRe ontology are divided into two parts. The first part, defined by experts during the design time, concerns the generic concepts that will be used for most industrial use cases, such as Severity level (Catastrophic, Critical, Moderate, Low). The second type of instance concerns the data provided by the users during the runtime of the SemKoRe. Users' instances include details about the failures and symptoms. To check for exceptions, we used the Pellet reasoner [68] to verify the correctness of our ontology model.

The defined machine failure ontology is presented in Figure 3.1. The list of defined classes, object properties, and data properties are shown respectively in Table 3.1, Table 3.2 and Table 3.3. This ontology model is created by interacting with the MSol LoB Team and by using the initial set of requirements described in Section 2.6. It acts as a common data model and will be enriched with new concepts by domain experts over time. Progressively, Schneider-Electric's design, engineering, configuration, and maintenance tools will use this ontology to create knowledge about the failures and allow to develop different services over it.

### 3.2.2 Machine Domain Ontology Model

For each recorded failure, we need to identify the machine components that are impacted by the failure or that need to be repaired or changed. So, we require the definition of a machine ontology that describes the machine or equipment, its components with their hierarchical structure. As such ontology is unique for each machine type, in our approach we consider that it should be defined by the machine builders. This ontology must describe all the

Table 3.1: Failure ontology classes description

Class Name	Super Class	Description	Example of Instances
Failure		This class allows describing machine failures with all their characteristics such as symptoms, root causes, solution or avoidance procedures, etc.	Inappropriate sealing quality, Defective manufactured products
Failure Occurrence	Failure	This class describes an event of a failure occurrence with the observed characteristics such as symptoms, impacts, root causes, etc.	Same as Failure
Failure Severity Level		Defines a severity level for the failure	Medium, Critical, Fatal
Failure Asset		This class is used to link a failure to an asset such as a machine component, machine part, or even the machine itself. Each asset has a "Function ID" attribute that allows differentiating similar assets in the same machine.	Robot, Servo drive, Component XYZ
Failure Symptom		Describes the symptom observed on an asset after a failure occurrence	Fire smoke, noisy servo drive, unstable belt
Failure Impact		Describes the Impact of a machine failure on a specific asset	Robot hand damaged, uncalibrated component
Failure Root Cause		Describes the assets that probably cause a machine failure	High servo drive's rotation speed, Incorrect configuration
Failure Analytics		Allows to reference data analytics services capable of automatically detecting one or more machine failures types	Schneider PrediAxe for servo drives failures detection
Failure Analytics Detection		Describes an automatic machine failure detection notified by a Failure Analytics service	Servo drive failure detected on 12/11/2021 at 18:53
Failure Analytics Model		This class allows referencing versioned ML models that are used by the Failure analytics services for automatic detection of machine failures	Model XYZ v1.0.0
Failure Mitigation Procedure		Defines an ordered list of maintenance actions that can be applied on a machine asset,	
Failure Avoidance Procedure	Failure Mitigation Procedure	Defines a mitigation procedure that can be used to avoid the occurrence of a specific machine failure	
Failure Solution Procedure	Failure Mitigation Procedure	Defines a mitigation procedure that can be used to repair the machine after a specific failure occurrence	
Failure Mitigation Step		Defines a maintenance action that is part of a mitigation procedure	Change the servo drive, Calibrate the component XYZ
Failure Occurrence Status		Allows to specify the status of a machine failure occurrence	
Types		This class encompasses the "Types" classes of the ontology that allow defining the types of the various machine failure characteristics/attributes. Categorizing instances using the "Types" classes instead of simply using "rdf:class" is voluntary, because the "Types" are provided by the SemKoRe users and are not standardized concepts. Thus, the "types" are managed as individuals and not classes.	
Failure Analytics Type	Types	Defines the type of a failure analytics service	Failure detection analytics, Failure prediction analytics
Failure Impact Type	Types	Defines the impact's type of a failure	Damaged part, Out of service, Calibration problem
Failure Mitigation Action Type	Types	Defines the type of a mitigation action	Component Repair/change, Configuration change,
Failure Root Cause Type	Types	Defines the type of a failure root cause	Asset, Chemical phenomenon, Configuration, Failure Cascade
Failure Symptom Type	Types	Defines the type of a failure symptom	Abnormal Mechanical Behaviors, Abnormal smell, Abnormal noise, Component stopped working
Failure Type	Types	Defines the type of a failure	Electrical Failure, Erroneous manipulation, Mechanical Failure, Control failure
Failure Update Type	Types	Defines the type of a failure update	Creation, Modification, Validation, Closed

Table 3.2: Failure ontology object properties description

Object Property	Parent	Domain	Range	Description
Analytics Model		Failure	Failure Analytics	Defines a Failure Analytics Service as suitable to detect a specific Failure
Avoidance Procedure		Failure	Failure Solution Procedure	Links a mitigation procedure as an avoidance procedure for a specific failure
Failure Type	hasType	Failure	Failure Type	Specifies the type of a machine failure
Has Impact		Failure	Failure Impact	Defines the impacts of a specific failure
Root Causes		Failure	Failure Root Cause	Defines the possible root causes of a specific failure
Severity		Failure	Failure Severity Level	Defines a severity level for a failure
Solving Procedure		Failure	Failure Solution Procedure	Specifies a solving procedure for the failure
Symptoms		Failure	Failure Symptom	Defines the symptoms of a failure
has Failure Asset		Failure	Failure Asset	Specifies the machine on which the failure has occurred
Analytics Type	hasType	Failure Analytics	Failure Analytics Type	Defines the type of the Analytics service, e.g.: Predictive, Detection.
Analytics Asset	hasAsset	Failure Analytics	Failure Asset	Associates the analytics service to a specific asset
Detected Failure		Failure Analytics Detection	Failure	Describes the failure detected by an analytics service
detectedByModel		Failure Analytics Detection	Failure Analytics Model	Specifies the model used for a failure detection
Real Failure		Failure Analytics Detection	Failure	Specifies the real failure that has occurred, it is provided by a human user
Asset Component		Failure Asset	MachineModel: Component	References a component from the Machine ontology as an Asset for the failure ontology
Machine Type		Failure Asset	MachineModel: Machine	Specifies the machine type of a given asset
Impacted Asset	hasAsset	Failure Impact	Failure Asset	Specifies the asset impacted by the failure
Impact Type	hasType	Failure Impact	Failure Impact Type	Specifies the type of a failure impact
Mitigation type	hasType	Failure Mitigation Procedure	Failure Mitigation Type	Specifies the type of a mitigation procedure
Mitigation Action Steps		Failure Mitigation Procedure	Failure Mitigation Step	Specifies the type of a mitigation action step
Action Asset	hasAsset	Failure Mitigation Step	Failure Asset	Specifies the asset on which the mitigation action will be applied
Action Type	hasType	Failure Mitigation Step	Failure Mitigation Action Type	Specifies the type of a mitigation action
Failure Detected By Analytics		Failure Occurrence	Failure Analytics Detection	Shows the analytics detection service that automatically detected the failure
Status		Failure Occurrence	Failure Occurrence	Defines the status of the failure occurrence. The failure occurrence remains modifiable until it is validated and closed
Update Operation		Failure Occurrence	Failure Update Operation	Specifies the updates that have been made on a failure occurrence
Root Cause Asset	hasAsset	Failure Root Cause	Failure Asset	Defines the asset involved as a root cause of a failure
Root Cause Type	hasType	Failure Root Cause	Failure Root Cause Type	Defines the type of a failure root cause
Symptom Asset	hasAsset	Failure Symptom	Failure Asset	Defines the asset involved as a symptom of a failure
Symptom Type	hasType	Failure Symptom	Failure Symptom Type	Defines the type of a failure symptom
Operator		Failure Update Operation	owl:Thing	Keeps track of the User that made updates on a failure occurrence
Update Type		Failure Update Operation	Failure Update Type	Defines the type of update done on a failure occurrence
hasAsset		Owl:Thing	Failure Asset	Used to as parent class of the hasAsset properties
hasType		Owl:Thing	Types	Used to as parent class of the hasType properties

Table 3.3: Failure ontology data properties description

Data Property	Domain	Range	Description
Description	owl:Thing	xsd:string	This property is used to provide a description for individuals
Algorithm Name	Failure Analytics	xsd:string	Specifies the name of the algorithms used by a Failure Analytics service
Data Sources	Failure Analytics	xsd:string	Specifies the data sources that are analyzed by an analytics service for automatic failure detection
Detection Time	Failure Analytics Detection	xsd:dateTime	Specifies the time of automatic detection of a failure
Analytics Data Chunk	Failure Analytics Detection	xsd:string	Specifies the analyzed data chunk that allowed to detect a failure
Is False Positive	Failure Analytics Detection	xsd:boolean	Provided by the user, it specifies if the automatic detection was correct or just a false positive
Analytics Model File	Failure Analytics Model	xsd:string	Defines the model file's name used for automatic failure detection
Analytics Model Format	Failure Analytics Model	xsd:string	Defines the format of the analytics model file
Analytics Model Version	Failure Analytics Model	xsd:string	Defines the version of the analytics model file
Analytics Training Data	Failure Analytics Model	xsd:string	Defines the data that was used to train/build the analytics model file
Generation Time	Failure Analytics Model	xsd:dateTime	Specifies the time needed to (automatically) generate the analytics model file
Asset Function ID	Failure Asset	xsd:string	Defines a unique function ID for each machine asset
Order Number	Failure Mitigation Step	xsd:int	Defines the order number of an action in a mitigation procedure
Occurrence Time	Failure Occurrence	xsd:dateTime	Specifies the time of failure occurrence
Severity Level	Failure Severity Level	xsd:int	Specifies the degree of criticality of a failure
State of the Asset	Failure Symptom	xsd:string	Specifies the state of an asset being identified as a failure symptom
Update Time	Failure Update Operation	xsd:dateTime	Specifies the time of update made by a user



Figure 3.1: Failure ontology.

vocabulary relative to a machine, and thus, can be used as a common machine taxonomy by all the owners of the same machine type.

This divide-and-conquer approach allows handling the fourth challenge identified in section 2.4.3. In fact, instead of trying to propose an exhaustive taxonomy for all types of equipment, machines, or failures for a specific industry field, supporting the definition of individual taxonomies for each machine type turns out to be more scalable.

Figure 3.2a shows an example of machine ontology that we created to validate our approach. Once a machine ontology model is defined, it must be integrated into the failure ontology. For this purpose, we created the OWL Class *FailureAsset*, to associate failures to the corresponding components in the machine domain ontology.

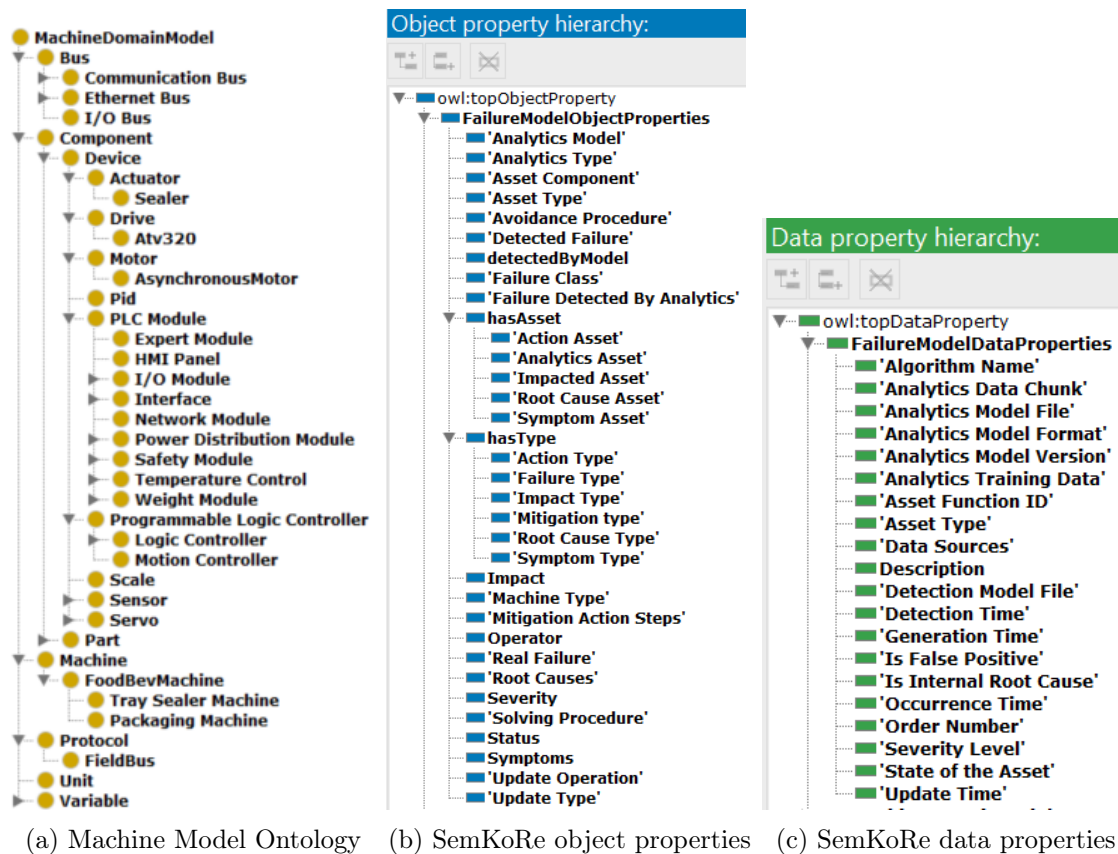


Figure 3.2: SemKoRe ontology design.

However, an issue can be encountered when trying to identify the exact machine component that caused or that is impacted by the failure. For example, consider a machine composed of two Servo Drives of the same type. These Servo Drives are described in the machine domain ontology as two instances (SDA and SDB) of the "ServoDrive" class and



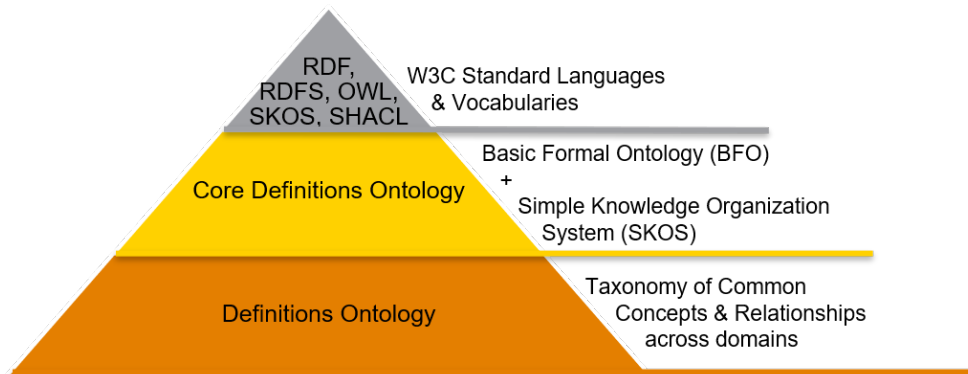


Figure 3.3: Ontology modeling approach

are associated with the instance of the machine. When a failure occurs in SDA, we should be able to identify it through ontology. Such detailed identification is especially useful when the failure knowledge is shared with the other sites using the same machine type, as it will help them to recognize the exact component responsible or impacted by the failure.

To address this issue, we require that each component in the machine domain ontology has a unique number "FunctionID", used to distinguish its role in the machine compared to other components of the same type.

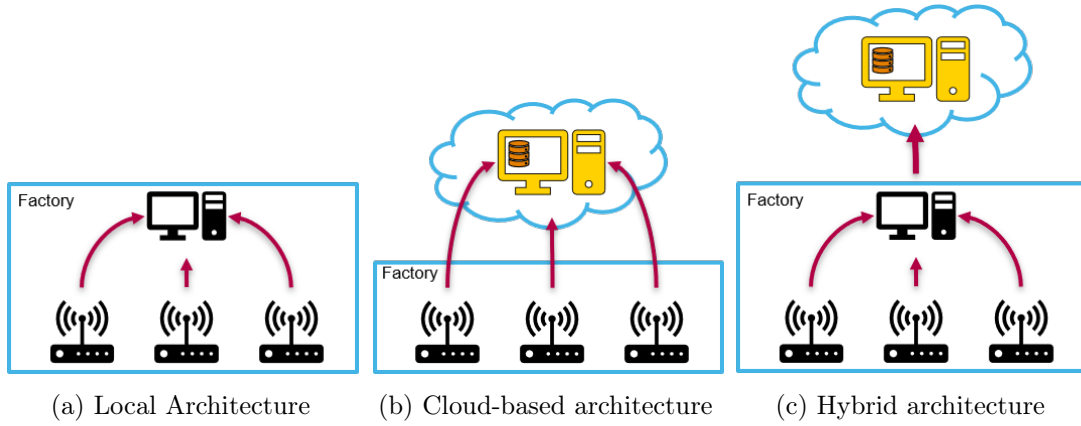
### 3.2.3 Upper-ontologies adoption

It is important to mention that our main focus in this work has been on the validation of the idea to the OEMs, that formalized knowledge about machines and their failures can be useful for quick resolution of failures and to improve Overall Equipment Effectiveness (OEE). Hence, the defined ontologies, in their current PoC form, do not rely on any upper-level ontology. The main reason is that there was a separate work in Schneider-Electric to decide on the right ontology to provide maximum data integration for the future. For example, the discussion revolved around using BFO <sup>1</sup> or SKOS <sup>2</sup> as upper-level ontology, as shown in the pyramid in figure 3.3. Some customers may also be interested in using domain ontologies like SSN <sup>3</sup>. Our view on this point is that once a decision is made, our current ontologies can be easily refactored.

<sup>1</sup><https://basic-formal-ontology.org/>

<sup>2</sup><https://www.w3.org/TR/2008/WD-skos-reference-20080829/skos.html>

<sup>3</sup><https://www.w3.org/2005/Incubator/ssn/ssnx/ssn>





- : IoT gateway connected to a real machine, : Local server.

Figure 3.4: SemKoRe's flexible architecture's configurations

### 3.3 Proposed Architecture

#### 3.3.1 High-Level Architecture

As mentioned before, our customers require different deployment options, so we divided them into three categories and developed a flexible architecture that fits all three categories. *In the first category*, customers prefer to not connect their machines to the cloud and some even do not want to connect to the Internet due to the sensitive nature of their business and to protect their data. The architecture proposed for this category is shown in Figure 3.4a. *The second category* of customers opted for an entirely connected architecture, in which the machines/gateways in their factories are directly connected to the cloud. For this category, we proposed the architecture shown in Figure 3.4b. *The third architecture* targets the customers who refuse to connect their machines to the cloud but are ready to deploy (or already have) a local on-premise server between the cloud and their machines. For this use case, we proposed a hybrid architecture Figure 3.4c, where most of the collected data stays in the local server, and only an anonymized part of the data is transmitted to the cloud.

In all these cases, each machine is connected to an industrial IoT gateway to collect the run-time data of the machine and the information provided by maintenance technicians or machine operators. Each gateway is connected to a central entity (either a local server or cloud server) which collects the data from all the gateways and then aggregates and shares with the gateways connected to the same type of machine.

In this thesis, we focus on cloud-based architecture because it covers all the constraints and features of the other architecture configurations. This architecture is also implemented

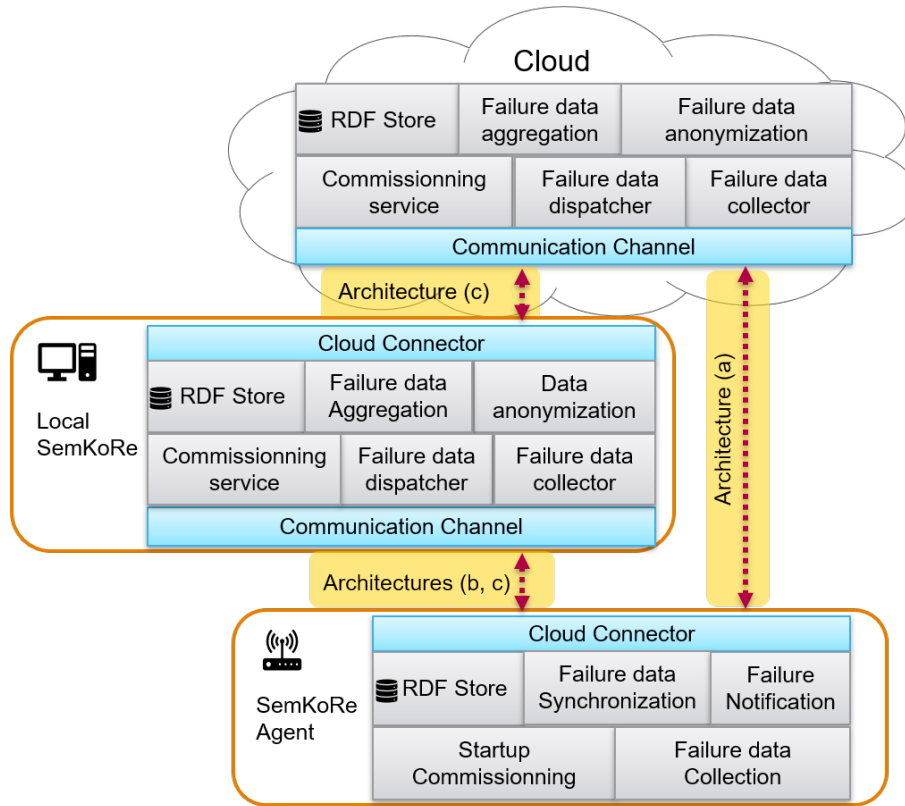


Figure 3.5: SemKoRe detailed architecture.

in a proof-of-concept to demonstrate the feasibility (see Section 3.5).

### 3.3.2 Detailed Architecture

Figure 3.5 shows the detailed architecture of the SemKoRe. The SemKoRe consists of three entities:

1. A SemKoRe Agent: Runs on industrial IoT gateways connected to the machines. It collects data when failures occur in the connected machines. According to the chosen architecture, the collected data are then shared with either the SemKoRe Server or Local SemKoRe. The SemKoRe Agent is designed to fit in all architectures (Fig. 3.4).
2. A SemKoRe Server: Running on the cloud, it manages several failure data producers, i.e., Local SemKoRes or SemKoRe Agents. The collected failure data are validated by an expert and aggregated and afterward shared with the SemKoRe Agents and/or Local SemKoRes.
3. Local SemKoRe: Lightweight instance of the SemKoRe Server deployed on a local

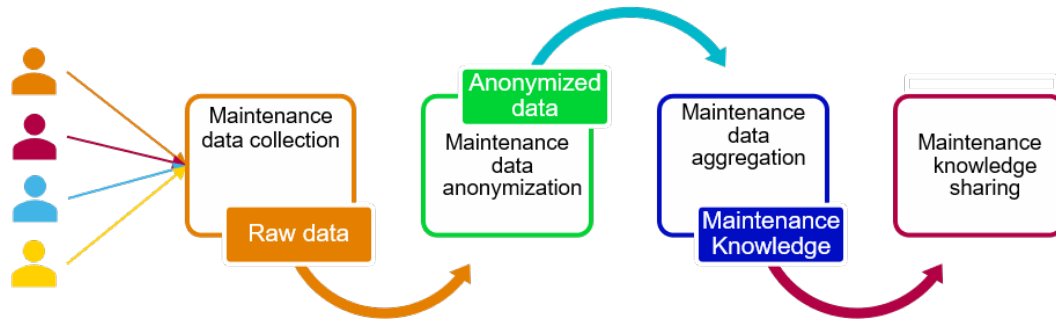


Figure 3.6: SemKoRe Workflow.

server to manage the machines located in a site or factory. It collects data produced by the SemKoRe Server and the SemKoRe Agents on local gateways. The data aggregation is done locally, and only aggregated data is shared with the SemKoRe Server. The cloud then merges its aggregations with the Local SemKoRe aggregation and pushes back the updates to the corresponding entities.

### 3.4 SemKoRe Workflow & Process

SemKoRe is provided with many services (Figure 3.5) running on the different IoT layers. These services collaborate in order to achieve various SemKoRe purposes such as data collection, data aggregation, data anonymization, and data sharing. Figure 3.6 shows the main SemKoRe workflow and how maintenance data is transformed from row data to maintenance knowledge. In fact, we define the maintenance knowledge to be the result of aggregation and anonymization applied to the maintenance row data.

Figure 3.7 shows the SemKoRe process of failure data collection and sharing. The process is distributed on two layers: on the edge with the SemKoRe Agent, and on the cloud with the SemKoRe Server.

The failure data collection starts when a machine failure occurs. The failure information collection service generates the Human Machine Interface (HMI) for the user (Bob or Alice) to offer the details of the failure. Through the survey, we first try to know if the failure has really occurred or it was only a false positive case triggered by some failure detection service. Then the user is asked to provide details about the symptoms of the failure by selecting known symptoms or by creating new ones, when necessary.

The user checks if the identified failure is already known by the SemKoRe before providing additional details. If the failure already exists, the user follows the instructions to repair the machine. Otherwise, the failure will be documented by Alice or by machine domain experts as shown in Figure 3.7. This process is a typical application of Case-Based

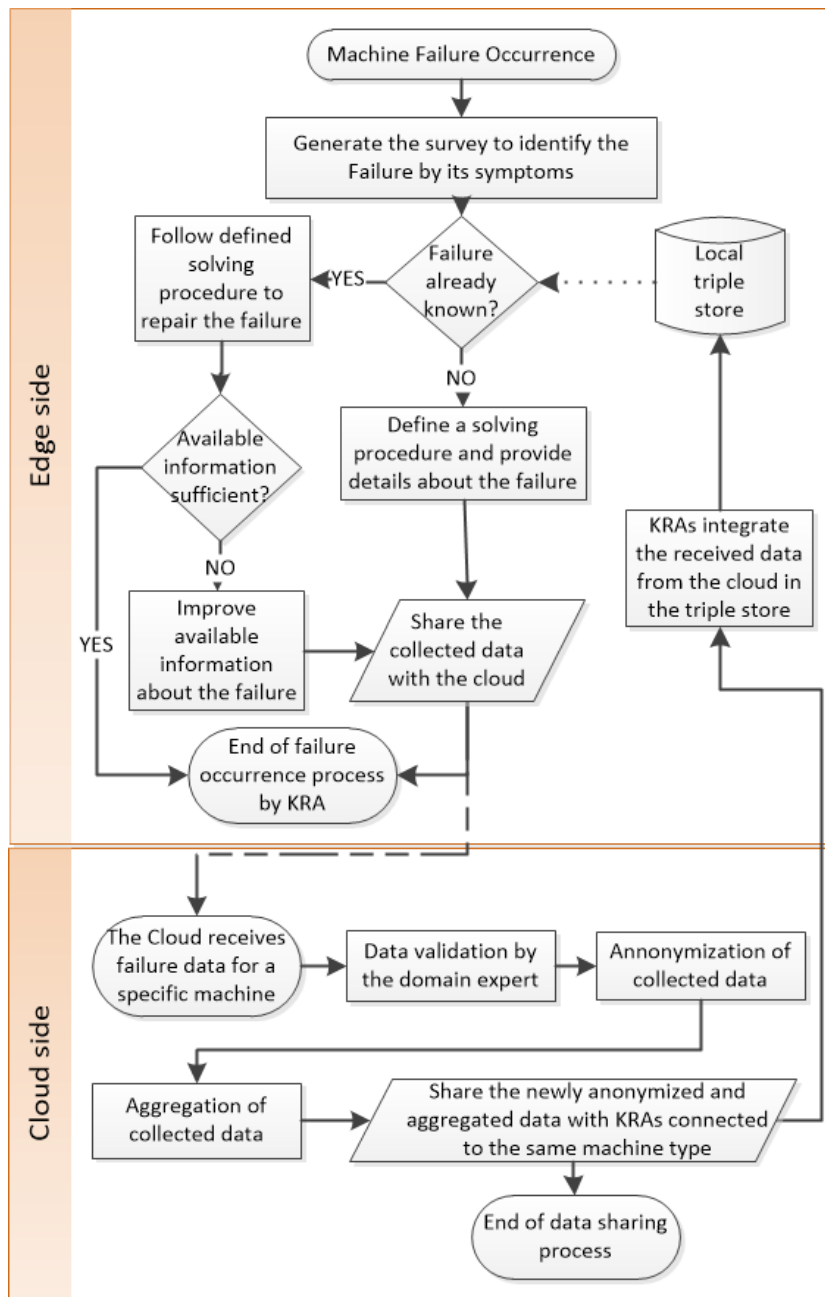


Figure 3.7: SemKoRe process.

Reasoning (CBR) [66], which consists of solving new problems based on the solutions of similar past problems. Sometimes, the impacts of a failure may differ from one machine to another. So, existing repair procedures might be adapted or new procedures need to be created to repair the machine. This process ends, at the edge level, by sharing the collected

data with the SemKoRe Server in the cloud.

When the SemKoRe Server receives failure data from a SemKoRe Agent instance, the data must be validated by a machine domain expert before it is integrated into the SemKoRe Knowledge Graph. After the validation process, data is anonymized to protect the data and customer privacy and business' sensitive information of the customers. The anonymized data is then aggregated in order to get insights about the occurrence frequency of failures, their impacts and the most adapted/used repairing procedures.

Finally, the aggregated data is shared with the SemKoRe Agents instances connected to the same machine type. On the other side, each SemKoRe Agent instance integrates the data it receives from the cloud into the local triplestores (Graph databases).

### 3.4.1 Case-Based Reasoning formalization

Case-Based Reasoning is a method that helps to solve new problems by reusing the knowledge and solutions of similar past problems. The foundation principle of CBR is the fact that "Similar problems have similar solutions" [67]. The figure 3.8 shows the different CBR steps followed for machine failure solving: Case Description, Retrieve, Reuse, Revise and Retain. Each of these steps is described below.

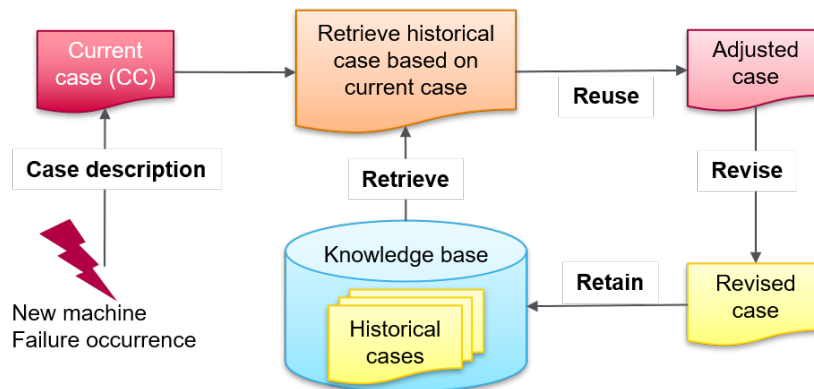


Figure 3.8: CBR steps for machine failure solving

#### 3.4.1.1 Case description

The formal description of each case consists of using basically several distinguishing parameters that describe the problems and that allow identifying similar cases based on the similarity of parameters. In our approach, we consider the machine failures as cases, and we use the failure symptoms as distinguishing parameters to describe the cases. For each

failure, we define the symptoms vector:

$$S = \{S_1, \dots, S_n\}$$

where  $n$  is the number of known symptoms in our knowledge graph, and for  $i \in \{1, \dots, n\}$ ,  $S_i = 1$  if the  $i^{th}$  symptom is observed for that failure, and  $S_i = 0$  otherwise. We define also the symptoms frequency vector, that is created from the aggregated values of the failure data:

$$FS = \{FS_1, \dots, FS_n\}$$

where for  $i \in \{1, \dots, n\}$ ,  $FS_i =$  if the frequency of observation of the  $i^{th}$  symptom for the considered failure.

Thus, we define each case as a combination of both vectors as follows:

$$Case = \{S, FS\}$$

Finally, we assume that we already collected details about  $N$  failures in our KG, thus, the historical cases are defined as:

$$HC = \{HC_1, \dots, HC_N\}$$

### 3.4.1.2 Case retrieval

When a machine failure occurs, the maintenance technician starts by identifying the failure symptoms. Thus, the current case  $CC$  is defined as:

$$CC = \{Sc_1, \dots, Sc_n\}$$

where  $Sc_i$  are (1 or 0) values reflecting the observed symptoms for the current failure.

Now, to identify the similar previous situations, we evaluate the similarity between the current case ( $CC$ ) vector and each of the historical cases ( $HC_i$ ) as follows:

$$Similarity(CC, HC_i) = abs(CC - HC_i^S) \times HC_i^{FS}$$

where  $abs$  is the absolute value and  $HC_i^S$  and  $HC_i^{FS}$  are respectively the symptoms and the frequencies vectors of the historical case ( $HC_i$ ). The multiplication with  $(HC_i^{FS})$  instead of simply using the standard distance evaluation ( $|CC - HC_i^S|$ ) is used to weight the symptoms by giving more importance to the most frequent ones.

Finally, the retrieved historical cases (failures) are the ones showing the minimal *Similarity* values with the current case, and we ask the maintenance operator to select the correct failure ( $HC_i$ ). If none of the historical cases match, the maintenance technician creates a new Failure entry (new case), that will be documented and saved in the KG.

### 3.4.1.3 Case reuse

As shown in the section 3.5.2, every aggregated failure has a list of solving procedures associated with frequencies of successful use. In our approach, we let the maintenance technician choose the correct maintenance procedure to be applied for his case.

### 3.4.1.4 Case revise

Sometimes, it happens that an existing procedure is not completely fitting to the current problem and needs to be adapted to the situation. Thus, we propose to the maintenance technician to adjust the proposed procedure or to create a new procedure in case of major changes.

### 3.4.1.5 Case retain

Finally, the current maintenance operation is stored (retained) in the KG either as a new case for newly discovered failures or to adjust the existing (cases) failures and update the aggregated frequencies of the failure symptoms and the maintenance procedures.

## 3.5 Proof-of-Concept Prototype

In this section, we describe the implementation details for the proof-of-concept prototype. As stated previously, we only focus on the architecture where gateways are directly connected to the cloud (see Figure 3.4).

### 3.5.1 Implemented services

The SemKoRe approach requires several services running on the different IoT layers, Edge, Fog, and Cloud. Hereafter, we describe the implemented services for our prototype.

#### 3.5.1.1 Startup Commissioning

Knowing that our Failure ontology and the collected knowledge will evolve, this service allows the SemKoRe agents to get the latest failure knowledge corresponding to the type of the connected machine. It is only deployed on the SemKoRe Server. On-premise, the startup commissioning service retrieves two types of information from the SemKoRe Server:

- The machine failure T-Box, containing the concepts defined in the failure ontology
- A-Box data, containing the instances of the T-Box concepts. Knowing that the SemKoRe Server manages data relative to several types of machines, the A-Box re-



trieved by a gateway contains only the information relative to the machine to which it is connected.

The startup commissioning service sends a request to the SemKoRe Server with the identity and the type of the connected machines. The SemKoRe Server runs then a Construct SPARQL query to create a sub-graph containing all the data (T-Box and the A-Box) related to the provided machine type. The resulting sub-graph is sent back to the gateway. Listing 3.1 shows a simplified version of the construct query executed on the SemKoRe server side.

```

PREFIX ns: <http://www.schneider-electric.com/KnowledgeRepository#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
construct { ?s ?p ?o}
where {
    ?failure rdf:type ns:Failure.
    ?failure ns:hasAsset ?asset.
    BIND(<Machine_URI> as ?machine) .
    ?asset ns:MachineType ?machine .

    ?failure (ns:|!ns:)* ?s .
    ?s ?p ?o .
}

```

Listing 3.1: Simplified startup commissioning query using <Machine\_URI> as parameter

### 3.5.1.2 Failure Data Collection

This service runs exclusively in the SemKoRe Agent and is used during or after the maintenance phase. It proposes intuitive and easy-to-use interfaces to collect maintenance data and store them in the local triple store. The thesis author has implemented the back-end part of this service, while the UI design team implemented the front-end interfaces following Schneider's unified UX design rules. Two types of UIs are offered by SemKoRe. The first type is a survey interface guiding the user through a set of predefined questions to simplify the reporting of new failure occurrences. The second type of interface, called model-driven UIs, is generated automatically from the ontology data model and allows the addition of extra details to the reported failures occurrences. All these UIs rely on the annotations defined in the ontology such as *@rdfs:label* and *@rdfs:comment*. The former is used as a human-readable label of the input fields shown to the user, while the latter is displayed to explain the nature of the field and the expected input. We defined additional annotations such as *@semkore:hidden* to hide a field on the UI in case it should be defined automatically or exclusively by an expert. On the back-end side, we mainly use the API described in Chapter 6 for all the interactions of the UI with the SemKoRe Knowledge Graph.

### 3.5.1.3 Anonymization Service

Privacy protection is a very important concern for Schneider's customers. They do not want to share any machine and process-related data in a way that could potentially expose sensitive business information. To address this concern, we implemented a sensitive data detection service (details in Chapter 4), that helps domain experts avoid sensitive data disclosure during maintenance knowledge sharing.

When a failure occurs, the gateway creates an instance of "Failure Occurrence Class", containing information about the failure, e.g., symptoms, impact, root causes if known, and the failure context, which includes the machine ID, its location, and the failure occurrence timestamp. The whole process consists of three steps:

1. The anonymization service removes the structured data considered, by default, as sensitive such as machine ID, location, and owner-related information. It also analyzes automatically all textual details in order to detect potentially sensitive data.
2. A human expert reviews and validates all of the failure information before integrating it into the SemKoRe Knowledge Graph.
3. Finally, all the validated failure information is aggregated before being shared with the connected gateways. The aggregation increases the abstraction level on the data and reduces the risk of deducing the origin of the data, the failure location or the ownership details.

### 3.5.2 Failure Data Aggregation

Hosted in SemKoRe Server, this service aggregates the failure data collected from different gateways in order to produce deep insights of the machine failures and their characteristics including symptoms, impacts, and root causes. We have defined a list of simple aggregations that are applied to the failure data:

1. For each machine type, get the list of all failures and their frequency;
2. For each failure, compute the list of all possible symptoms with the frequency of each symptom;
3. For each failure, compute the list of all possible impacts with the frequency of each impact;
4. For each failure, compute the list of all possible root causes with the frequency of each root cause; and

5. For each failure, get the list of solutions with the number of times each solution was successfully used to repair that failure.

A simplified query used for data aggregation is shown in Listing 3.2.

```

PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX ns: <http://www.schneider-electric.com/KnowledgeRepository#>
Select distinct ?Failure ?aggregated_type ?label (COUNT(?ID) as ?times)
where{
  ?Failure rdf:type ns:Failure.
  ?Occurrence rdf:type ns:FailureOccurrence.
  ?Occurrence ns:hasFailure ?Failure
  ?ID rdfs:label ?label .
  { # Get the list of impacts
    BIND("Impact" AS ?aggregated_type)
    ?Occurrence ns:hasImpact ?ID}
  UNION{
    BIND("Symptom" AS ?aggregated_type)
    ?Occurrence ns:hasSymptoms ?ID}
  UNION{
    BIND("RootCauses" AS ?aggregated_type)
    ?Occurrence ns:hasRootCauses ?ID}
  UNION{
    BIND("solvingProcedure" AS ?aggregated_type)
    ?Occurrence ns:hasSolvingProcedure ?ID}
}
group by ?Failure ?aggregated_type ?ID ?label

```

Listing 3.2: Simplified data aggregation query

### 3.5.2.1 Failure Data Sharing

Hosted in SemKoRe server, this service shares periodically the newly collected maintenance knowledge with the connected SemKoRe Agents. All the data sharing process is done through the SemKoRe Server's message broker. Every SemKoRe agent receives only the updates relative to the machine to which it is connected by subscribing to the message broker topic ".../failure\_updates/{machine\_type}", where {machine\_type} is the type of machine to which the gateway is connected.

### 3.5.3 Implementation Details

Various technologies were used to develop our prototype. To demonstrate the feasibility of the SemKoRe approach, we developed a proof-of-concept (Figure 3.9) using the following

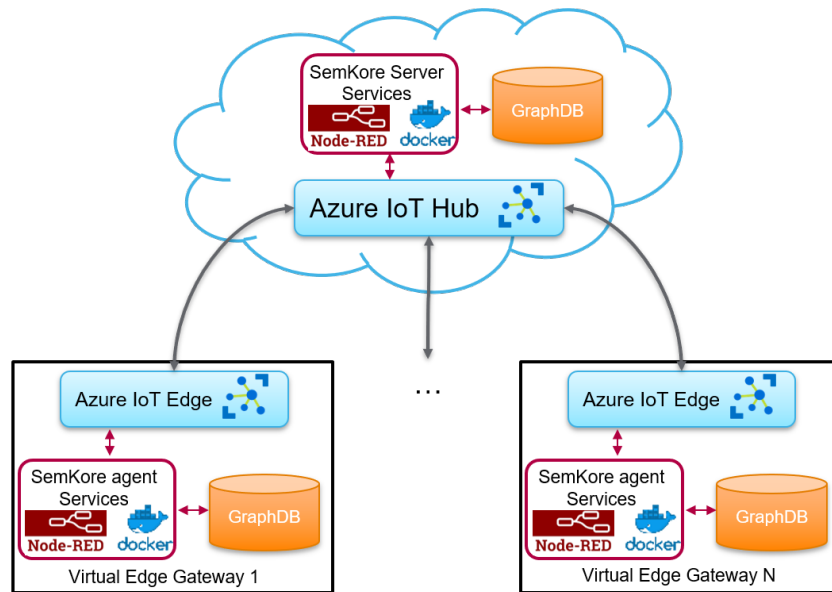


Figure 3.9: SemKoRe implementation setup. Simulated virtual edge gateways with SemKoRe Agents managed by a SemKoRe server instance hosted on the Microsoft Azure cloud.

technologies:

- GraphDB: We used GraphDB [70] as triplestore in the cloud and in the gateway. It provides high performance and scalability in addition to reasoning capabilities.
- Node-Red: Node-Red was used to develop all above-mentioned services for the SemKoRe Server and the SemKoRe Agent. Node-Red is a flow-based development tool for visual programming originally developed by IBM for wiring together hardware devices, APIs (Application Programming Interfaces), and online services as part of the Internet of Things [71]. Node-Red is gaining popularity for rapid application development in Schneider’s Industrial Automation business.
- Microsoft Azure: The SemKoRe Server services are hosted on Microsoft Azure, which provides high-performance cloud services. The server uses Azure IoT Hub [72] to connect the IoT devices to the cloud using several communication protocols, including the MQTT (Message Queuing Telemetry Transport) messaging protocol [73], which we used to simplify the data flow transmission between our SemKoRe agent and server services. For the edge, the Azure IoT Edge service was used to easily connect the edge gateways to the cloud via Azure IoT Hub as shown in Figure 3.9.
- Docker: This is a popular container-based virtualization [74] tool. Docker supports

many operating systems and hardware architectures, and allows self contained applications to be packaged and executed with a high level of portability and reproducible results [72]. We used Docker to package all of the services of SemKoRe Server and SemKoRe Agent. In reality, SemKoRe agent will reside with many other components. Docker allows us to have the flexibility of deploying different components easily and managing/extending them without impacting others.

#### 3.5.4 Evaluation setup

As we are still in the early stage, we were not able to implement the SemKoRe in real conditions with SemKoRe Agent running on Industrial Gateways connected to real machines. The main obstacle was that the current hardware available for this work had ARM architecture and so it would require a significant effort to port the triplestore and other software components. That effort was beyond the scope of our work. However, after the successful PoC, the business team decided to use an industrial PC as a gateway with enough RAM (8GB), processing (Intel Atom), and storage (64GB) capability. The next iteration of this work will use this industrial PC when it is ready for commercialization. In the meantime, we evaluated our implementation by simulating many virtual SemKoRe Agent instances on a PC equipped with an Intel(R) Core(TM) i7-7820HQ processor, and 16Gb of RAM. We used a single GraphDB server to manage separate triplestores for all the SemKoRe Agent instances. We randomly defined a set of three machine types {Packaging Machine, Palletizing Machine, Pasteurization Machine} that we associated with the active SemKoRe Agents. Each machine type was associated with at least two SemKoRe Agent instances which were then connected to the cloud SemKoRe Server. Our current implementation choices will facilitate an easy transition to industrial PCs in the future.

We generated random machine failure data by defining a set of potential failures for each machine type. Each failure was then associated with a set of characteristics: symptoms, impacts, root causes, and solving procedures.

We were able to demonstrate that the failure data was collected by each SemKoRe Agent and successfully shared with the SemKoRe Server. The data was aggregated and shared back with the SemKoRe Agent instances connected to the same machine type.

#### 3.5.5 Performance measurements

To assess the performance of the SemKoRe prototype, we evaluated the execution time of various processes provided by the SemKoRe agent and the SemKoRe server.

In the first case, we measured the performances of two processes of SemKoRe agent: the creation of new failure instances, and the querying of failures details. Figure 3.10 shows the evaluation results as a function of the number of failures stored in the triplestore. We

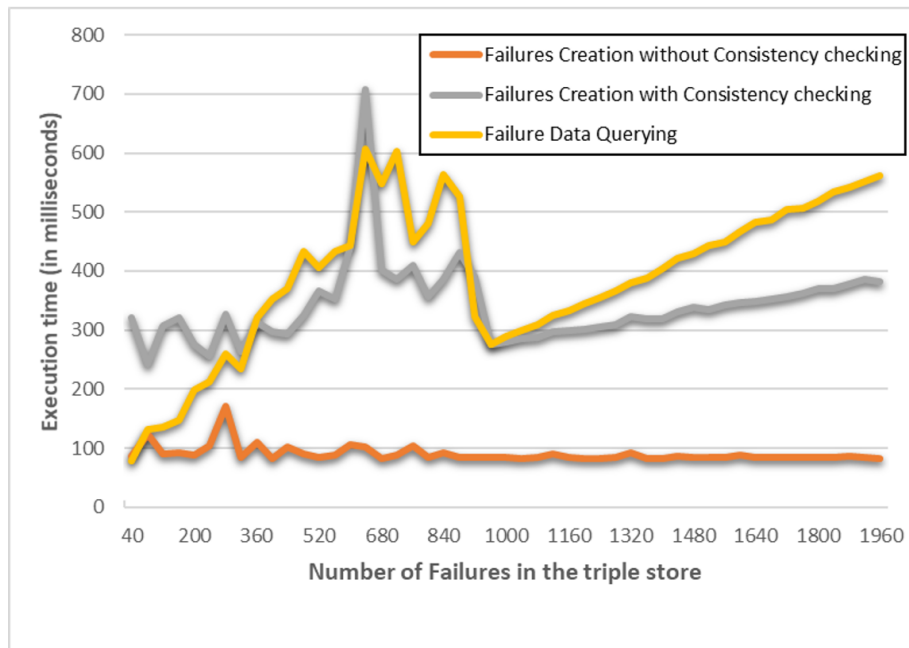


Figure 3.10: Execution time evaluation for the creation and querying of Failure instances

can remark that the data querying time is dependant on the number of failure instances stored in the triplestore. The trend change observed after the x-axis coordinate 840 reflects the activation of the indexing and cache mechanisms of the GraphDB triplestore, which stabilizes the querying time at around 300ms to query the details of 1000 failures.

Regarding the creation of new failure instances, we distinguish two modes: with and without consistency checking. In fact, all SemKoRe services use the REST API described in Chapter 6 for the interactions with the GraphDB triplestore, and that offers the consistency checking feature. When the consistency checking is disabled, we can see that the instances creation's time is stable and is on average less than 100ms. However, the creation of instances with enabled consistency checking requires on average 300ms to 700ms. This extra time can be explained by the need to execute a SPARQL query to check the consistency of the failure details before creating an instance in the triplestore.

In the second case, we evaluated two processes primarily executed on the SemKoRe server: the startup commissioning and the failure data aggregation. The evaluation is done by varying the number of failure instances (in the range [50, 2000]) stored in the triplestore. The box plot in Figure 3.11 shows the results of the evaluation. SemKoRe server needs on average 60ms to 110ms to generate the commissioning data for a specific machine, and the data aggregation takes around 20ms. For both processes, we didn't detect any influence of the number of failure instances on the execution time.

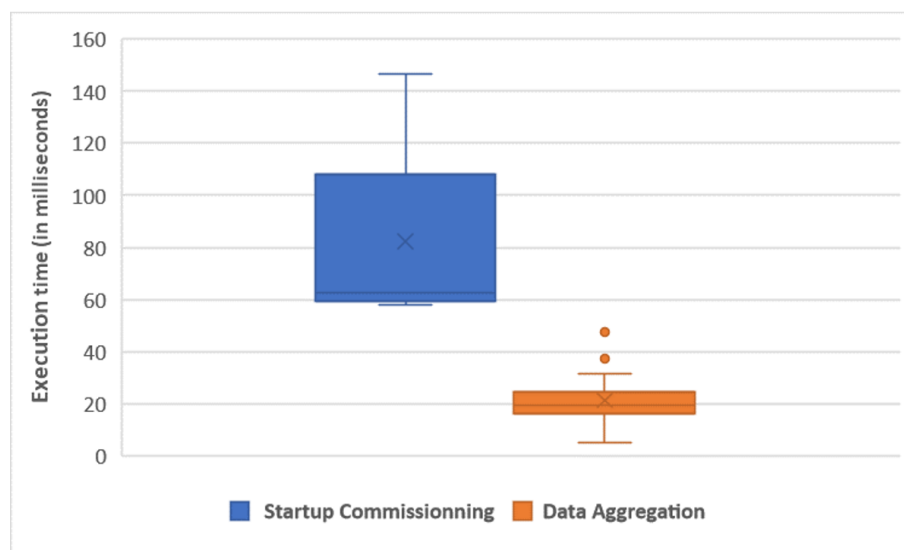


Figure 3.11: Failure ontology.

### 3.6 Learned Lessons

Conducting this study helped us to learn several lessons. The *first* lesson is that technologies such as triplestore are not easily adoptable to typical industrial use cases. Almost all triplestores are focused on big data and huge numbers of triples but, as our work demonstrates, there are several use cases where an efficient solution is needed for typical industrial gateways. Industrial PCs are an option but they are expensive and can only be used by large companies whereas small devices have a very large user base. While machine failures are a reality, they do not occur every minute, and so there is no need to use a complex solution that supports billions of triples. Outside the vendor space, the open-source community has some options like RedStore [76] but most are not in active development.

The second lesson is that the development of industrial-grade ontologies is still a herculean task and the existing toolset continues to act as a barrier to entry. In our experience, experts want to formalize their domain knowledge but they have no motivation to learn complex tools such as Protege that do not support collaborative ontology development. WebProtege is a possibility but lacks query, visualization, and documentation capabilities. New efforts such as Modom.io [77] and Zazuko [78] take a more simplified approach for non-experts to create ontologies but they are still works in progress.

Regarding ontology governance, there is no standard framework that can be applied to design and develop modular ontologies on an industrial scale. The evolution of ontologies is another area where no clear recommendations and no industrial tools are available to manage the required documentation, evaluation, release, and versioning. While some aca-

demio works such as [79] exist, they are not mature and often not easy to deploy and use in industrial settings. The Semantic Web community as a whole needs to address these points and improve the developer experience in order to mainstream these useful technologies.

### 3.7 Summary

In this chapter, we presented the design and prototyping of the SemKoRe approach. First, we detailed the definition of the ontology data models used to collect and structure the maintenance knowledge. Additionally, we presented our flexible architecture, which fits in three scenarios to cover the various configurations of Schneider's customers. We described the proof-of-concept implementation along with the performance evaluation.

Based on this early work, our customers showed an interest in using the SemKoRe approach to enhance their industrial maintenance processes. Furthermore, by using the SemKoRe approach, the overall machine building process can be optimized. The machine design phase can benefit from the maintenance feedback to identify any weaknesses of a machine and can improve its design. Furthermore, the collected statistics will allow the performance comparison of a particular machine working in different locations and contexts. Thus, additional services and recommendations can be proposed to the customers in order to optimize their manufacturing process.





# Privacy-preserving Sharing of Industrial Maintenance Reports

## Contents

---

<b>4.1 Sensitive Data in Maintenance Reports . . . . .</b>	<b>75</b>
<b>4.2 Sensitive Data Detection Approach in Maintenance Reports .</b>	<b>76</b>
4.2.1 Overview . . . . .	76
4.2.2 Sensitive data detection flow in maintenance reports . . . . .	78
<b>4.3 Data corpus collection &amp; preparation . . . . .</b>	<b>79</b>
4.3.1 IMDP - Industrial Machine Data Pool . . . . .	79
4.3.2 Data Sources and Data Structure . . . . .	81
4.3.3 Texts Annotation . . . . .	83
4.3.4 Summary . . . . .	84
<b>4.4 Machine ontology definition . . . . .</b>	<b>85</b>
<b>4.5 Proof-of-Concept Prototype . . . . .</b>	<b>86</b>
4.5.1 Training Named Entity Recognition models for machine components detection . . . . .	86
4.5.2 Implementation details . . . . .	92
4.5.3 Evaluation . . . . .	93
<b>4.6 Limitations and Improvement Areas . . . . .</b>	<b>94</b>
<b>4.7 Conclusion . . . . .</b>	<b>95</b>

---

This chapter presents a new approach to provide a remedy for sensitive data disclosure problems during the knowledge sharing activity, along with performance measurements. The chapter is based on papers II and III and is the subject of Patent III. It addresses the following research question:

*What is the nature of sensitive data in maintenance reports? How to avoid sensitive data disclosure during the sharing of maintenance knowledge? What techniques could be applied for the privacy-preserving needs and how?*

The interviews campaign results were unequivocal; sharing maintenance knowledge between manufacturers would have a positive impact on the optimization of maintenance routines and on manufacturing productivity. However, a major problem that would prevent such solutions from becoming a reality, is the presence of sensitive information in maintenance reports. It is therefore essential to guarantee data privacy-preserving and avoid sensitive data disclosure during the sharing of maintenance data. And to the best of our knowledge, there exists no work in the literature about sensitive data detection in maintenance reports and this is the first time such a topic is studied.

For this purpose, a new approach is proposed to avoid sensitive data disclosure during maintenance knowledge sharing through the SemKoRe solution. It relies on Semantic Web ontologies combined with different techniques, usually used for data anonymization [80], such as Natural Language Processing (NLP) and rule-based sensitive data detection.

We tackled a couple of challenges during the work on this contribution. The first challenge is the lack of datasets with a reasonable number of real maintenance reports containing sensitive data. Fortunately, the interview campaign allowed us to understand the nature of sensitive data in maintenance reports and some ways to recognize it. The second challenge is that sensitive data detection techniques, especially the NLP-based ones, require annotated data corpus with a considerable number of samples, while we have very few reports with industrial maintenance vocabulary. So, for the proof-of-concept needs, we implemented a generic solution that we used to collect, annotate and construct, in a collaborative manner, our own data corpus.

First, the collected data corpus was used to train and evaluate three different NLP models for Named Entity Recognition (NER). NER remains one of the most used NLP techniques for sensitive data detection. It consists of identifying, within a text, the entities (words or group of words) that are relative to real-world objects with associated names. Secondly, all the trained models are deployed and used on an Edge gateway. The results are promising, they show that our approach can be used for on-premise detection of potentially sensitive data in maintenance reports. Several areas or improvements are identified to make our solution usable in real use cases.

## 4.1 Sensitive Data in Maintenance Reports

During several investigation months that coincide with the COVID-19 context, we were unable to collect more than five maintenance reports containing sensitive data. So, the interviews campaign was our main source of information about sensitive data in maintenance reports. In fact, there exist mainly three types of sensitive data in maintenance reports:

- Personal data: all information relative to a specific person within the company. E.g.: Personally Identifiable Information (PII) like employee's name or number, role, addresses, phone number. However, this type of data is almost impossible to find in the *content* of a maintenance report.
- Business data: every information about the company or its activity. This includes the manufacturer's name, products, location, customers, or subcontracting companies.
- Manufacturing data: especially information about the manufacturing process that may leak details about the manufactured products or the trade secrets, e.g. secret recipe of Coca-Cola. Also, some industries believe that the machine configuration is sensitive since it is part of the competitive know-how and the company's industrial property.

Business data differ from one company to another but have a common form. They are mostly a set of proper nouns and well-known entities such as employees' names, products, customers, companies, etc...

Concerning the manufacturing data, there is a common pattern of the machine configuration and manufacturing process in the majority of maintenance reports. Actually, for every machine failure, the maintenance technician usually describes how a machine component has been used or has behaved, or what configuration it had when the failure occurred. Such information is most of the time relative to machine components since they are the most representative landmarks in a machine to describe a failure or a maintenance action. This hypothesis was approved by the domain experts to be the most representative of the manufacturing data commonly found in the maintenance reports. Nevertheless, they do not exclude other rare forms not covered in this thesis, and for which the proposed approach can provide a solution.

Finally, it is important to note that for standard machines, most of the machine configurations or manufacturing processes are provided by the OEM or the machine builder, therefore, they are not sensitive nor confidential. The exceptions are more relative to the customization applied on the standard machines such as adding new components or using a secret configuration that requires high engineering expertise. In both cases, only the

manufacturing engineer or responsible can evaluate the sensitivity of each manufacturing data.

## 4.2 Sensitive Data Detection Approach in Maintenance Reports

### 4.2.1 Overview

To detect sensitive data in maintenance reports, we adopted a hybrid approach relying on rule-based techniques and Named Entity Recognition (NER). In addition to these techniques, we used Semantic Web ontologies to guarantee the customizability, portability, and usability by different stakeholders for different machine types.

For the detection of business data such as the manufacturer name, products names, customers, and subcontracting companies; we use a dictionary lookup technique. We ask the stakeholders to provide a list of all typically used proper names (products, people, companies, ...etc) in the company. This list can be updated whenever new entities are introduced in the company.

As presented above, the machine components are the most common form of machine configuration or manufacturing process leaks. For that purpose, we use Named Entity Recognition techniques for the identification of the machine components in the text.

Finally, as discussed previously, recognizing the sensitive nature of a piece of data can only be done (for the moment) by a human expert. Therefore, to provide a highly customizable solution, we propose an ontology-based approach to allow the domain experts to specify the machine components that they judge as being potentially sensitive.

Figure 4.1 shows a detailed overview of our approach. It requires four different inputs:

1. A machine components taxonomy bringing together the components' names, their synonyms, and abbreviations.
2. An annotated text corpus that will be used for the training of the NER module. This corpus should be composed of numerous text entries with tagged machine components. As a requirement, all tagged machine components must be part of the machine taxonomy.
3. A machine ontology that describes a physical machine and all its components. This ontology must be restricted to the components defined in the machine taxonomy. In this way, we guarantee the synchronization between the components identified by the NER model and the components referenced in the machine ontology.
4. A data anonymization rules ontology that defines the different ways used to identify

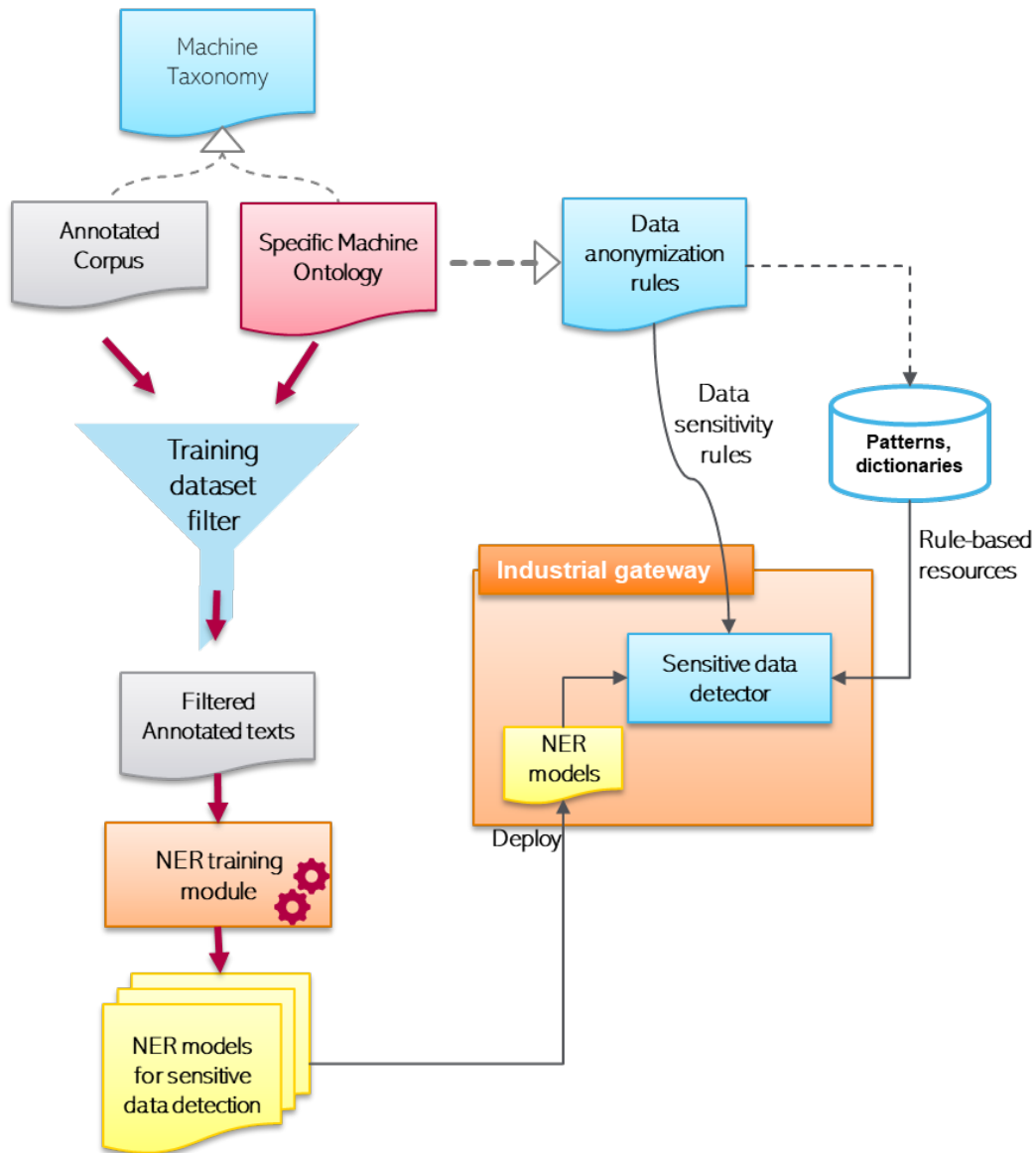


Figure 4.1: Detailed overview of sensitive data detection in maintenance reports

the various types of sensitive data. It is imported in the machine ontology to allow domain experts to specify the sensitivity rules or flags for each machine component, i.e. flag a component as potentially sensitive or not sensitive. This ontology also references the rule-based resources (e.g. dictionaries) that are used during the sensitive data search phase.

Note that the outcome of our approach is a custom tool for the detection of (potentially) sensitive data in the maintenance reports of a specific target machine. Although, with

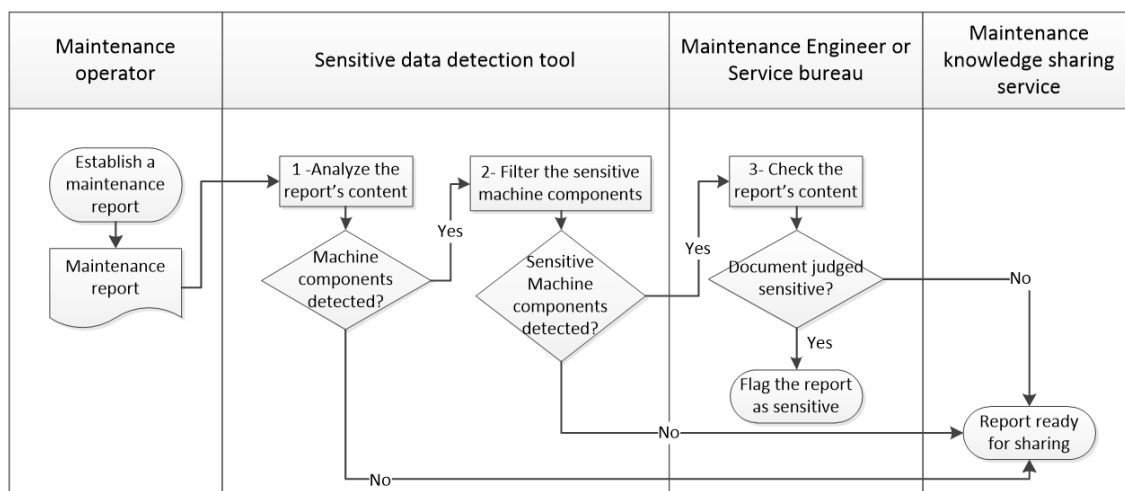


Figure 4.2: Sensitive data detection flow for maintenance data sharing

adequate inputs, our approach is applicable to a variety of machines from different domains.

Once all the inputs are provided, the *training dataset filter* uses the *specific machine ontology* to filter the *annotated corpus*. It also ignores all tagged machine components that are not part of the target machine. As a result, we get a *filtered annotated corpus* that contains samples tagged only with the components of the target machine. The reason for this choice is that we consider the annotated corpus of general-purpose and that might reference several thousands of machine components, while only a few dozens of components are part of the target machine.

The *filtered annotated corpus* is then used to train a *NER model* for the machine components detection. It is known that reducing the training datasets helps in accelerating the training phase. Also, this allows the NER models to focus on efficiently recognizing a limited number of entities instead of trying to recognize all existing machine components. This step provides a *custom NER model* trained to detect only the components of the machine for which the maintenance reports are drawn up.

Finally, we deploy the trained NER model on the industrial gateway connected to the target machine. Then, the *Sensitive data detector* service will use on-premise: 1) the trained NER model, 2) the data sensitivity rules, and 3) the rule-based resources during the analysis of the maintenance technician's inputs for the identification of (potentially) sensitive data.

#### 4.2.2 Sensitive data detection flow in maintenance reports

Figure 4.2 describes the flow of sensitive data detection in maintenance reports, it consists of three main steps:

1. The first step consists of automatically analyzing the content of the maintenance

report and identifying all machine components in it.

2. The list of identified machine components is then filtered in order to keep only the items judged potentially sensitive. When sensitive components are detected, the maintenance report is flagged as containing potentially sensitive data and requires human checking.
3. Finally, the domain expert decides if the report can be judged sensitive or not. All reports identified as non-sensitive are tagged as ready for sharing with the maintenance knowledge sharing service.

### 4.3 Data corpus collection & preparation

The major challenge that we had to face in this contribution was to find the needed datasets with industrial maintenance vocabulary. Traditional sources such as internal documents or open datasets were not helpful for our needs. Therefore, we decided to collect and construct the required data corpus ourselves. We defined two steps to build our datasets:

1. Collect structured data and texts about machine components from the internet. The structured data will be used to construct the machine components' taxonomy, and the texts will be used to build the data corpus.
2. Annotate the collected texts using the machine components taxonomy. This step is divided into two sub-tasks:
  - (a) Automatic annotation: It consists of using tools to annotate automatically the collected texts and to optimize the annotation effort.
  - (b) Manual annotation: In this task, the user must annotate and correct the entities not detected or incorrectly annotated by the automatic annotation tools.

For this purpose, we implemented the IMDP (Industrial Machine Data Pool), which is a collaborative solution with the necessary services for data collection and automatic and manual annotations.

#### 4.3.1 IMDP - Industrial Machine Data Pool

Several tools are proposed in the internet for web scraping and corpus construction, e.g.: Data Scrapper from Data Miner <sup>1</sup>, OpenLink Structured Data Sniffer <sup>2</sup>, or Web Scraper

---

<sup>1</sup><https://dataminer.io/>

<sup>2</sup><https://osds.openlinksw.com/>



## Machine element

From Wikipedia, the free encyclopedia

**Machine element** or **hardware** refers to an elementary component of a [machine](#). These elements consist of three basic types:

1. *structural components* such as frame members, [bearings](#), [axles](#), [splines](#), [fasteners](#), [seals](#), and [lubricants](#).
2. *mechanisms* that control movement in various ways such as [gear trains](#), [belt](#) or [chain drives](#), [linkages](#), [cam](#) and [follower](#) systems, including [brakes](#) and [clutches](#), and
3. *control components* such as buttons, switches, indicators, sensors, actuators and computer controllers.<sup>[1]</sup>

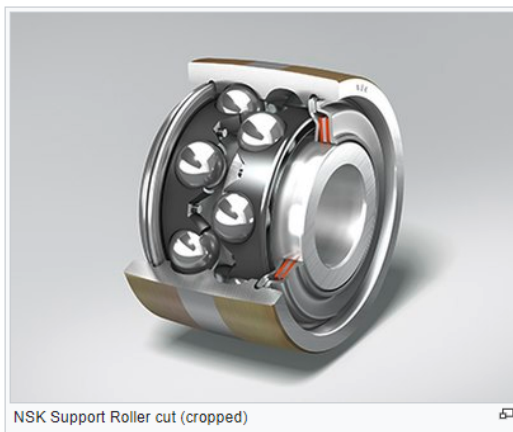


Figure 4.3: Example of Wikipedia page with highlighted links

from OpenLink Software <sup>3</sup>. These tools are more designed to automate the collection of structured data, however, the manual effort needed to collect unstructured text data makes them useless. Thus, we decided to develop our own tool to accelerate the data collection task. We designed an extension<sup>4</sup> that can be integrated into internet browsers. This allows the user to interact with the data collection tool directly from the visited web pages and collect adequate content. Indeed, the data collection interface is accessible directly from the browser's contextual (right-click) menu. In addition, several users can simultaneously use the extension and participate in the data collection effort. Each instance of the browser extension manages a local database permanently synchronized with a central cloud database that is common to all users (see Figure 4.4). In order to avoid duplicate entries, all the links of the pages that are already collected are highlighted on every visited web page as shown in Figure 4.3.

IMDP server offers also a web service that can be used similarly to the internet extension, but without integration in the visited web pages.

<sup>3</sup><https://webscraper.io/>

<sup>4</sup>The developed extension is hosted in <https://github.com/hicham0100/Taxonomy-Collector-Extension>

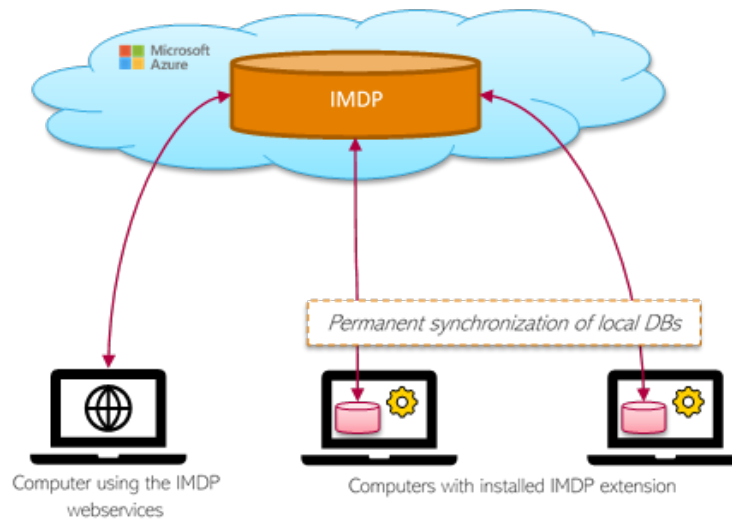


Figure 4.4: IMDP's architecture

### 4.3.2 Data Sources and Data Structure

We configured IMDP to collect data about machine components, and we choose Wikipedia as a starting point because of the availability of a large number of pages describing the industrial machines and components. Also, the unified structure of Wikipedia's articles makes it possible to automate the data extraction. From each web page, we gather the following details:

- **Entry type:** differentiating three sub-types of machine component: structural, mechanical, and control element.
- **Name:** usual English name of the entity
- **Synonyms:** the different names and abbreviations used to describe the same entity
- **Description:** short description of the entity
- **Category or parent entity** if the entity is a derivation or sub-type of another entity
- **URLs** (preferably from Wikipedia): list of web pages describing the entity, we found that many different links may be redirected to one single page in Wikipedia. So, we keep track of all found links relative to each entity.
- **Relative texts** that will be used to construct our data corpus.

Figure 4.5: Screenshot of the data collection Popup

Many of these details are automatically gathered from Wikipedia pages. For example, to collect data about "Engine" as a mechanical element, we just need to open the Wikipedia page <https://en.wikipedia.org/wiki/Engine>, right-click to get the contextual menu, and click on the option "New entry from this page". Then, a modal form/popup (Figure 4.5) appears with some pre-filled input fields (name, Description, Wikipedia URL and relative texts). The name, image and description information are gathered from the Wikipedia's Rest API<sup>5</sup>, while the relative texts are extracted from the content of the current page by excluding non textual elements such as: images, tables and references. On the other hand, the remaining fields (Entry type, Synonyms and parent entity) need to be filled by the user.

The collected data can be then extended, updated or deleted by clicking on the extension's button as shown in the Figure 4.6.

In addition to the machine components data, other categories were collected such as (Machine Failure, Maintenance Process, Manufacturing Process, and Materials). These categories are not considered in this thesis but are potentially useful for future works about

<sup>5</sup>[https://en.wikipedia.org/api/rest\\_v1/](https://en.wikipedia.org/api/rest_v1/)

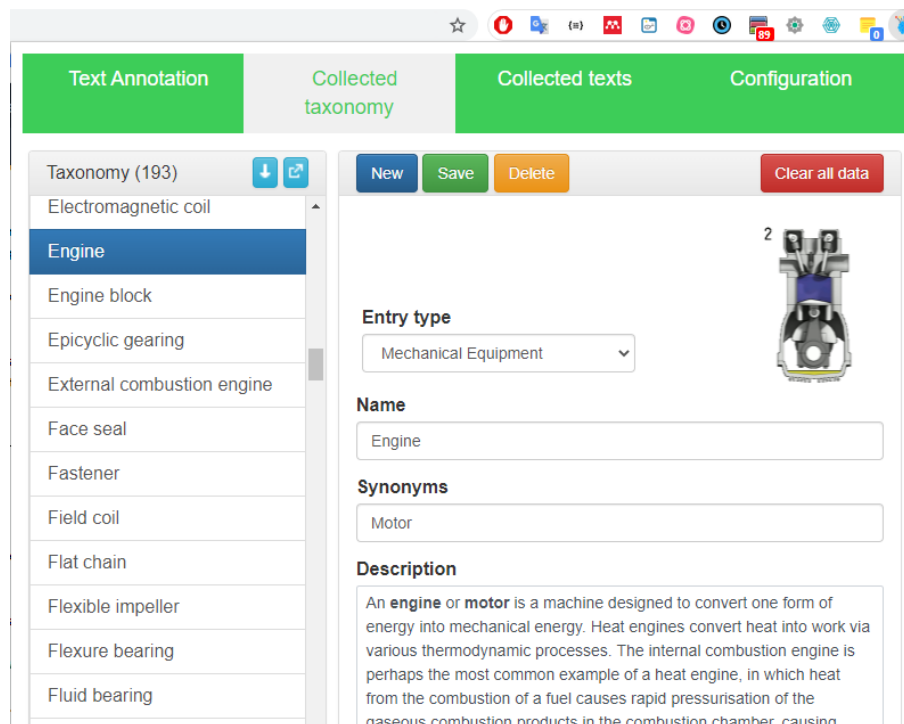


Figure 4.6: Screenshot of the MDPI's collected taxonomy view

maintenance reports analysis and information extraction.

Afterward, the collected texts must be manually cleaned and annotated. We split all collected texts into paragraphs with less than 800 characters. The goal is to simplify the text annotation and to attract more colleagues for a collaborative text annotation effort.

### 4.3.3 Texts Annotation

#### 4.3.3.1 Automatic Annotation

To accelerate the text annotation task, we implemented an automatic annotation tool that identifies and automatically tags machine components in the text. Three different techniques are used for automatic annotation:

- Exact keywords lookup from the collected machine taxonomy. We also include the plural form of every entity by applying the English grammar's plural rules.
- Use of Wikipedia annotations: in Wikipedia articles, the annotations appear in the form of links to other Wikipedia pages. When we find a link with a known URL related to a collected entity, we consider the link's text as a reference to that entity. Even if it is not exhaustive, this source of annotations is very accurate since it is done

manually by Wikipedia’s contributors. However, it helps to improve the annotations and to find synonyms of the existing entities.

- Use of Tagme annotations: Tagme<sup>6</sup> is a service that performs on-the-fly semantic annotation of short text using Wikipedia as a knowledge base. It proposes a REST interface that can be used to automatically annotate texts; every entity detected by Tagme service is tagged with a corresponding Wikipedia URL. Then, in practice, we follow the same steps as for Wikipedia’s annotations, but with a lower confidence degree. In fact, Tagme allows the detection of many machine components in the texts, but it generates a non-negligible rate of false-positive annotations. Even by varying the sensitivity parameters, it tends to identify too many entities using semantic equivalence, e.g. the verb ‘sense’ is identified as the entity ‘sensor’. Also, many multi-words entities are broken like in the example: with the sentence *"Conical rotor brake motors are used to power micro speed drives."*, Tagme identifies 6 entities {"Conical", "rotor", "brake", "motors", "speed", "drives"} instead of 2 entities {"Conical rotor brake motors", "speed drives"}.

Using the automatic annotation approach, all the collected texts are automatically annotated before being proposed to the users for manual annotation.

#### 4.3.3.2 Manual Annotation

During the manual annotation task, the user needs to check the correctness of automatic annotations and annotate the missed entities. Figure 4.7 shows the interface used for manual annotation. When a new entity is identified by the user, it is added to the machine taxonomy either as a synonym of an existing entity or as a new entity for which the user is asked to provide the different details required by the IMDP. This allows to keep track of all entities in the texts and to know to which entity every annotation refers.

For the collaborative texts annotation, 10 documents (paragraphs) were selected and proposed without annotations to all the annotators. After, we analyze the annotation provided by each annotator in order to evaluate the inter-annotator agreement. This method is used in the literature [81] to estimate the relevance of annotations of each annotator. However, this aspect was not evaluated due to a lack of volunteers.

#### 4.3.4 Summary

At the time of writing these lines, we built a dataset composed of 193 taxonomy entries with 283 synonyms and abbreviations, 188 articles’ texts, 3523 cleaned paragraphs, and 333 fully annotated paragraphs containing 1591 sentences.

---

<sup>6</sup><https://tagme.d4science.org/>

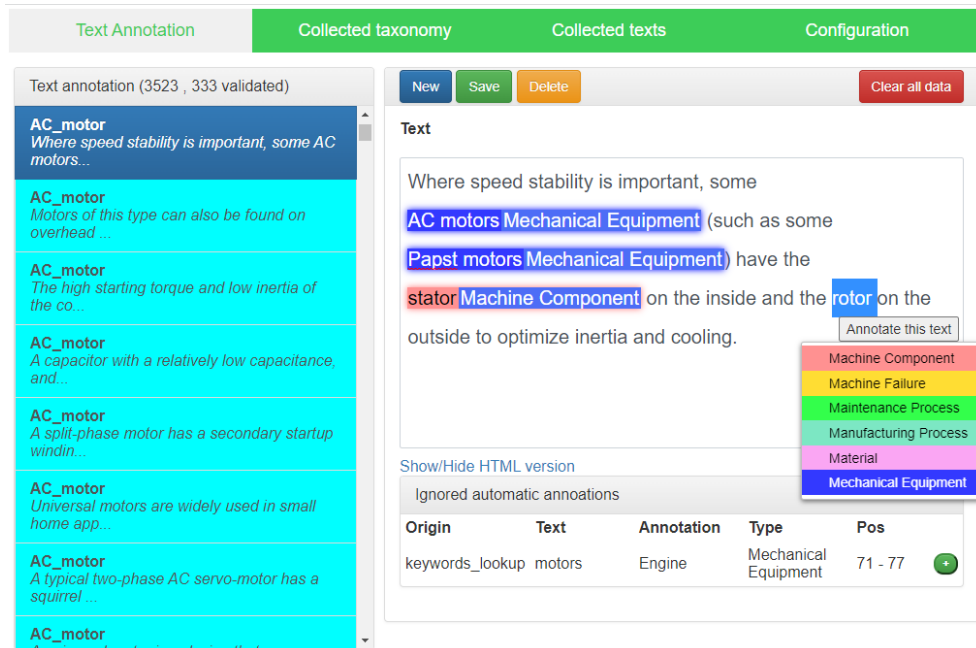


Figure 4.7: MDPI's Annotation interface

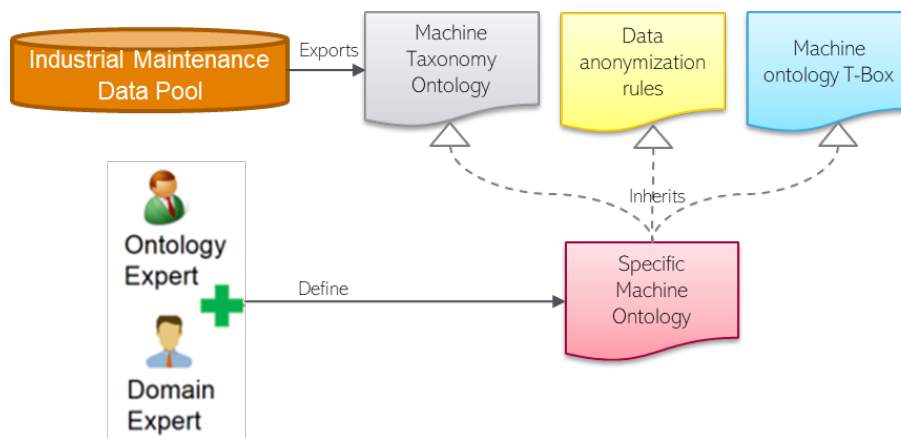


Figure 4.8: Machine ontology design

## 4.4 Machine ontology definition

In Chapter 3, we have seen that the SemKore approach requires the definition of an ontology that describes the physical machine and the different machine components. That ontology is usually defined by a domain expert (e.g. machine designer) with the help of an ontology expert to fulfill the requirements defined in section 3.2.2.

Now, we have extended the set of requirements of machine ontology in order to support the sensitive data detection approach (see Fig. 4.8). We defined a T-Box for the Machine

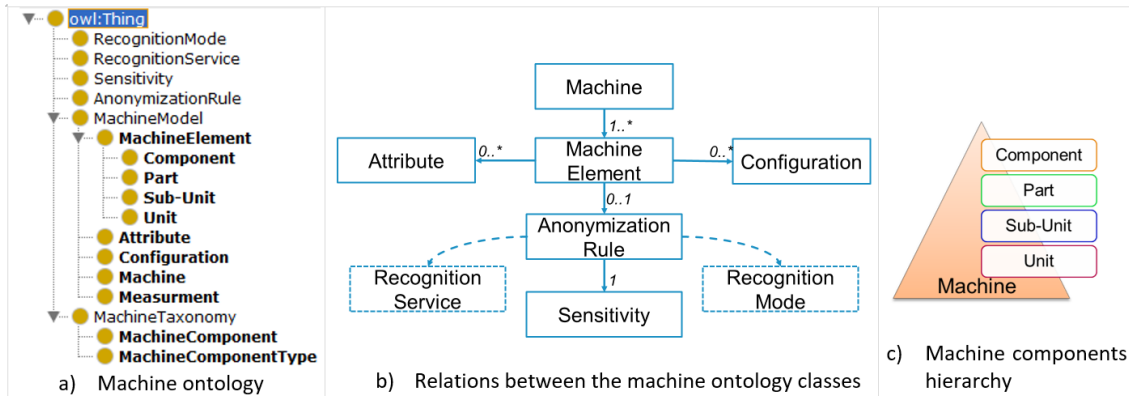


Figure 4.9: T-Box & Machine ontology specifications

Ontology Model (see Fig. 4.9a) that must be used to create an instance of the physical machine. The T-Box ontology describes the components of the machine and defines the characteristics of each component such as its attributes or its configuration (Figure 4.9b).

Every machine is described as a hierarchical assembly of the different physical elements (Unit, Sub-unit, Part, and Component) as shown in the Pyramid in Figure 4.9c. One of the requirements of the Machine T-Box is that all machine components must be referenced in the Machine taxonomy ontology that was generated previously by IMDP.

A Data Anonymization Rules ontology is also created to define, at this stage, the sensitivity flags relative to each machine component. In other words, it classifies a machine component as "sensitive" or "not sensitive". The sensitivity flag can also be defined for the other machine elements (Unit, Sub-unit, and Part). This ontology will be extended in the future to cover more data anonymization aspects such as managing multiple sensitive data detection services (e.g. dashed concepts in Figure 4.9.b) or supporting the sensitive data masking or replacement rules.

The tables 4.1, 4.2 and 4.3 describe in detail the classes, object, and data properties of the ontologies: Anonymization rules, Machine model, and Machine taxonomy ontologies.

## 4.5 Proof-of-Concept Prototype

### 4.5.1 Training Named Entity Recognition models for machine components detection

The goal of this task is to train Name Entity Recognition models in order to detect machine components and mechanical equipments in maintenance reports' textual content. Three different NER models were trained:

1. Custom Spacy NER model: Spacy [82] is a very known open-source framework for NLP. It offers several features including a sophisticated deep CNN-based NER system

Table 4.1: Classes description of the Anonymization rules, Machine model, and Machine taxonomy ontologies.

Ontology	Class Name	Super Class	Description	Example of Instances
Anonymization Rules	Sensitivity		Defines the sensitivity flags for each machine element.	True, False
Machine Model	Attribute		Defines an attribute of a machine element, attributes are immutable	SerialNumber, Manufacturer
	Configuration		Defines a configuration of a machine element along with default configuration values	IP Address, Rotation speed
	Machine		Defines a physical machine as a set of machine elements	Tray Sealer machine XYZ
	Machine Element		Defines the components, parts units and sub-units of machines	
	Unit	Machine Element	Defines the machine as a composition of one or more machine Units	Soldering Unit
	Sub-unit	Machine Element	Defines the machine Units as a composition of one or more Sub-units	Robot XYZ
	Part	Machine Element	Defines the machines Sub-Units as composition of one or more Parts	Servomotor
	Component	Machine Element	Defines the machine Parts as composition of one or more machine components	Variable Speed drive
	Measurement		Describes a values measured by/on a machine element	Temperature, Pressure, Speed
Machine Taxonomy	Machine Component		Defines a machine component in the machine taxonomy	Belt, Gear, Servodrive
	Machine Component Type		Defines the type of the machine component	Control, Motion, Structure

achieving state-of-the-art performances. It also offers the possibility to train custom NER models based on its NER system.

2. NER model with CRF (Conditional Random Fields): CRF is a class of statistical methods designed for the analysis of sequential data (such as text, images, DNA) [83]. They're often used in pattern recognition and machine learning. One reason for their good performances in the NER task is that they consider the input's context by taking into account the neighboring or surrounding samples.
3. BERT-based NER model: BERT (for Bidirectional Encoder Representations for Transformers) is a deep learning model that has given state-of-the-art results on a wide variety of natural language processing tasks. It has been pre-trained on Wikipedia and



Table 4.2: Object properties description of the Anonymization rules, Machine model, and Machine taxonomy ontologies.

Ontology	Object Property	Parent	Domain	Range	Description
Anonymization Rules	has Anonymization Rules		Machine Element	Sensitivity	Associates a machine element to a sensitivity flag. It will evolve in the future to cover more complex anonymization rules.
Machine Model	Has Attributes		Machine Element	Attribute	Defines an attribute for a machine element
	Has Configuration		Machine Element	Configuration	Defines a configuration for a machine element
	Has Elements		Machine Element	Machine Element	Specifies all the elements of a specific machine
	has Machine Element Parent		Machine Element	Machine Element	Defines a hierarchy between machine elements by specifying a parent element
	has Measurement		Machine Element	Measurement	Associates a measurement to a machine element
	has Unit	Has Elements	Machine	Unit	Defines a machine as a composition of machine units
	has Sub-Unit	Has Elements	Unit	Sub-unit	Defines a machine unit as a composition of machine sub-units
	has Part	Has Elements	Sub-unit	Part	Defines a machine sub-units as a composition of parts
	has Component	Has Elements	Part	Component	Defines a part as a composition of machine components.
Machine Taxonomy	has Component Type		Machine Element	Machine Component Type	Defines the type of a specific machine component in the machine taxonomy

BooksCorpus [84], and requires task-specific fine-tuning [85]. In our case, we applied BERT to build our NER model for machine components detection.

#### 4.5.1.1 Data input

To train the NER models, we used the dataset collected via the IMDP tool. The size of the dataset is shown in the table 4.4. As presented previously, many tags are used for the data annotations and the number of instances of each class is shown in the table 4.5. In this task, we consider only the classes: *MACHINE COMPONENT* and *MECHANICAL EQUIPMENT* as they represent more than 95% of the tagged entities and are more meaningful for our evaluation.

Table 4.3: Data properties description of the Anonymization rules, Machine model, and Machine taxonomy ontologies.

Ontology	Data Property	Domain	Range	Description
Anonymization Rules	has Sensitivity Flag	Sensitivity	xsd:boolean	Defines a sensitivity flag (True or False) for a machine element
Machine Model	Measurement Timeseries ID	Measurement	xsd:string	Defines the identifier used to store the measurements in the time series database
	Measurement Timestamp	Measurement	xsd:dateTimeStamp	Defines the timestamp of the last measured value
	Measurement Value	Measurement	rdfs:Literal	Defines the last value measured by/on a machine element
	Attribute Value	Attribute	xsd:string	Specifies the attribute value of a machine element
	Attribute Name	Attribute	xsd:string	Specifies the attribute name of a machine element
	Attribute Type	Attribute	xsd:string	Specifies the attribute type of a machine element
	Description	owl:Thing	xsd:string	Provides a description for an individual
	Config Name	Configuration	xsd:string	Specifies the configuration name of a machine element
	Config Type	Configuration	xsd:string	Specifies the configuration type of a machine element
	Config Value	Configuration	xsd:string	Specifies the configuration value of a machine element
	Default Values	Configuration	xsd:string	Specifies the configuration' default values for a machine element
Machine Taxonomy	Component Name	Machine Component	xsd:string	Defines the usual component name in the machine taxonomy
	Has Abbreviation	Machine Component	xsd:string	Defines the abbreviations frequently used for a machine component
	has Synonym	Machine Component	xsd:string	Defines the synonyms of a machine component
	Machine Components Type Name	Machine Component Type	xsd:string	Defines the name of the machine component's type

Table 4.4: The dataset volumes used for the training of the NER models

	Paragraphs	Sentences	Tokens
Volume	333	1591	37244

Table 4.5: Number of instances of each class used in the dataset

Class	Number of instances
MACHINE COMPONENT	1736
MECHANICAL EQUIPMENT	802
MATERIAL	85
MANUFACTURING PROCESS	24
MACHINE FAILURE	9
MAINTENANCE PROCESS	0

#### 4.5.1.2 Data output

The expected result from this task is to detect "Machine components" and "Mechanical Equipments" entities in the input text. For example, let's consider the following text: *"A dead axle, also called a lazy axle, is not part of the drivetrain, but is instead free-rotating. The rear axle of a front-wheel drive car is usually a dead axle."*

The expected result from the analysis of this text with the NER models is shown in the figure 4.10 bellow.

A dead axle MACHINE COMPONENT, also called a lazy axle MACHINE COMPONENT, is not part of the drivetrain MACHINE COMPONENT, but is instead free-rotating. The rear axle U-MACHINE\_COMPONENT of a front-wheel drive car MECHANICAL EQUIPMENT is usually a dead axle MACHINE COMPONENT.

Figure 4.10: Example of automatic detection of Machine Components and Mechanical Equipments in the text.

#### 4.5.1.3 Data preprocessing

**Data preprocessing for Spacy** To train the Spacy model, the annotated data needs to be encoded according to the BILUO [86] scheme. This format consists of tagging each token in the data as follows:

- B-*<class>*: the first token of a multi-token entity belonging to *<class>*.
- I-*<class>*: an inner token of a multi-token entity belonging to *<class>*.
- L-*<class>*: the last of a multi-token entity belonging to *<class>*.
- U-*<class>*: a single-token entity belonging to *<class>*.
- O: a non-entity token.

**Data preprocessing for CRF** The CRF requires extracting features from the texts before the training step. For that purpose, we first tokenize the input texts, then we extract

Table 4.6: Training parameters for Spacy, CRF, and BERT

SPACY		CRF		BERT	
Parameter	Value	Parameter	Value	Parameter	Value
Base model	Empty model	L1 penalty	0.44	Base model	uncased BERT base model
Batch size	compounding (4, 32, 1.001)	L2 penalty	1e-4	Batch size	16
Iterations number	15	Max iterations	60	Epochs	3

a vector of 11 features from each token, such as the lower case format, suffixes characters, various flags (uppercase, capital, alphanumeric, stop word), and part of speech (PoS) Tags. Afterward, each token’s vector is concatenated with the vectors of the previous 5 tokens and the vectors of the next 5 tokens. This allows covering the long multi-token machine components like “*glass-ceramic-to-metal seal*” that is composed of 8 tokens (5 words and 3 hyphens).

**Data preprocessing for BERT** BERT uses a built-in tokenizer that allows preprocessing data before the training step. For instance the sentence “*agitators come in many sizes and varieties.*” is tokenized into [’ag’, ’###ita’, ’###tors’, ’come’, ’in’, ’many’, ’sizes’, ’and’, ’varieties’, ’.']. This creates flexibility, as the tokenizer can always create tokens for a given sequence, regardless if the word has been seen previously by the model. This is especially useful for NER as some names may be very unusual and not occur in the training dataset [87].

#### 4.5.1.4 Training setup

For the training phase, we used different parameters for our NER models as shown in the table 4.6. For Spacy’s base model, we can use either an empty model that will be trained from scratch to detect the machine components entities or reuse the Spacy’s trained NER model to make it able to detect as well the already supported entities (Person, Organization, Addresses, ...etc) and the machine components entities. We chose to use an empty model to have an accurate evaluation of the trained model for our specific task. Regarding the BERT’s base model, several possibilities are proposed such as uncased BERT base model, uncased BERT large model, cased BERT base model, or Cased Multilingual with 104 languages. We chose an uncased BERT model since we have no need to preserve the upper case format for the detection of machine components. Also, we used the smaller model as it requires less expensive computer hardware for training.

#### 4.5.1.5 Evaluation setup

We adopted the cross-validation or k-folding approach for the evaluation of the trained NER models. This, we divided our dataset into  $n=5$  splits and trained the models in 5 iterations. For each iteration  $i$ , we trained each of the NER models by considering the  $i^{th}$  data split as testing data, and the remaining data splits as training data. We also adopted also the F1-score as an evaluation metric. It is defined as follows:

$$F1 = 2 * \frac{precision * recall}{precision + recall}$$

Where the *precision* is the percentage of named entities found by the NER model that are correct, and the *recall* is the percentage of named entities present in the corpus that are found by the model.

#### 4.5.2 Implementation details

Multiple technologies are used to implement this proof-of-concept. For IMDP we used:

- Google Chrome as a target for the browser extension as it is quite a popular browser worldwide.
- HTML+CSS and Javascript to implement the web UI, and the different chrome extension services, such as data collection, cleaning, annotation, and data export.
- PouchDB, as a storage database on both, local storage and on the cloud storage. PouchDB is a no-SQL database with native data synchronization features between the local and cloud databases.

The creation of the different T-Box ontologies (Machine Taxonomy, Data Anonymization Rules, Machine Model) was mainly done with the Protégé tool, which is widely used for the creation of semantic web ontologies.

We used the Python programming language to implement the "Training dataset filer" service, and for the NER training module. Specific python libraries were used for the implementation and training of the different NER models:

- Spacy library for the custom spacy NER model.
- pyCRFsuite for the NER model with CRF.
- PyTorch library for NER model with BERT.

For the training of the different NER models, we used a capable laptop setup running Ms. Windows 10, with 32Gb of RAM, an octa-core i7 processor, and a CUDA-enabled

GPU (NVIDIA QUADRO M2020) with 12Gb of memory. As an industrial gateway, we used RaspBerry Pi 4, with 4Gb of RAM, ARM-Cortex-A72, and running Linux Ubuntu 5.4. Docker was used as a containerization technology for the different Edge services such as the *Sensitive data detector* service.

Finally, we used Microsoft Azure to host the IMDP server containing the master PouchDB. Currently, the training dataset filter and NER training module services were implemented and tested on a laptop, and there is a plan to deploy them on a dedicated Ms. Azure server. On the RaspBerry Pi, we deployed the trained NER models and the *sensitive data detector* service.

### 4.5.3 Evaluation

To evaluate our approach, we defined a sample machine ontology containing all the components of our machine components taxonomy, and we used the complete annotated dataset, generated by IMDP, to train the NER models. We also defined a dictionary of some names, found in the annotated texts for the rule-based part. We can deduce that the performances of our approach are equal to the performances of the NER models since the rule-based resources are defined to achieve (biasedly in our PoC) an accuracy of 100%. However, in order to fully evaluate this approach, we need to have much more data than we currently use.

The evaluation results of the trained NER models are shown in the table 4.7. The iteration number is relative to the 5-steps adopted cross-validation approach as explained in section 4.5.1.5. The Precision, Recall, and F1 score are computed for each model for each of the 5 iterations, and the average row shows the global performances of the trained models. We can see that even with a relatively small dataset we can achieve good F1 scores. The NER model with BERT achieves an F1 score around 0.79, and CRF gets an F1 score of 0.89. While the best results were achieved by Spacy's NER model getting close to 0.93 for the F1 score. The standard deviation measured for all metrics was between 0.001 and 0.03 meaning that the models' performances are relatively stable even with the change of the training data sets.

We also evaluated the execution times of the different NER models, since our main execution target is an embedded industrial gateway. We found that the time needed to analyze a text and extract the sensitive entities is perfectly linear with the number of words in the text. Table 2 shows the time needed to analyze a sample of 1000 words. BERT showed exceptional performances with the ability to analyze a sample in less than 0.2ms on the laptop. However, it takes more than 9 seconds for the same task on RaspBerry PI which makes it not suitable for our case. CRF's model was also fast with 8ms on the laptop and 14ms on the RPI, while Spacy's NER model takes more execution time (than CRF) on both,

Table 4.7: Cross-validation evaluation of Spacy, CRF, and BERT NER models (P: Precision, R: Recall, F1: F1 score)

Iteration	SPACY NER			NER with CRF			NER with BERT		
	P	R	F1	P	R	F1	P	R	F1
1	0,932	0,914	0,923	0,937	0,852	0,891	0,814	0,783	0,798
2	0,931	0,934	0,932	0,945	0,866	0,902	0,79	0,815	0,803
3	0,926	0,934	0,930	0,919	0,874	0,896	0,782	0,834	0,807
4	0,923	0,943	0,933	0,936	0,901	0,918	0,749	0,816	0,781
5	0,928	0,917	0,922	0,881	0,817	0,846	0,74	0,781	0,76
<b>Average</b>	0,928	0,928	0,928	0,924	0,862	0,890	0,775	0,8058	0,7898

Table 4.8: Execution time of the NER models on laptop and Raspberry pi (ms/sample of 1000 words)

Model	Time on Laptop	Time on Raspberry Pi
<b>SPACY NER</b>	41	183
<b>NER with CRF</b>	8	14
<b>NER with BERT</b>	0.17	9239

the Raspberry Pi with 183ms and the laptop with 41ms, which also remains reasonable for an on-the-fly text analysis feature. Finally, once a machine component is detected, looking in the ontology if it is sensitive or not takes a negligible time ( $< 10^{-4}ms$ ).

## 4.6 Limitations and Improvement Areas

It is important to note that this contribution is the first step towards a fully automated maintenance data anonymization. Actually, data anonymization consists of two basic steps: sensitive data detection, and sensitive data masking or replacement. Paper II is providing a detailed analysis of data anonymization of maintenance reports.

With current PoC implementation, we conducted some tests on a limited set of maintenance reports, for which we created a simple machine ontology and found promising results. However, several limitations and areas of improvement are identified:

The first area of improvement is that we used a restrictive hypothesis as a basis of our approach. In fact, not all sensitive data are relative to machine components. As an example, the maintenance operator could describe a manufacturing step instead of the machine component, e.g. "The packets sealing has small holes". Also, the machine components might be sometimes described by their use such as using "the milk container" instead of the "liquid tank or reservoir". This can be improved by adopting digital tools to prepare maintenance reports like a UI based on a set of ontologies for taking input from the operators

by suggesting standard terms.

The second challenge is the nature of the language used for real maintenance reports, and that makes the automatic analysis awkward. In fact, maintenance operators do not provide literature texts, they usually use short informal texts, or even use street slang or urban vocabulary with frequent typos and missing punctuation. Similarly, the operator's vocabulary understanding may not be coherent with the normative definition, which often results in the use of non-standard abbreviations. Here again, the use of digital tools can be helpful to facilitate the operator to provide details as per normative definitions. Also, using real maintenance reports to train the NLP models could help in capturing the specificities of the maintenance reports' language.

Another area of improvement is the simplification of the data annotation process by decoupling the data annotation and the machine taxonomy. Some works like [88] showed that it is possible, for some NLP tasks, to train models on a corpus annotated with a taxonomy different from the one it is designed to output annotations for.

Finally, to have an accurate evaluation of our approach, we need, obviously, datasets with a considerable number of real maintenance reports containing sensitive data. Otherwise, the evaluation results make sense only for the used datasets.

## 4.7 Conclusion

In this chapter, we propose a new approach to avoid sensitive data disclosure during the maintenance data sharing activity. We showed the sensitive data types and nature in maintenance reports based on the interviews campaign, and that judging -with certitude- a piece of data to be sensitive or not, remains the role of a human expert such as a manufacturing engineer. In this approach, we aim to simplify the work of the human expert by identifying the potentially sensitive data in maintenance reports' content. For this purpose, we extend the SemKoRe ontology models to allow users to specify the items that can be considered as potentially sensitive. The evaluation of three NER models, shown promising results for on-premise sensitive data detection. We also identified some limitations and improvement areas to continue working on this topic.





# Chapter 5

## Embedded Semantic Querying Tool for Distributed Time Series Data

### Contents

---

<b>5.1</b>	<b>Proposed Solution</b>	<b>99</b>
5.1.1	Overview	99
5.1.2	Past Event Detection Features	100
<b>5.2</b>	<b>Implementation details</b>	<b>101</b>
5.2.1	Overview	101
5.2.2	DTSE Abstract layers	102
5.2.3	DTSE Query Language	102
5.2.4	DTSE query decoding	104
5.2.5	Distributed Times Series Engine's query processing	105
<b>5.3</b>	<b>Proof-of-concept Prototype &amp; Evaluation</b>	<b>106</b>
<b>5.4</b>	<b>Conclusion</b>	<b>107</b>

---

This chapter presents the embedded tool for distributed times series data using semantic criteria. It describes the contribution on Patent I and presents implementation details and performance measurements. It addresses the following research questions:

*How Semantic Web technologies can be used to enrich and to pull out insights from IIoT historical data? More particularly, how to deal with the diversity of data sources in IIoT? How to have an efficient mechanism for querying historical data stored on multiple IIoT low-end gateways?*

Today, several industrial IoT (IIoT) applications designed to monitor, benchmark, and discover trends reveals being very useful to gather valuable insights and to facilitate improvements in productivity and efficiency as well as other economic benefits [89]. Such applications, usually Cloud-based, rely on deployed gateways or systems to collect data from various sources such as sensors, actuators, or industrial machines. With advances in hardware technology coupled with low costs, gateways with storage capabilities are now quite common. Hence, easy-to-use solutions located closer to the devices are preferred but issues like device and protocol heterogeneity, multi-vendor solutions, and variety of data models make it a challenging task.

In our case, one of the short-term evolutions planned for the SemKoRe is to enable automatic failures detection and predictive maintenance features. Such features rely heavily on the analysis of time-series data. The project team adopted classical approaches for the management of time-series data, such as using relational databases with simple data structures like (id, timestamp, value). Such an approach is not efficient for data querying and was not considering the context of the measurements. However, it satisfies the technical requirements of the project team, such as simplicity, backward compatibility, and compatibility with the numerous existing IoT time-series databases.

With this context, we implemented a generic solution "DTSE" for IIoT gateways, to support new services by using locally stored time series data. It is important to note that our goal is not to disrupt and change the current IoT solutions, but rather aim to offer new services that can be easily integrated into the existing ecosystems. To tackle the heterogeneity issues, we need to decouple the problem space from the solutions space with the right level of abstraction. Hence, we rely on semantic web technologies to provide a high level of abstraction without focusing on protocols or communication mechanisms.

We believe that the existing works on RDF stream like [90,91] and temporal RDF [92–94] offer better support of semantics for time-series data querying and analysis. However, despite the disruption of the existing IoT ecosystems, these solutions are intended to be used on powerful platforms with triple stores capable of ingesting important streams of data, which is not practically adapted to platforms like IoT gateways with limited resources.

This contribution is an extension of the work described in [95], that was made by the

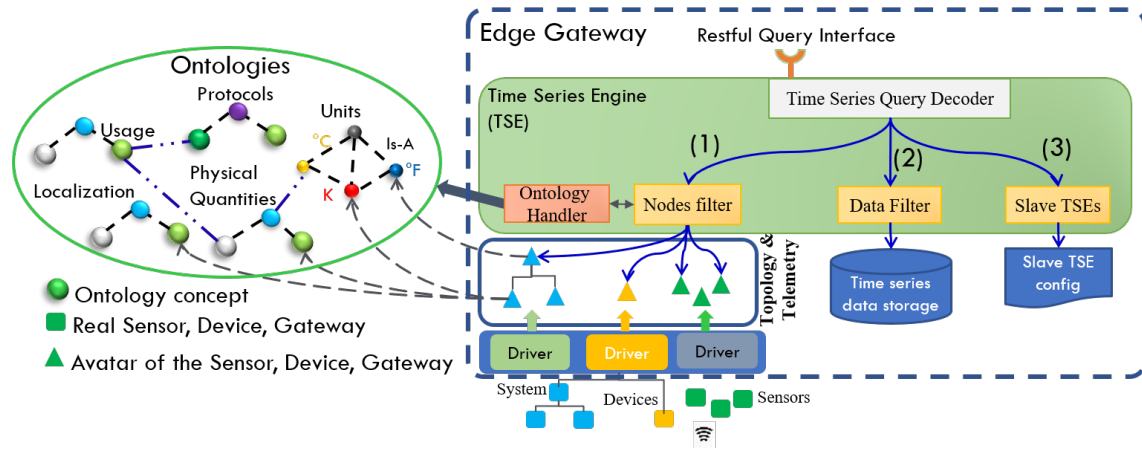


Figure 5.1: Overview of the Semantic Time Series Engine deployed on a gateway

author a couple of months before starting the Ph.D.

## 5.1 Proposed Solution

### 5.1.1 Overview

The proposed solution, called DTSE (for Distributed Time Series Engine), is shown in Figure 5.1. DTSE is a software module deployed in an edge gateway exposing annotated data of the connected devices. The semantic annotation process, described in [96], consists of linking concepts from Schneider-Electric’s common ontologies (e.g Protocols, Units, Usage) with the actual topology nodes exposing data by the gateway. Such annotations are useful and allow to identify the context of the collected data without a need to use the identifiers of the data sources. For example, the annotations : *usage:Temperature*, *protocol:Modbus*, and *location:Floor1*, describe the nature of the collected data as a temperature measurement, along with the used communication protocol and the location where data has been collected.

We consider that the collected data is stored in a local database on the gateway. The choice of the data storage is out of the scope of this work, our solution is designed to be flexible and to interact with any data storage used by the gateway.

DTSE offers a Restful interface that can be used by any application running locally on the gateway or externally. The Restful interface can mainly be used to execute time-series queries or to configure the different modules of the application: such as managing the common ontologies or configuring the slave DTSE instances. As shown in Figure 5.2, we can define a hierarchical structure between various DTSE instances through the specification of Master and Slave relationships.

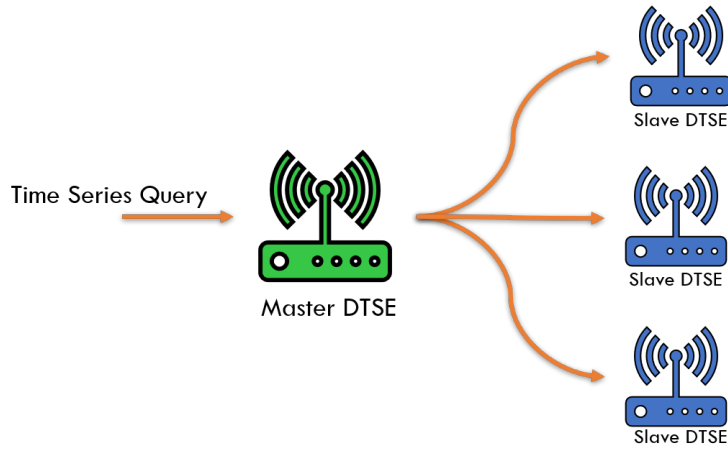


Figure 5.2: Distributed times series engines configuration

### 5.1.2 Past Event Detection Features

In addition to the time-series querying capability, DTSE proposes the following features:

- Distributed queries execution: A query can be executed on multiple DTSE instances running on different edge gateways. This feature is detailed in the next section.
- Past event detection: A past event is a set of predefined time and value conditions associated with the data collected by one device. DTSE allows to detect the time ranges during which an event has occurred and during which all the event's conditions are met. We can also filter the detected events by specifying the minimum occurrence duration without interruption, e.g.: *"High temperature for more than 1 hour during the weekends"*.
- Independent events detection: It is also possible to detect multiple independent events occurring on one or several devices using one single query, e.g. *"Machine1 is working for more than 8 hours, Machine2 is working for more than 12 hours"*. This type of query can be useful to monitor several assets with one single query.
- Parallel events detection: DTSE proposes also a mechanism to detect events occurring simultaneously on multiple devices, e.g. *"Machine1 is working while the temperature is high (>30degree) during more than 15 minutes."*

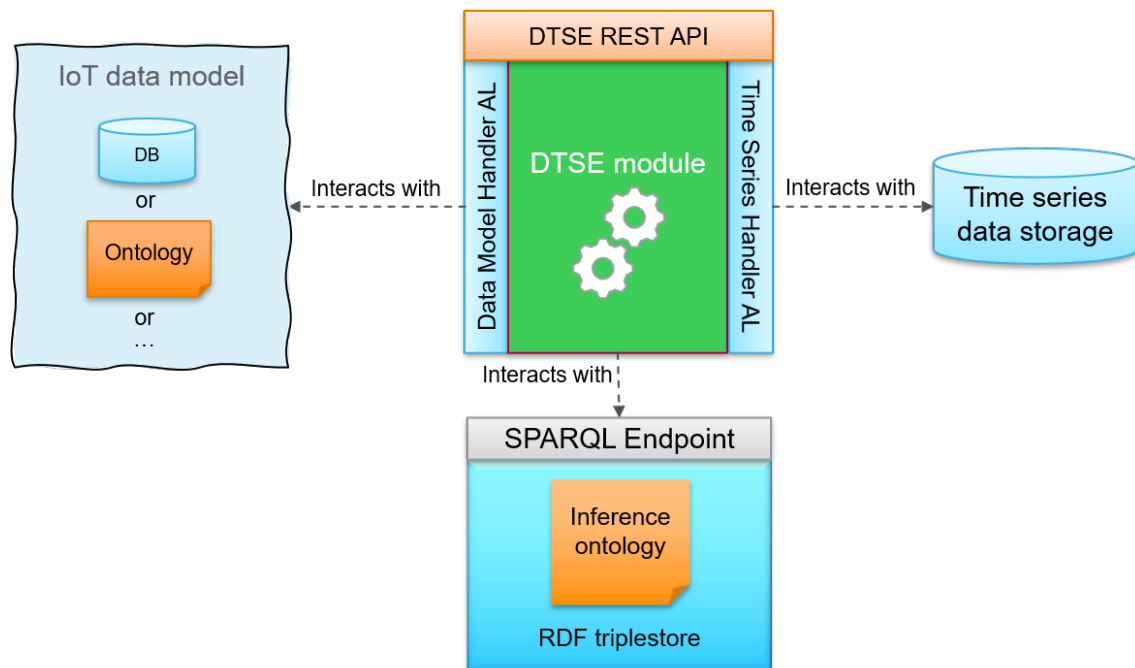


Figure 5.3: Components view of the DTSE module and interactions with other services

## 5.2 Implementation details

### 5.2.1 Overview

Figure 5.3 shows the component's view of the DTSE module, and how it interacts with the other components running on the IoT gateway. In fact, it needs to interact with the services that are usually deployed in IoT solutions, such as:

- The IoT data model service: This service manages the topology of all IoT nodes (devices, sensors, actuators) controlled or managed by the gateway. Several techniques are used by the IoT software developers to implement this service, such as using databases, ontologies, configuration files, ...etc.
- The time-series data storage: we assume that a gateway dealing with time-series data may probably have its own data store, and probably with already available data.
- Inference ontology: This optional module is only required when the inference feature is enabled on DTSE. Providing a SPARQL Endpoint, it allows DTSE to execute some predefined SPARQL queries in order to infer additional knowledge about the topology nodes' annotations.

## 5.2.2 DTSE Abstract layers

For the sake of genericity, we designed DTSE to be independent of any of the IoT data model services or data storage modules, and we impose almost no requirements on the implementation of these services. Instead, we provide two simple interfaces that need to be customized by the integrator: Data Model Interface, and Time Series Interface. DTSE uses these interfaces as abstraction layers to interact with the required modules without being intrusive.

### 5.2.2.1 Data Model Interface

The data model interface is proposed to allow DTSE to get the details and meta-data of the topology nodes. On one hand, we require a limited set of details for each node, such as the identifier, node's nature (Device, sensor, actuator), time series ID if exists, name, and the parent node to have the hierarchical view of the topology. On the other hand, the meta-data are the nodes annotations using the concepts from ontologies (e.g. Schneider Common ontologies in our case). Note that we have no requirements on the ontologies used for the nodes annotations, all that DTSE needs are pairs of "*Key:Value*" that describe the context of the topology nodes, e.g. "usage:Temperature", "location:Floor1". The source code of this interface is implemented in C language and is hosted in <https://github.com/hicham0100/DTSE/blob/main/dmapi.h>. The integrator must implement all the defined functions and produce input in the expected data structures.

### 5.2.2.2 Time Series Interface

This interface allows DTSE to interact with the time-series data storage without being too intrusive. It defines various functions that need to be implemented by the integrator to:

- Query the data of existing time-series data by providing the corresponding ID.
- Add new entries (if allowed by the integrator) to a time-series data.
- Make searches on time-series data based on simple value and time conditions.

The source code of this interface is implemented in C language and is available at [https://github.com/hicham0100/DTSE/blob/main/TS\\_api.h](https://github.com/hicham0100/DTSE/blob/main/TS_api.h).

## 5.2.3 DTSE Query Language

The DTSE' queries rely on Schneider's domain-specific query language<sup>1</sup> and extend the basic query language used in [97] to support time-series data. It takes into account a

---

<sup>1</sup>[https://github.com/hicham0100/DTSE/blob/main/Query\\_Grammar](https://github.com/hicham0100/DTSE/blob/main/Query_Grammar)

<p><b>Search Times</b>      TAGS_FILTER          [with            ATTRIBUTES_FILTER]          [where          VALUE_FILTER]          [during        DURATION_FILTER]          [when          TIME_FILTER]          [TIME_RANGE_FILTER]          [union        NESTED_TIMES_QUERY]          [run on        SLAVE_DTSE_LIST]</p>	<p><b>COMMAND Values</b>      TAGS_FILTER          [with            ATTRIBUTES_FILTER]          [where          VALUE_FILTER]          [when          TIME_FILTER]          [TIME_RANGE_FILTER]          [group by        GROUP_BY_FIELD]          [run on        SLAVE_DTSE_LIST]</p>
--	--

(a) DTSE time ranges queries structure

(b) DTSE values queries structure

Figure 5.4: Structures of the DTSE queries

combination of tags and expressions to filter and aggregate data using several dimensions such as: semantic context, static attributes, measurements values, time components and time ranges as shown in Table 5.1 and Figure 5.4. Two major types of queries are supported by DTSE:

Table 5.1: Description of the DTSE queries fields

Field	Description	Example
<i>COMMAND</i>	Specifies the search command or the aggregation function that will be applied on the result	search, min, max, sum, avg
<i>TAGS_FILTER</i>	Filters the topology nodes based on semantic tags conditions	usage:Temperature or protocol:Modbus
<i>VALUE_FILTER</i>	Filters the topology nodes based on static attributes such as: unit, name, id	<b>with</b> id=="sensor123" or name != "Humidity"
<i>TIME_FILTER</i>	Filters the timeseries data of the previously selected nodes based on time conditions	<b>when</b> hours >= 8 and hours <= 18 and wday == 1
<i>DURATION_FILTER</i>	filters the time ranges by the uninterrupted duration of occurrence of the query conditions.	<b>during</b> time >= 00:10:00 and time <= 00:20:00
<i>TIME_RANGE_FILTER</i>	Restricts the analyzed timeseries data to specific time ranges	<b>from</b> 2021-12-01 00:00:00 <b>to</b> 2021-12-02 00:00:00
<i>GROUP BY FIELD</i>	Defines the dimension for the aggregation of data	<b>group by</b> hours
<i>SLAVE DTSE LIST</i>	Specifies the list of DTSE slave on which the query will be executed in parallel	<b>run on</b> DTSE_5, DTSE_7

- Values Queries: Figure 5.4b shows the Values Queries structure. This is the basic



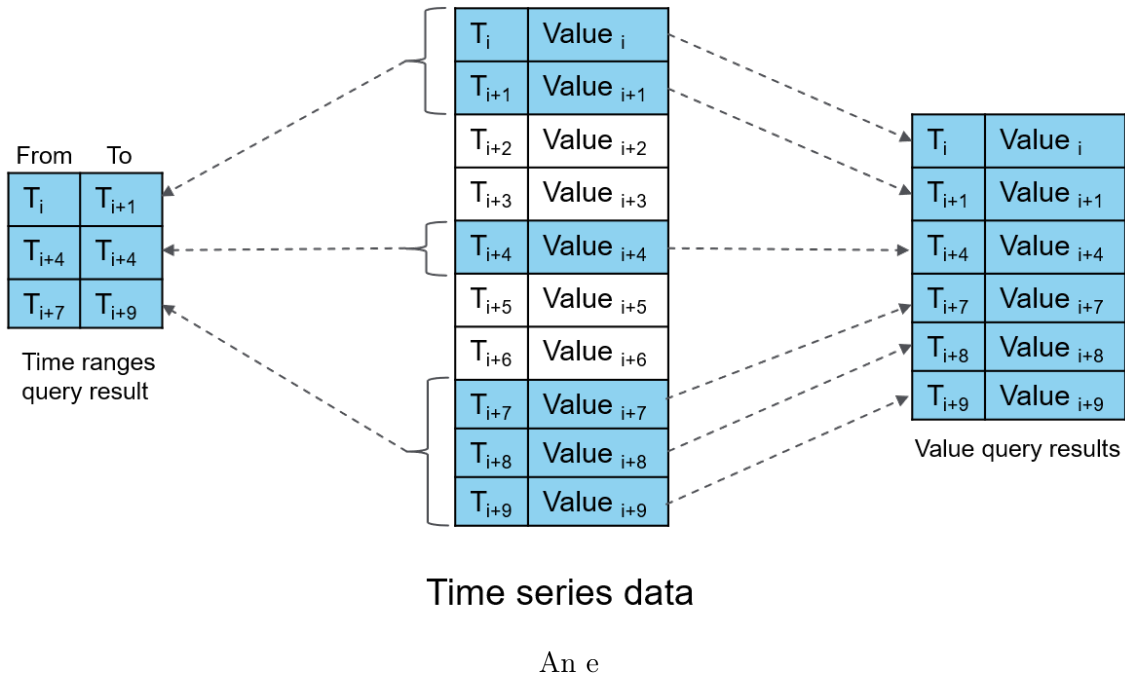


Figure 5.5: Example highlighting the differences between the Values Queries and the Time ranges queries

feature provided by most time-series databases. It allows searching and aggregating the historical values of a sensor node. Many filtering options are available such as Semantic Tags, nodes attributes, value, time components, and also time ranges.

- **Time Ranges Queries:** The structure of these queries is depicted by Figure 5.4a. These queries are intended to search the time ranges within which the list of query conditions is met. For every returned time range, we are sure that all conditions are met without interruption. Despite its important usage, to our knowledge, no existing embedded TSDBMS is supporting such a feature. The Figure 5.5 highlights the differences between the values queries and the time ranges queries.

#### 5.2.4 DTSE query decoding

For the query decoding, we rely on the ANTLR (ANother Tool for Language Recognition) framework<sup>2</sup>. ANTLR allows to parse the input query and to generate an Abstract Syntax Tree (AST) according to the defined query grammar. Figure 5.6 shows an example of AST generated for the query *"search times usage:Temperature where value < 25 and value >= 20 during time > 00:10:00"*. The resulting AST allows to distinguish mainly 3 parts in the

<sup>2</sup><https://www.antlr3.org/>

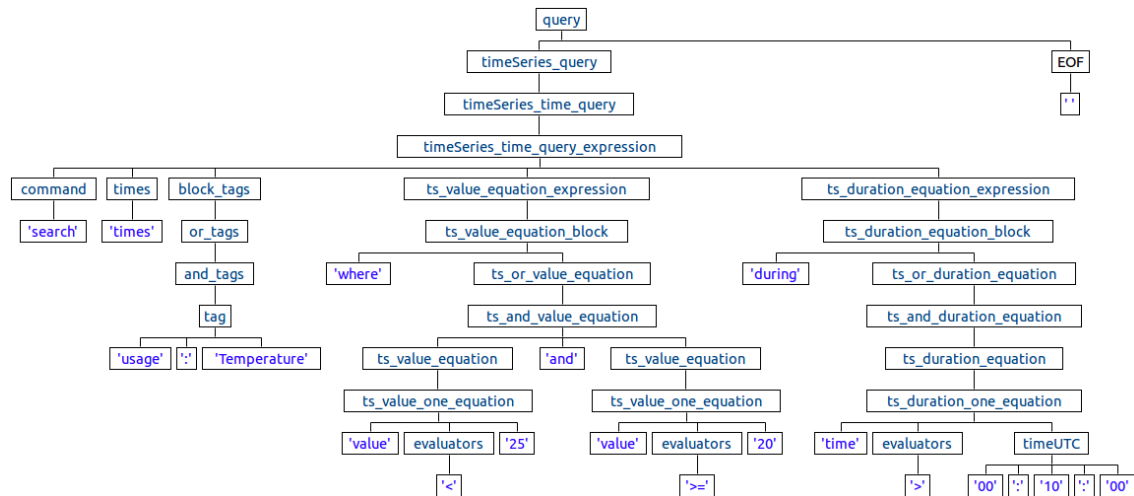


Figure 5.6: Example of Abstract Syntax Tree (AST) generated for a query example query:

- The topology nodes conditions that allow filtering the topology nodes based on the provided criteria,
- The time series conditions that provide the criteria to search and filter the time series data.
- The last part serves to know if the query will be executed only locally or will be distributed and executed in parallel on the designated slave DTSE gateways.

### 5.2.5 Distributed Times Series Engine’s query processing

The Figure 5.7 depicts the DTSE query processing diagram. The queries are first parsed and decoded to identify the different query parts, described above, to:

1. Filter the topology nodes using the semantic criteria defined in the query Fig. 5.7(1). These criteria are composed of one or more semantic annotations, e.g.: (*usage:Temperature* or *usage:Humidity*) and *Location:Room123*. The semantic criteria can be enriched by enabling the inference feature using the "@" decoration symbol, e.g. *@Location:Grenoble* returns all nodes located in Grenoble or any sub-location of Grenoble. The inference mechanism provides a higher level of abstraction and relies on the used ontologies (e.g. Location) to identify the inferred annotations.
2. Analyze the historical data collected by the nodes found during the first step Fig. 5.7(2). Various criteria are supported such as:

- Data values conditions: used to filter data by values e.g. *"value >38 and value <40"*
  - Time conditions: including conditions on the timestamps and time components such as year, month, day, weekday, hours, minutes (e.g. *"month == 3 and day == 10 and (hours >= 8 and hours < 18)"*)
  - Duration criteria: specifying the conditions on the duration of the events during which the values and time conditions are met (e.g. *"during time > 00:05:00 and time <= 01:00:00"*).
  - The time ranges conditions: limiting the scope of the data analysis to specific time periods (e.g. *"from 2021-12-03 00:00:00 to 2021-12-31 23:59:59"*)
3. Check the distributed execution criteria Fig.5.7(3), two possible scenarios are supported:
- Scenario 1 - local query: the query is executed locally and the results are immediately returned.
  - Scenario 2 - distributed query: the query is executed locally and is in parallel transferred (without distribution criteria) to the specified slave DTSEs. Each slave DTSE executes the received query locally on its own topology and time-series data, then returns back the results to the requesting DTSE. On the other side, we collect the results from all designated slave DTSEs, merge them with the local results before being returned as final results.

### 5.3 Proof-of-concept Prototype & Evaluation

We implemented a prototype of DTSE in C and we interfaced it with SQLite as it is a popular data storage for low-end devices.

To evaluate our implementation, we used a Raspberry Pi 2 board. We simulated two temperature and humidity sensors and generated a data set of 20 million records (10 million per sensor). The generated data spread over three years starting from 2014-06-01.

Four benchmark queries were used to analyze the generated data sets (Table 5.2). They allow to analyze one time series against the other and to mainly detect parallel events. And to compare DTSE performances to another existing solution, we contacted the Machbase<sup>3</sup> team, which provides the embedded Machbase DBMS dedicated to time series, and provided them our evaluation queries. Since Machbase is SQL-based, no single query was able to

---

<sup>3</sup><https://www.machbase.com/>

Table 5.2: DSTE evaluation queries

No	Query
$Q_1$	Get time ranges when humidity $> 65$ and temperature $> 25$
$Q_2$	Get temperature when humidity $> 65$ for more than 10 minutes
$Q_3$	Get temperature when humidity $> 50$ during the weekends
$Q_4$	Get the maximum humidity during June when temperature $\geq 30$

handle alone our benchmark queries. The Machbase team took care of creating an optimized combination of queries and software to achieve the same goals.

The chart in Figure 5.8 shows the queries execution times using DTSE and Machbase. DSTE requires on average less than 7 seconds to execute each of the benchmark queries. While Machbase takes between 9 seconds to more than 6 minutes depending on the complexity of queries.

The evaluation of distributed query execution showed similar performances for local executions on each DTSE instance. Afterward, the Master DTSE needed additional time to receive and process the results returned by the slave DTSE instances. This extra time, estimated in our case between 1 to 2 seconds per slave DTSE instance, depends on the size of returned data, the latency of the HTTP protocol, and the communication channel bandwidth.

## 5.4 Conclusion

This chapter presented the work on the embedded distributed time series solution. It uses semantic annotations and relies on ontologies to provide a high level of abstraction. Features such as past event detection, missing in existing time series solutions, revealed a high potential for IIoT monitoring or benchmark solutions. The prototype showed reasonable performances on low-end gateways, making it a potential candidate for data analysis in the SemKoRe ecosystem.

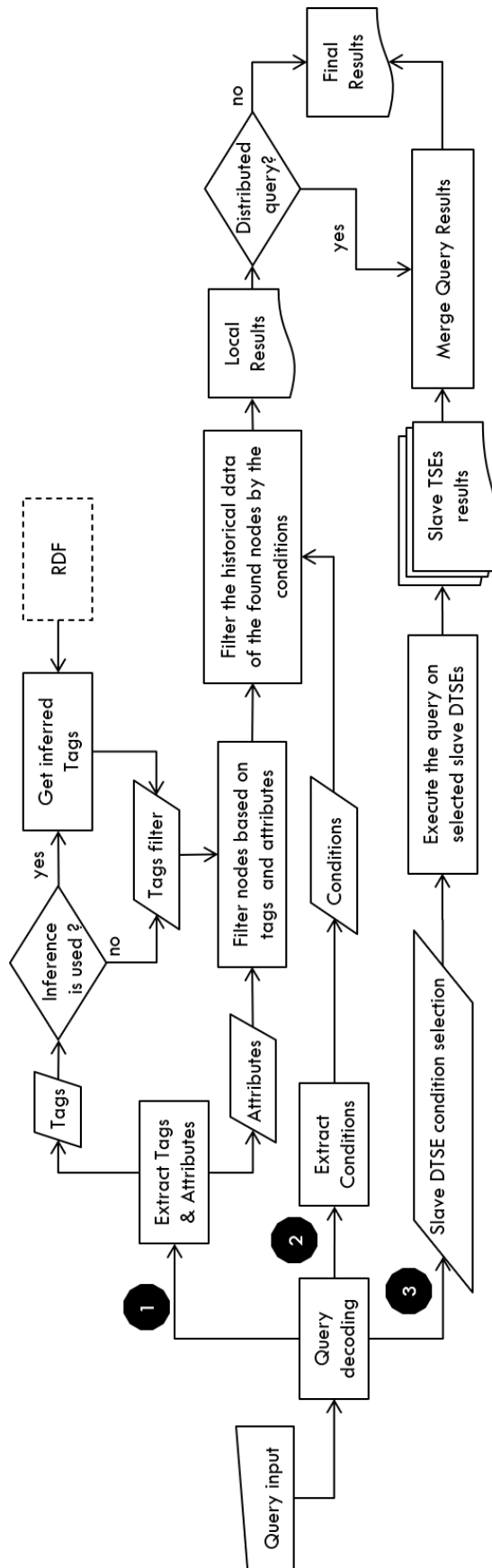


Figure 5.7: Distributed Times Series Engine's query processing diagram

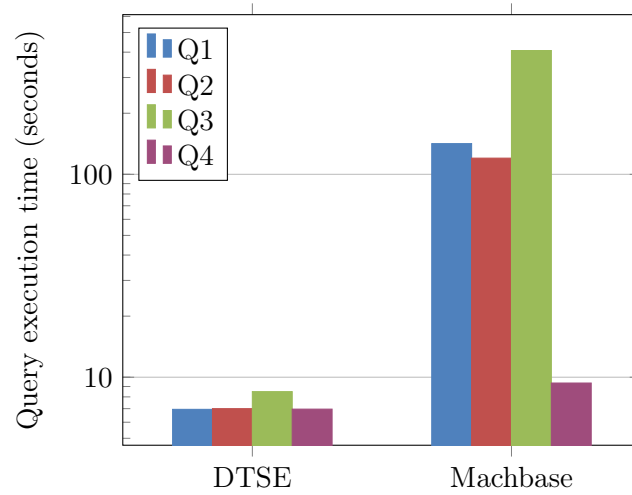


Figure 5.8: Execution time comparison of DTSE and Machbase



# Consistent Knowledge Graphs Manipulation for Semantic Applications

## Contents

---

<b>6.1</b>	<b>Runtime API for consistent Ontology Instantiation . . . . .</b>	<b>112</b>
6.1.1	Overview . . . . .	112
6.1.2	Data Manipulation Features . . . . .	114
<b>6.2</b>	<b>Technical details . . . . .</b>	<b>115</b>
6.2.1	Data Querying interfaces . . . . .	116
6.2.2	Data update interface . . . . .	118
6.2.3	Consistent KG manipulation . . . . .	118
6.2.4	Consistency Checking Alternative . . . . .	120
<b>6.3</b>	<b>Conclusion . . . . .</b>	<b>120</b>

---



This chapter presents the Knowledge Graphs manipulation tool destined to software developers in order to simplify the development of Semantic Applications. This contribution is the subject of Patent II. It addresses the following research question:

*How can Knowledge Graphs (KG) consuming and manipulation by application developers be facilitated? How to guarantee the consistency of the manipulated KGs? How can ontology engineers be involved to enhance the developers' experience?*

During the last decade, Knowledge Graphs have proved their effectiveness for structuring data and knowledge in various domains. They became the most important denominator in large IT companies such as Google, Microsoft, Facebook, and Amazon. These successful experiences pushed several companies, including Schneider-Electric, to adopt the Knowledge Graphs as an efficient way for data integration, unification, analytics, and sharing.

However, software developers find it difficult to manipulate Knowledge Graphs and to develop ontology-based applications. This complexity can be explained by two main reasons. The first reason is that most software developers lack awareness and knowledge of Semantic technologies. The second reason is the complexity of the ontology models that are usually defined to cover a large domain or activity. In fact, many W3C recommendations and well-known ontologies have complex mechanisms to describe data.

Since 2008, several works in the literature propose various tools and APIs to simplify the interactions with Knowledge Graphs. A recent survey [98] presents comprehensive state-of-the-art of KG read-only tools like Pubby [99], Puelia [100], ELDA [101], LODI [102] and the recent Walder [103], and read-write tools like Open Semantic Framework [104], Trellis [105], BASIL [106], RAMOSE [107]. Their analysis confirms our observations that these tools require developers to be familiar with the ontology to design and configure APIs. They also need proficiency in SPARQL and GraphQL.

In this work, we address the needs of typical software developers who want to build new applications and services using Knowledge Graphs without deep knowledge of semantic web technologies. We propose a Restful API, that requires minimal configurations and allows to manipulate KG. It allows also to guarantee the consistency of the models without needing extra ontology tools like a reasoner. Our first goal from this contribution is to simplify and accelerate the development of SemKoRe services. Thus, the proposed solution is designed to be used on the different IoT layers: Edge, Fog, and Cloud.

## 6.1 Runtime API for consistent Ontology Instantiation

### 6.1.1 Overview

Figure 6.1 presents an overview of our API for KG manipulation. It allows manipulating Knowledge Graphs through a Rest interface and by using JSON requests. As a unique

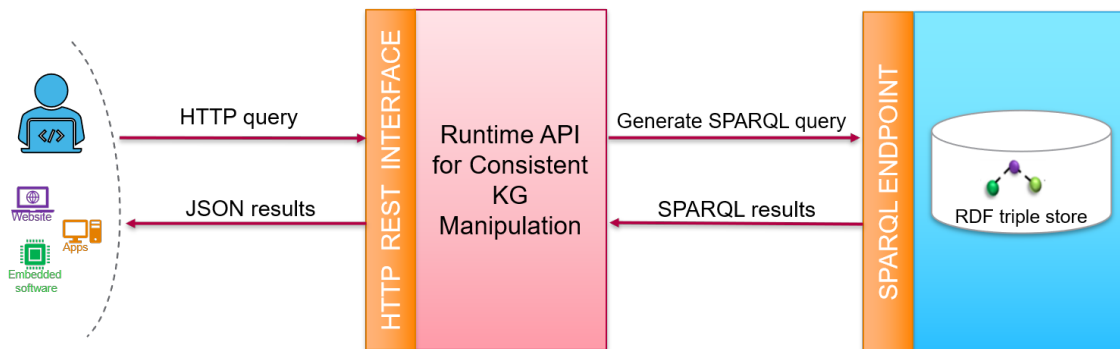


Figure 6.1: Runtime API for Consistent KG Manipulation

prerequisite, the Knowledge Graphs must be stored in an RDF triple store exposing a SPARQL Endpoint. Also, the only required configuration for the API is the URL of the SPARQL Endpoint.

The proposed API can be used by software developers or by applications such as websites, embedded software, or any application type. This solution is designed to be used on various platforms: Cloud, laptop, and low-end edge gateways. We use the term "*Runtime*" since the API can connect to any RDF triple store and immediately adapt to the stored KG, without needing to add configurations or to generate custom APIs. The term "*Consistent*" is used because the API guarantees the consistency of the KG after each modification or update.

It is important to note that, in this approach, we consider that the ontology description is exhaustive and that it covers all the needs and expected usages. Hence, we clearly restrict the KGs manipulation to the definition of the ontology, and we ignore the "*open-world*" assumption in which the absence of a particular statement doesn't mean it is false or impossible.

Also, this API is a simple proof of concept and is, in its current form, not compliant with the Linked Data Platform standard (LDP) [108], as recommended by W3C, for two reasons:

1. The user/consumer of LDP services needs to be familiar with Semantic Web technologies, especially for the update operations that require a good understanding of ontology concepts, JSON-LD/Turtle or SPARQL.
2. LDP allows the user to freely manipulate the managed KG, without being restricted to the defined ontology models, which is an undesired scenario in our case.

Finally, the choice of JSON format is made for simplicity in our PoC, and other formats could as well be used such as SPARQL-JSON or JSON-LD.

### 6.1.2 Data Manipulation Features

The Figure 6.2 shows the flow of KG manipulation through the proposed API. Three types of queries are supported for data manipulation: data querying, data update, and parameterized SPARQL Queries. For all query types, a SPARQL query is generated and executed over the KG in the triple store. The SPARQL query results are then converted into JSON format before being returned as a result to the caller.

#### 6.1.2.1 Data Querying Feature

Regarding the data querying feature, 4 possible queries are proposed:

- Get the list of classes
- Get the structure of a specific class
- Get all the instances/individuals of a specific class
- Get the details of a specific instance

These simple queries showed to be sufficient to query most of the data in a KG. They also remind the structure of relational databases. Which makes them easily assimilated by the developers used to use RDBMS.

#### 6.1.2.2 Data Update Feature

The data update feature offers 3 querying possibilities:

- Create a new instance of a specific class.
- Update the details of a specific instance.
- Delete an instance and all its details.

During the data update, the validity of the received data is checked against the ontology model definition before applying any change in the RDF triple store. The next section describes the model consistency checking process in detail.

#### 6.1.2.3 Parametrized SPARQL Queries

To bypass the limitations of the data querying and data update features, the API supports the use of configurable queries or what we call Parametrized SPARQL Queries (PSQ). The PSQs are templates of queries using ad-hoc decorators of which values may be injected. This feature requires prior configuration and allows KG consumers to run complex (pre-defined)

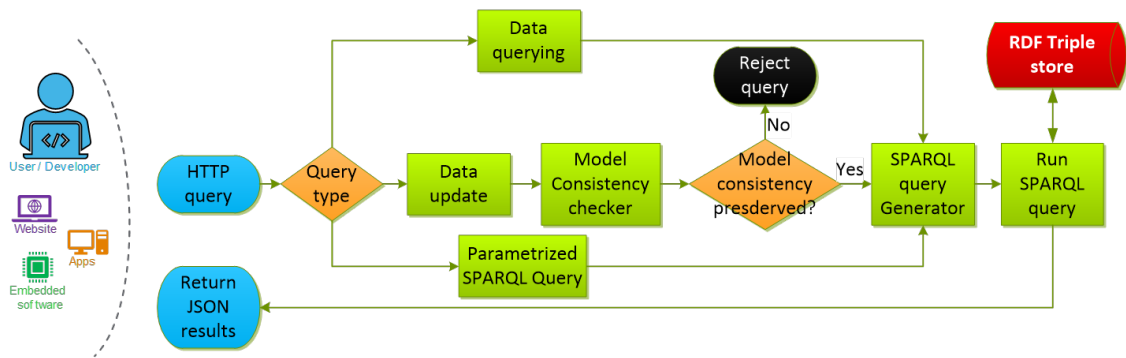


Figure 6.2: KG Manipulation Flow

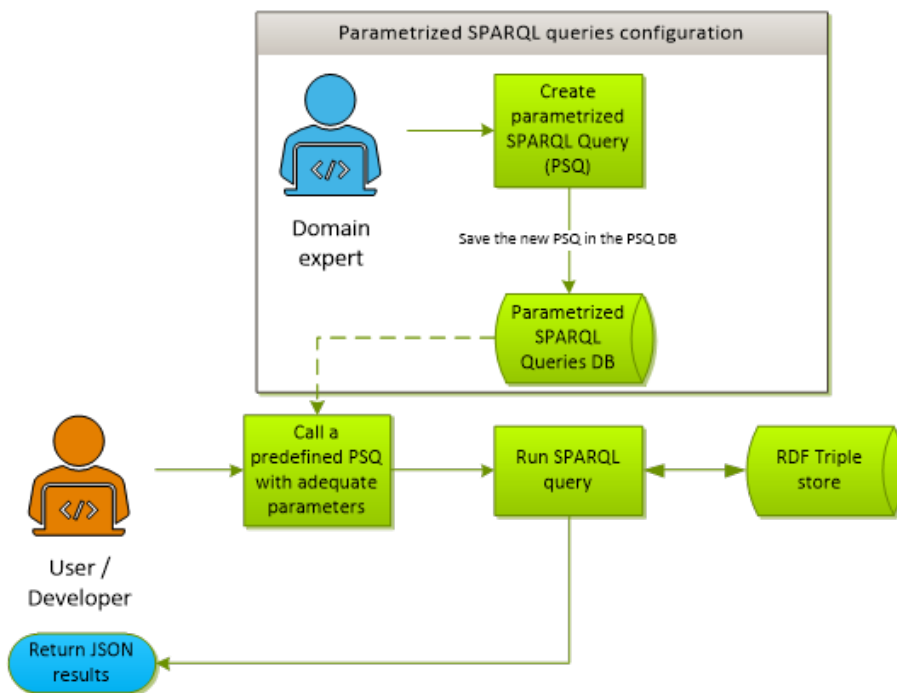


Figure 6.3: Parametrized SPARQL Query flowchart

queries without any knowledge of SPARQL. As shown in Figure 6.3, a domain expert defines and configures the PSQ judged necessary for a specific use or project. Then, the application developer can execute the configured PSQs by providing the necessary parameters.

## 6.2 Technical details

The prototype of the runtime API for consistent ontology instantiation is implemented using Node-Red. This section provides the implementation details of the supported features by our API.

### 6.2.1 Data Querying interfaces

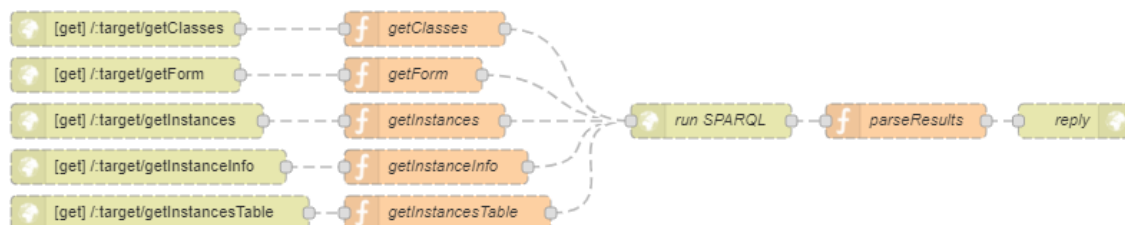


Figure 6.4: Node-Red flow implementation for the data querying feature

The Figure 6.4 shows the Node-Red flow for the data querying interfaces. The same processing steps are repeated for each interface:

1. The received request is forwarded to a dedicated node that checks request's format and extracts the request parameters. These parameters will be used to generate the corresponding SPARQL query.
2. The generated SPARQL query is then executed on the pre-configured triple store through a SPARQL endpoint.
3. The SPARQL query results are then parsed and converted to JSON before being returned to the caller.

#### 6.2.1.1 Get the list of classes

This interface retrieves the list of classes defined in an ontology. Each class is described by three attributes:

- The class's identifier, which is the class's URI
- The class's label (if defined)
- The identifier of the class's parent (if exists), allows having a view on the classes' hierarchy in the ontology.

We can also restrict the classes to a specific "namespace" in case of the use of multiple ontologies in the same triple store.

#### 6.2.1.2 Get the structure of a class

This interface allows the extraction of information about a class's structure. We adopted an object-oriented view, and we consider the class description as a template or prototype from which the class's instances or objects are created.

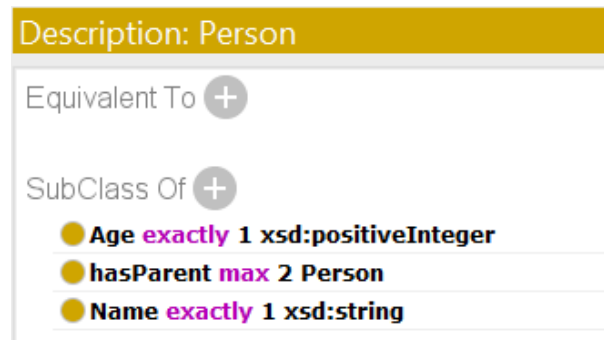


Figure 6.5: Example of a class description

As an example, the description of the class "Person" shown in the Figure 6.5 is translated by the API as Every instance of the class "Person" has *exactly* three attributes:

- Age: with exactly one positive integer value.
- Name: with exactly one string value.
- hasParent: links the current instance to at most 2 existing "Person" instances as parents.

As discussed previously, we restrict the manipulation of the KGs to the ontology definitions and, thus, we ignore the *open-world* assumption.

We also consider the hierarchy between classes in such a way every *subclass* inherits the definition (attributes) of its *superclasses*. For example, the class "Man" as a subclass of "Person", has the attributes defined in the class "Man" itself, as well as the attributes defined in the class "Person".

The returned results of this interface contain the list of all attributes (direct and inherited), the label and value type of each attribute, and the "arity" which is the number of supported values for each attribute. When an attribute is defined as the relation between two instances via an object property (e.g. Person:hasParent), it is possible to include, in the result, all the possible values of that attribute, e.g. all the instances of the class "Person" from the example above.

### 6.2.1.3 Get the instances of a class

This interface simply returns the list of instances of a class given as a parameter. The returned result contains the identifier (URI) and the labels of each instance. The interface "getInstancesTable" provides the same result but includes the details of all returned instances.

### 6.2.1.4 Get the details of an instance

This interface gathers the details/attributes of each instance in the ontology. The returned result contains a list of attributes names, their values, and the labels of their values (if applied). Note that only the attributes defined in the description of the instance's class (as described in section 6.2.1.2) are considered.

### 6.2.2 Data update interface



Figure 6.6: Node-Red flow implementation for the data update features

The figure 6.6 shows the flow for the data update feature. Three types of requests are supported:

- Create a new instance of a specific class: this requires providing the values of all expected attributes as defined in the Section 6.2.1.2.
- Update the details of a specific instance: this operation deletes the previous instance and creates a new one with the same URI.
- Delete an instance and all its details. A cascade deletion mechanism is adopted when the deleted instance is referenced by other individuals in the KG.

Before creating new instances, the API checks the conformity of the provided parameters based on the expected structure (Section 6.2.1.2). This step is called "Consistency checking", and it is described in detail in the next section.

### 6.2.3 Consistent KG manipulation

One of the major challenges encountered when a KG is manipulated by non-expert users is to guarantee the consistency of the KG and compliance with the ontology definition. In fact, by nature, it is possible to extend a KG in an RDF triple store with new axioms without necessarily respecting the description of the ontology behind it. Usually, some additional tools such as reasoners, are necessary to check the consistency of ontologies and KG after changes are applied. There exist several ontology reasoners supported by various RDF triple stores. Unfortunately, all these reasoners are designed to be used on high-end platforms and are too complex to be used on embedded systems.

To tackle this issue, the proposed API provides a built-in mechanism to guarantee consistency during the manipulation of KG. This mechanism is solicited only for update

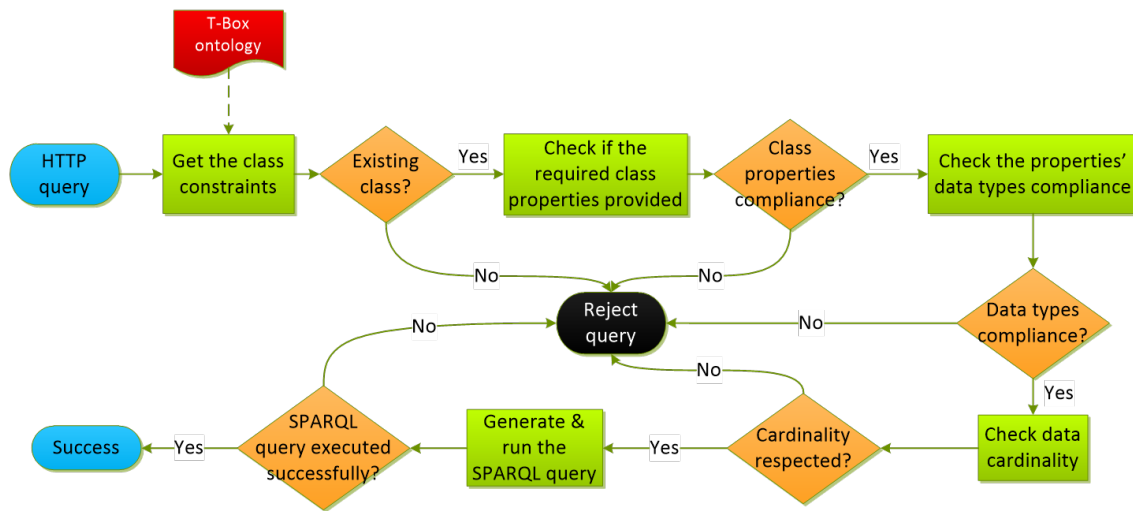


Figure 6.7: Diagram flow of consistent creation of new class instances

queries. It consists of checking the compliance of the new axioms/data with the ontology definition before applying any change in the RDF triple store. Figure 6.7 shows the diagram flow of the consistency checking process during the creation of a new class instance.

When a new class instance request is received, the consistency checking mechanism checks the compliance for three main aspects: Class description, Value restrictions, and Cardinality restrictions.

### 6.2.3.1 Class Description Compliance

To create a new class instance using the proposed API, a JSON object with key-value pairs is needed. Each key-value pair reflects a property (object or data) with the corresponding value. It is possible to get the list of properties needed by a specific class through the *Get Class Structure* query.

For each received update request, all the provided attributes are checked to be part of the class description in the ontology as described in the section 6.2.1.2. If a non-expected attribute is provided, the request is rejected. Otherwise, the consistency checking process continues.

### 6.2.3.2 Value Restrictions Checking

In the W3C OWL reference [109], a value restriction is used to enforce restrictions on the range of a property when applied to a particular class description. Value restrictions can be applied to data properties, for which the value is literal, and object properties, for which the value is an individual. We handle each type of property differently:

- Data properties: we make sure that the data type of the provided value is correct as



per the defined restrictions; and

- Object properties: The provided value is checked to correspond to the URI (ID) of an existing individual, and that it is an instance of the expected class.

Note that the existential quantifier *owl:someValuesFrom* is ignored by our API. In fact, it defines a restriction that is applied to at least one value, which means that the property could have other values types without any restrictions.

### 6.2.3.3 Cardinality Restrictions Checking

In the W3C OWL reference [109], a cardinality restriction restricts the (min, max, or exact) number of values a data or object property can have. To validate the cardinality restrictions, the number of values provided for each property is verified.

### 6.2.4 Consistency Checking Alternative

It is important to mention that adopting the *closed-world* assumption pushes us to interpret the classes, data, and object properties' definitions as validation schema for the KG manipulation, while they are only inference schemas. An alternative, that is recommended by W3C, is to use the Shapes Constraint Language (SHACL) [110] as a validation schema, it allows to define explicitly a large set of conditions to restrict the manipulation of the KGs. We didn't adopt SHACL in our work because it was not supported, at that time, by our main triple store (GraphDB), but we strongly recommend adopting it for the KG consistency checking.

## 6.3 Conclusion

This chapter presents the Rest API solution proposed to simplify the instantiation of ontologies and manipulation of Knowledge Graphs by software developers. The API is destined to be used on the different IoT layers: Cloud, Fog, and Edge, and it proposes various ways to consume and extend the Knowledge Graphs in an RDF triple store. A focus is made on the consistency preserving of the ontology model during the manipulation of the KG. Finally, the proposed API was primarily destined for the development of new SemKoRe services but reveals being useful for several other semantic-based projects.

# Conclusion and Future Work

## Contents

---

7.1	Summary . . . . .	122
7.2	Future Work . . . . .	123

---

This chapter provides a summary of the thesis along with some work items for the future potential that will aid in extending the work done in this thesis.

## 7.1 Summary

Since its birth with the first industrial revolution (18th century), industrial maintenance was considered a marginal activity for more than two centuries. Now, With the advent of Industry 4.0, organizations realize its impact on their efficiency and bottom line, making it a backbone of business operations. Whereas traditional practices still persist in the industrial landscape. Maintenance knowledge sharing between industries is proving to be a potential solution to significantly optimize the maintenance activity and improve the processes' efficiency.

This thesis proposes a knowledge graph-based approach for maintenance knowledge sharing. It aims to enhance the maintenance process for the customers of Machine Builder OEMs. The idea consists of collecting and sharing, among the OEM customers, the maintenance data generated for different machines owned by many customers in different locations and in different business segments. The proposed approach helps reduce maintenance costs by sharing experiences and by promoting the best maintenance practices between OEM customers. Based on this early work, many customers of Schneider-Electric showed an interest in using our approach to enhance their industrial maintenance processes. In total five contributions were made in this thesis.

The first contribution is a field study through an interview campaign with domain experts from various industrial segments. This survey allowed us to discover the current practices in terms of industrial maintenance, as well as the problems confronted by these experts in their daily routine. It also showed a real interest in our approach for maintenance knowledge sharing and helped to identify the challenges that need to be tackled in order to achieve our goals.

The second contribution is "SemKoRe", a technical solution for maintenance knowledge sharing. Relying on Knowledge Graphs, SemKoRe allows to easily capture various maintenance aspects relative to machine failures. The collected knowledge is then shared with several stakeholders thanks to a flexible IoT architecture that adapts to the varied customers' requirements. For the proof-of-concept prototype, several services have been developed for the different IoT layers: Cloud, Fog, and Edge. It was successfully demonstrated to the MSol LoB team and has proved the feasibility of the proposed approach.

The third contribution is a novel approach for sensitive data detection in maintenance reports. It was considered by the experts as the make or break point for the SemKoRe solution. Our approach relies on Natural Language Processing techniques, combined with

semantic web technologies in order to meet the varied customers' needs and maintenance reports' specificities. A proof-of-concept prototype was implemented considering Edge gateways as the main platform target. However, the lack of maintenance reports datasets was the major roadblock towards the evaluation of our approach. Hence, we implemented and used a custom tool for collaborative data corpus construction, and that supports the data collection, structuring, and annotation.

In the fourth contribution, we designed a distributed semantic engine for time series data for low-end edge gateways. It shows that the use of semantic concepts provides a high level of abstraction and simplifies the interaction with time-series databases. Also, the distributed queries execution enhances the user experience with the possibility to simultaneously query several times series databases in several edge gateways. The performance evaluations were conclusive, and show good efficiency compared to the existing solutions.

In the fifth contribution, we propose a REST API for consistent ontology instantiation. The proposed API provides an easy way for the manipulation of knowledge graphs data without requiring any Semantic Web background. Our main target is the software developers having no knowledge about the Semantic Web, and that want to develop Semantic-based applications (e.g. SemKoRe services). Also, the API constrains the possible interactions with Knowledge Graphs in order to guarantee the permanent consistency of the manipulated models.

## 7.2 Future Work

We have identified several avenues as the future potential to continue the work of this thesis.

Currently, in the SemKoRe approach, only one triplestore is used in the cloud to collect the data of all customers. However, this could become an issue for scalability and data privacy. A potential solution may be to have separate triplestores tenants for each customer and then use a common triplestore instance to collect and aggregate data from the other customers' instances. Managing these tenants and synchronizing them will be a big challenge. This configuration does not only impact the cloud services. It has also an important impact on the knowledge synchronization between the SemKoRe Server and the SemKoRe agent. Since not all customers are keen to have a cloud connection or can have an always-on connection. Therefore, it is necessary to redesign the SemKoRe synchronization services to ensure that there is no inconsistent knowledge.

In this work, we target a large set of customers from various domains and with different needs. Schneider-Electric is not expected to create or modify ontology for each and every customer. Therefore, an important future work item is to develop a framework along with a tool suite and set of services for non-experts to allow them to create and extend their

ontology models. We will also need to address more advanced topics like ontology matching, alignment, and conflict resolution to ensure consistency.

Another future potential would be to work on lightweight triplestores for small industrial devices. Many triplestores for embedded/small platforms exist. Most of them are based on the Redland RDF Libraries [111] and are using SQLite as backend storage (e.g., RedStore [112]). However, all these solutions lack reasoning engines and do not support SWRL rules.

Regarding the sensitive maintenance data preserving, the current approach requires the involvement of a human expert. This work can be enhanced by learning from the expert's feedback to improve the accuracy of the sensitive data detection models. It can also be extended to provide a completely automated anonymization tool, that allows to identify and replace the sensitive data with adequate surrogates. The replacement of sensitive data with appropriate surrogates is still an open challenge. This process must ensure the semantic consistency and the usefulness of the resulting data, and current approaches in the literature are far from being efficient.

Finally, we focused, in all the contributions of this thesis, on maintenance reports written in English. However, standard machines are used in several countries around the world, and maintenance reports are written in multiple local languages. Thus, it would be valuable to work on a multilingual approach to prevent the language from becoming a barrier for maintenance knowledge sharing. This requires to take up various challenges, such as semantic multilingual search to find similar problems identified in other languages, or support multilingual sensitive data detection by using a cross-lingual transfer learning approach.

# Bibliography

- [1] Alan Weissberger, Cisco's Annual Internet Report (2018–2023) forecasts huge growth for IoT and M2M; tepid growth for Mobile, IEEE Communications Society, 2020, Available online: <https://techblog.comsoc.org/2020/02/20/ciscos-annual-internet-report-2018-2023-forecasts-huge-growth-for-iot-and-m2m-tepid-growth-for-mobile/> [Accessed 2021 10 18]
- [2] Bernard Marr, The 5 Biggest Internet Of Things (IoT) Trends In 2021 Everyone Must Get Ready For Now, Forbes, 2020, Available online: <https://www.forbes.com/sites/bernardmarr/2020/10/26/the-5-biggest-internet-of-things-iot-trends-in-2021-everyone-must-get-ready-for-now/> [Accessed 2021 10 18]
- [3] Laurence Goasduff, Gartner Says 5.8 Billion Enterprise and Automotive IoT Endpoints Will Be in Use in 2020, Gartner, 2019, Available online: <https://www.gartner.com/en/newsroom/press-releases/2019-08-29-gartner-says-5-8-billion-enterprise-and-automotive-io> [Accessed 2021 10 18]
- [4] CIOTTI, Marco, CICOZZI, Massimo, TERRINONI, Alessandro, et al. The COVID-19 pandemic. Critical reviews in clinical laboratory sciences, 2020, vol. 57, no 6, p. 365-388.
- [5] Juili Eklahare, Manufacturing Operations Post-Covid-19 – From Survival To Revival, Efficient Manufacturing - Industr.com, 2020, Available online: <https://www.industr.com/en/manufacturing-operations-post-covid-from-survival-to-revival-2522950> [Accessed 2021 10 18]
- [6] JAVAID, Mohd, HALEEM, Abid, VAISHYA, Raju, et al. Industry 4.0 technologies and their applications in fighting COVID-19 pandemic. Diabetes & Metabolic Syndrome: Clinical Research & Reviews, 2020, vol. 14, no 4, p. 419-422.

- [7] DAVIS, Jim, EDGAR, Thomas, PORTER, James, et al. Smart manufacturing, manufacturing intelligence and demand-dynamic performance. *Computers & Chemical Engineering*, 2012, vol. 47, p. 145-156.
- [8] SWANSON, Laura. An information-processing model of maintenance management. *International Journal of Production Economics*, 2003, vol. 83, no 1, p. 45-64.
- [9] A.-U.-H. Muhammad, S. Anwar. "A systematic review of knowledge management and knowledge sharing: Trends, issues, and challenges," *Cogent Business & Management*, vol. 3, no. 1, p. 1127744, 2016.
- [10] Norme NFX 60-010, "Maintenance - Concepts et définitions des activités de maintenance", AFNOR Editions, Normes nationales et documents normatifs nationaux, 1994.
- [11] Norme NF EN 13306, "Maintenance - Terminologie de la maintenance", AFNOR Editions, Normes nationales et documents normatifs nationaux, 1994.
- [12] Bengtsson, M., Olsson, E., Funk, P., & Jackson, M. (2004). Design of condition based maintenance system—A case study using sound analysis and case-based reasoning. *Condition Based Maintenance Systems—An Investigation of Technical Constituents and Organizational Aspects*; Malardalen University: Eskilstuna, Sweden, 57.
- [13] Norm NF X60-000, "Maintenance industrielle - Fonction maintenance", AFNOR Editions, Normes nationales et documents normatifs nationaux, 2016.
- [14] "Industry 4.0 How to navigate digitization of the manufacturing sector", McKinsey Digital, 2015, Available online: <https://www.mckinsey.com/~media/McKinsey/Business%20Functions/Operations/Our%20Insights/Industry%2040%20How%20to%20navigate%20digitization%20of%20the%20manufacturing%20sector/Industry-40-How-to-navigate-digitization-of-the-manufacturing-sector.ashx> [Accessed 2021-10-10]
- [15] Bagadia, Kishan . *Computerized Maintenance Management Systems Made Easy: How to Evaluate, Select, and Manage CMMS*. McGraw Hill Professional. 2010, ISBN 9780071491273.
- [16] GEISS, Christian T. et GRAMLICH, Manuel. Semantic Interoperability as Key for a NDE 4.0 Data Management. *Handbook of Nondestructive Evaluation 4.0*, 2021, p. 1-15
- [17] MALHOTRA, Vishal. CTO Foresees Future of CMMS-Enabled 'True Interoperability'. *Biomedical instrumentation & technology*, 2018, vol. 52, no 1, p. 60-62

- [18] K. Dalkir, "Knowledge Management In Theory And Practice," Oxford, Elsevier Inc: Jordan Hill., 2005, p. 132–133.
- [19] JONES III, D. M. G. E. Knowledge sharing and technological innovation: The effectiveness of trust, training, and good communication, *Cogent Business & Management*, 4 (1), 1387958. 2017
- [20] Cárcel-Carrasco, J., Cárcel-Carrasco, J. A., & Peñalvo-López, E. (2020). Factors in the relationship between maintenance engineering and knowledge management. *Applied Sciences*, 10(8), 2810.
- [21] SINTEF Technology and Society, Norges teknisk-naturvitenskapelige universitet, OREDA. Offshore Reliability Data Handbook, OREDA participants, 2002.
- [22] ISO 14224: 2016. Petroleum, petrochemical and natural gas industries – Collection and exchange of reliability and maintenance data for equipment. Third edition, Geneva, Switzerland.: International Organization for Standardization (ISO).
- [23] Portfolio Review 2016, System Performance, Availability and Reliability Trend Analysis (SPARTA), Northumberland, UK, 2016.
- [24] "WInD-Pool: Wind-energy-Information-Data-Pool," (Online). Available: <http://www.wind-pool.de>. [Accessed 18 10 2021].
- [25] GE Aviation, "GE Aviation launches Configuration Data Exchange to reduce maintenance costs," [Online]. Available: Available online: <https://www.geaviation.com/press-release/systems/ge-aviation-launches-configuration-data-exchange-reduce-maintenance-costs>. [Accessed 18 10 2021].
- [26] J. R. Bengt Lydell, "OPDE—The international pipe failure data exchange project," *Nuclear Engineering and Design*, vol. 238, pp. 2115-2123, 2008.
- [27] Nuclear power plants - Reliability data exchange - General guidelines, International standard ISO 6527, 1982: International Organization for Standardization.
- [28] Aromaa, S., Väättänen, A., Aaltonen, I., & Heimonen, T. (2015, July). A model for gathering and sharing knowledge in maintenance work. In *Proceedings of the European Conference on Cognitive Ergonomics 2015* (pp. 1-8).
- [29] RUIZ, Paula Potes, FOGUEM, Bernard Kamsu, et GRABOT, Bernard. Generating knowledge in maintenance from Experience Feedback. *Knowledge-Based Systems*, 2014, vol. 68, p. 4-20.



- [30] YANG, Heng-Li et WU, Ted CT. Knowledge sharing in an organization. *Technological Forecasting and Social Change*, 2008, vol. 75, no 8, p. 1128-1156.
- [31] Polanyi, Michael. 1958. *Personal Knowledge: Towards a Post-Critical Philosophy*. Chicago: University of Chicago Press. ISBN 0-226-67288-3.
- [32] M. Ben-Daya, S.O. Duffuaa, A. Raouf, J. Knezevic, D. Ait-Kadi, *Handbook of Maintenance Management and Engineering*, vol. 1, Springer, 2009.
- [33] J. Hogan, F. Hardiman, M.D. Daragh Naughton, Asset management: a review of contemporary & individualised strategies, *Lect. Notes Eng. Comp. Sci.* 2190 (1) (2011) 545–549.
- [34] Dhillon, B. S. and Liu, Y. Human error in maintenance: a review. *Journal of Quality in Maintenance Engineering* 12, (2006) 21-36.
- [35] Dhillon, B. S. *Human Reliability, Error, and Human Factors in Power Generation*. Springer Series in Reliability Engineering, Springer International Publishing, Switzerland, 2014.
- [36] SALONEN, Antti et DELERYD, Mats. Cost of poor maintenance: A concept for maintenance performance improvement. *Journal of Quality in Maintenance Engineering*, 2011.
- [37] CÁRCEL-CARRASCO, Javier et CÁRCEL-CARRASCO, José-Antonio. Analysis for the Knowledge Management Application in Maintenance Engineering: Perception from Maintenance Technicians. *Applied Sciences*, 2021, vol. 11, no 2, p. 703.
- [38] LI, Dan, FAST-BERGLUND, Asa, et PAULIN, Dan. Current and future Industry 4.0 capabilities for information and knowledge sharing. *The International Journal of Advanced Manufacturing Technology*, 2019, vol. 105, no 9, p. 3951-3963.
- [39] OLIVEIRA, A. Marcelo, LOPES, Isabel da Silva, et FIGUEIREDO, Danielle. *Maintenance management practices of companies of the industrial pole of Manaus*. 2014.
- [40] A. Crespo Marquez, J.N.D. Gupta, *Contemporary maintenance management: process, framework and supporting pillars*, *Omega – Int. J. Manage. Sci.* 34 (3) (2006) 313–326
- [41] WAN, Shan, GAO, J. X., LI, Dongbo, et al. Knowledge management for maintenance, repair and service of manufacturing system. In : *International Conference on Manufacturing Research*. Southampton Solent University, 2014.

- [42] OLIVIER, S en echal, PIRES, S. P., LOURES, E. R. F., et al. Knowledge management for sustainable performance in industrial maintenance. In : IIE Annual Conference & Expo. 2015.
- [43] D. O'Leary. Using AI in knowledge management: Knowledge bases and ontologies. *IEEE Intelligent Systems*, 13(3):34–39, May/June 1998.
- [44] BREWSTER, Christopher et O'HARA, Kieron. Knowledge representation with ontologies: the present and future. *IEEE Intelligent Systems*, 2004, vol. 19, no 1, p. 72-81.
- [45] Nu nez, D.L.; Borsato, M. An ontology-based model for prognostics and health management of machines. *J. Ind. Inf. Integr.* 2017, 6, 33–46.
- [46] Horrocks, I.; Patel-Schneider, P.F.; Boley, H.; Tabet, S.; Grosz, B.; Dean, M. SWRL: A semantic web rule language combining OWL and RuleML. *W3C Memb. Submiss.* 2004, 21, 1–31.
- [47] Melik-Merkumians, M.; Zoitl, A.; Moser, T. Ontology-based fault diagnosis for industrial control applications. In *Proceedings of the 15th Conference on Emerging Technologies & Factory Automation (ETFA 2010)*, Bilbao, Spain, 13–16 September 2010.
- [48] Palacios, L.; Lortal, G.; Laudy, C.; Sannino, C.; Simon, L.; Fusco, G.; Ma, Y.; Reynaud, C. Avionics maintenance ontology building for failure diagnosis support. In *Proceedings of the 8th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management (IC3K 2016)*, Porto, Portugal, 9–11 November 2016; Volume 2, pp. 204–209.
- [49] Peng, H.; Chang, D.; Wang, Y. Fault Diagnosis of Conveyor Based on Ontology. *Sens. Transducers* 2013, 157, 338–345.
- [50] Chen, R.; Zhou, Z.; Liu, Q.; Pham, D.T.; Zhao, Y.; Yan, J.; Wei, Q. Knowledge modeling of fault diagnosis for rotating machinery based on ontology. In *Proceedings of the IEEE 13th International Conference on Industrial Informatics (INDIN)*, Cambridge, UK, 22–24 July 2015; pp. 1050–1055.
- [51] Xu, F.; Liu, X.; Chen, W.; Zhou, C.; Cao, B. Ontology-based method for fault diagnosis of loaders. *Sensors* 2018, 18, 729.
- [52] AROMAA, Susanna, AALTONEN, Iina, et VAATANEN, Antti. Technology concepts to improve knowledge sharing during maintenance. In : *9th International Conference on Advances in Computer-Human Interactions, ACHI 2016*. International Academy, Research, and Industry Association IARIA, 2016. p. 429-435.

- [53] MEDINA-OLIVA, Gabriela, WEBER, Philippe, et IUNG, Benoît. Industrial system knowledge formalization to aid decision making in maintenance strategies assessment. *Engineering Applications of Artificial Intelligence*, 2015, vol. 37, p. 343-360.
- [54] Roberts, J. From Know-how to Show-how? Questioning the Role of Information and Communication Technologies in Knowledge Transfer. *Technology Analysis & Strategic Management* 12, 4 (2000), 429- 443
- [55] BESBES, Bassem, COLLETTE, Sylvie Naudet, TAMAAZOUSTI, Mohamed, et al. An interactive augmented reality system: a prototype for industrial maintenance training applications. In : 2012 IEEE international symposium on mixed and augmented reality (ISMAR). IEEE, 2012. p. 269-270.
- [56] AROMAA, Susanna, AALTONEN, Iina, KAASINEN, Eija, et al. Use of wearable and augmented reality technologies in industrial maintenance work. In : Proceedings of the 20th international academic mindtrek conference. 2016. p. 235-242.
- [57] GAVISH, Nirit, GUTIÉRREZ, Teresa, WEBEL, Sabine, et al. Evaluating virtual reality and augmented reality training for industrial maintenance and assembly tasks. *Interactive Learning Environments*, 2015, vol. 23, no 6, p. 778-798.
- [58] ERKOYUNCU, John Ahmet, DEL AMO, Iñigo Fernandez, DALLE MURA, Michela, et al. Improving efficiency of industrial maintenance with context aware adaptive authoring in augmented reality. *Cirp Annals*, 2017, vol. 66, no 1, p. 465-468.
- [59] WOLFARTSBERGER, Josef, ZENISEK, Jan, et WILD, Norbert. Data-driven maintenance: Combining predictive maintenance and mixed reality-supported remote assistance. *Procedia Manufacturing*, 2020, vol. 45, p. 307-312.
- [60] AMAN, Rafael, AROMAA, Susanna, et KAASINEN, Eija. Knowledge sharing solutions for field service personnel: Remote assistance. In : S-STEP-Smart Technologies for Lifecycle Performance: Final Report 1/2017. DIMECC Oy, 2017. p. 108-111.
- [61] FERRISE, Francesco, CARUSO, Giandomenico, et BORDEGONI, Monica. Multimodal training and tele-assistance systems for the maintenance of industrial products: This paper presents a multimodal and remote training system for improvement of maintenance quality in the case study of washing machine. *Virtual and Physical Prototyping*, 2013, vol. 8, no 2, p. 113-126.
- [62] ZHENG, Xianjun Sam, MATOS DA SILVA, Patrik, FOUCAULT, Cedric, et al. Wearable solution for industrial maintenance. In : Proceedings of the 33rd Annual ACM

- Conference Extended Abstracts on Human Factors in Computing Systems. 2015. p. 311-314.
- [63] JUNGWIRTH, Florian, GOLLAN, Benedikt, BREITENFELLNER, Marcel, et al. Eye-Control: wearable assistance for industrial maintenance tasks. In : Adjunct Proceedings of the 2019 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2019 ACM International Symposium on Wearable Computers. 2019. p. 628-632.
- [64] Wan, S.; Li, D.; Gao, J.; Roy, R.; Tong, Y. Process and knowledge management in a collaborative maintenance planning system for high value machine tools. *Comput. Ind.* 2017, 84, 14–24.
- [65] WAN, Shan, LI, Dongbo, GAO, James, et al. A collaborative machine tool maintenance planning system based on content management technologies. *The International Journal of Advanced Manufacturing Technology*, 2018, vol. 94, no 5-8, p. 1639-1653.
- [66] KOLODNER, Janet. *Case-based reasoning*. Morgan Kaufmann, 2014.
- [67] Bergmann, R. and Wilke, W. (1996). On the role of abstraction in case-based reasoning. *Third European Workshop on Case-Based Reasoning, Lecture Notes in Artificial Intelligence*, Springer Verlag, pp 28-43
- [68] Pellet Reasoner. Available online: <https://www.w3.org/2001/sw/wiki/Pellet> (accessed on 24 April 2020).
- [69] Berners-Lee, T.; Hendler, J.; Lassila, O. The semantic web. *Sci. Am.* **2001**, 284, 34–43.
- [70] GraphDB. Available online: <http://graphdb.ontotext.com> (accessed on 24 April 2020).
- [71] Node-Red. Available online: <https://nodered.org> (accessed on 24 April 2020).
- [72] Azure IoT Hub. Available online: <https://azure.microsoft.com/en-us/services/iot-hub> (accessed on 24 April 2020).
- [73] Banks, A.; Gupta, R. MQTT Version 3.1.1. *OASIS Stand.* **2014**, 29, 89.
- [74] Docker: Empowering App Development for Developers. Available online: <https://www.docker.com> (accessed on 24 April 2020).
- [75] Boettiger, C. An introduction to Docker for reproducible research. *ACM SIGOPS Oper. Syst. Rev.* **2015**, 49, 71–79.

- [76] Haslhofer, B.; Roochi, E.M.; Schandl, B.; Zander, S. *Europeana RDF Store Report*; Austrian National Library, University of Vienna, Vienna, Austria, 2011.
- [77] Modom.io. Available online: <https://modom.io> (accessed on 24 April 2020).
- [78] Zazuko Ontology Manager. Available online: <https://zazuko.com/products/ontology-manager> (accessed on 24 April 2021).
- [79] Alobaid, A.; Garijo, D.; Poveda-Villalón, M.; Santana-Pérez, I.; Fernández-Izquierdo, A.; Corcho, Ó. Automating ontology engineering support activities with OnToology. *J. Web Semant.* **2019**, *57*, 100472.
- [80] M. Nuno, B. Jorge, D. Francisco, "Automated anonymization of text documents," IEEE congress on evolutionary computation (CEC), 2016.
- [81] LANDIS, J. RICHARD, & KOCH, GARY G. 1977. The measurement of observer agreement for categorical data. *Biometrics*, *33*, 159–174.
- [82] M. Honnibal, M. Ines . "spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing." *To appear 7.1* (2017): 411-420.
- [83] Y. N. LEE, W. Sun, H. L. CHIEU, et al. Conditional random fields with high-order features for sequence labeling. *Advances in Neural Information Processing Systems 22* (NIPS 2009), 2009, p. 2196-2204.
- [84] Z. Yukun, K. Ryan, Z. Rich, "Aligning books and movies: Towards story-like visual explanations by watching movies and reading books," IEEE international conference on computer vision, pp. 19-27, 2015.
- [85] D. Jacob, C. Ming-Wei, L. Kenton: Pre-training of deep bidirectional transformers for language understanding," arXiv preprint arXiv:1810.04805, 2018.
- [86] KONKOL, Michal et KONOPIK, Miloslav. Segment representations in named entity recognition. In : *International Conference on Text, Speech, and Dialogue*. Springer, Cham, 2015. p. 61-70.
- [87] ROSVALL, Erik. Comparison of sequence classification techniques with BERT for named entity recognition. 2019.
- [88] S. Soufian, H. Nicolas, M. Emmanuel, "Dialogue act taxonomy interoperability using a meta-model," *International Conference on Computational Linguistics and Intelligent Text Processing*, 2017.

- [89] Boyes, Hugh; Hallaq, Bil; Cunningham, Joe; Watson, Tim (October 2018). "The industrial internet of things (IIoT): An analysis framework". *Computers in Industry*. 101: 1–12. doi:10.1016/j.compind.2018.04.015
- [90] DELL'AGLIO, Daniele, DELLA VALLE, Emanuele, CALBIMONTE, Jean-Paul, et al. RSP-QL semantics: A unifying query model to explain heterogeneity of RDF stream processing systems. *International Journal on Semantic Web and Information Systems (IJSWIS)*, 2014, vol. 10, no 4, p. 17-44.
- [91] DELL'AGLIO, Daniele, CALBIMONTE, Jean-Paul, DELLA VALLE, Emanuele, et al. Towards a unified language for RDF stream query processing. In : *European Semantic Web Conference*. Springer, Cham, 2015. p. 353-363.
- [92] GUTIERREZ, Claudio, HURTADO, Carlos, et VAISMAN, Alejandro. Temporal rdf. In : *European Semantic Web Conference*. Springer, Berlin, Heidelberg, 2005. p. 93-107.
- [93] GUTIERREZ, Claudio, HURTADO, Carlos A., et VAISMAN, Alejandro. Introducing time into RDF. *IEEE Transactions on Knowledge and Data Engineering*, 2006, vol. 19, no 2, p. 207-218.
- [94] HURTADO, Carlos et VAISMAN, Alejandro. Reasoning with temporal constraints in RDF. In : *International Workshop on Principles and Practice of Semantic Web Reasoning*. Springer, Berlin, Heidelberg, 2006. p. 164-178.
- [95] H. Hossayni, I. Khan and C. E. Kaed, "Embedded Semantic Engine for Numerical Time Series Data," 2018 Global Internet of Things Summit (GIoTS), 2018, pp. 1-6, doi: 10.1109/GIOTS.2018.8534580.
- [96] EL KAED, Charbel, KHAN, Imran, HOSSAYNI, Hicham, et al. SQenIoT: Semantic query engine for industrial Internet-of-Things gateways. In : *2016 IEEE 3rd World Forum on Internet of Things (WF-IoT)*. IEEE, 2016. p. 204-209.
- [97] EL KAED, Charbel, KHAN, Imran, HOSSAYNI, Hicham, et al. SQenIoT: Semantic query engine for industrial Internet-of-Things gateways. In : *2016 IEEE 3rd World Forum on Internet of Things (WF-IoT)*. IEEE, 2016. p. 204-209.
- [98] ESPINOZA-ARIAS, Paola, GARIJO, Daniel, et CORCHO, Oscar. Crossing the chasm between ontology engineering and application development: A survey. *Journal of Web Semantics*, 2021, vol. 70, p. 100655.
- [99] R. Cyganiak, C. Bizer, Pubby – A linked data frontend for SPARQL endpoints, 2007, Available online: <http://wifo5-03.informatik.uni-mannheim.de/pubby/>. (Accessed: 2021-10-20).

- [100] Google, Puelia, 2010, Available online: <https://code.google.com/archive/p/puelia-php/>. (Accessed: 2021-10-20).
- [101] Epimorphics, Elda: The linked-data API in Java, 2011, Available online: <http://epimorphics.github.io/elda/index.html>. (Accessed: 2021-10-20).
- [102] M. Fernandez-Sellers, Linked open data inspector, 2015, Available online: <https://github.com/marfersel/LODI/>. (Accessed: 2021-10-20).
- [103] P. Heyvaert, B. De Meester, H. Pandit, R. Verborgh, Walder, 2020, Available online: <https://github.com/KnowledgeOnWebScale/walder>. (Accessed: 2021-10-20).
- [104] S. Mayer, J. Hodges, D. Yu, M. Kritzler, F. Michahelles, An open semantic framework for the industrial internet of things, *IEEE Intell. Syst.* 32 (1) (2017) 96–101.
- [105] G. Jansen, A. Coburn, A. Soroka, R. Marciano, Using data partitions and stateless servers to scale up fedora repositories, in: 2019 IEEE International Conference on Big Data (Big Data), IEEE Computer Society, 2019, pp. 3098–3102.
- [106] E. Daga, L. Panziera, C. Pedrinaci, A BASILar approach for building web APIs on top of SPARQL endpoints, in: *CEUR Workshop Proceedings*, vol. 1359, 2015, pp. 22–32.
- [107] M. Daquino, I. Heibi, S. Peroni, D. Shotton, Creating restful APIs over SPARQL endpoints with RAMOSE, 2020, ArXiv Preprint ArXiv:2007.16079
- [108] Linked Data Platform 1.0, W3C Recommendation 26 February 2015, Online:<https://www.w3.org/TR/ldp/>
- [109] Web Ontology Language Reference. Available online: Available online: <https://www.w3.org/TR/owl-ref> (Accessed: 2021-10-20).
- [110] Shapes Constraint Language (SHACL), W3C Recommendation 20 July 2017<https://www.w3.org/TR/shacl/>
- [111] Redland RDF Libraries. Available online: <http://librdf.org> (accessed on 24 April 2021).
- [112] Haslhofer, B.; Roochi, E.M.; Schandl, B.; Zander, S. *Europeana RDF Store Report*; Austrian National Library, University of Vienna, Vienna, Austria, 2011.







Annex **A**

# Paper I

Article

# SemKoRe: Improving Machine Maintenance in Industrial IoT with Semantic Knowledge Graphs

Hicham Hossayni <sup>1,\*</sup>, Imran Khan <sup>1</sup>, Mohammad Aazam <sup>2</sup>, Amin Taleghani-Isfahani <sup>1</sup> and Noel Crespi <sup>3</sup>

<sup>1</sup> Schneider Electric, 38000 Grenoble, France; imran2.khan@se.com (I.K.); amin.taleghani@se.com (A.T.-I.);

<sup>2</sup> University of Nottingham, School of Computer Science, Semenyih 43500, Malaysia; aazam@ieee.org

<sup>3</sup> Institut Polytechnique de Paris, IMT, Télécom-SudParis, 91011 Evry Cedex, France; noel.crespi@it-sudparis.eu

\* Correspondence: hicham.hossayni@se.com

Version January 27, 2022 submitted to Appl. Sci.



**Abstract:** The recent focus on sustainability and improved efficiency requires innovative approaches in industrial automation. We present SemKoRe, a knowledge graph developed to improve machine maintenance in the industrial domain. SemKoRe is vendor-agnostic, it helps Original Equipment Manufacturers (OEMs) to capture, share and exploit the failure knowledge generated by their customers machines located around the world. Based on our interactions with actual customers, it usually takes several hours to days to fix a machine-related issue. During this time, production stops and incurs cost in terms of lost production. SemKoRe significantly enhances the maintenance process by reducing the failure diagnostic time, and by centralizing machine maintenance knowledge fed by the experts and technicians around the world. We developed flexible architecture to cover our customers' varying needs, along with failure and machine domain ontologies. To demonstrate the feasibility of SemKoRe, a proof-of-concept is developed. SemKoRe gathers all failure related data in the knowledge graph, and shares it among all connected customers in order to easily solve future failures of the same type. SemKoRe received the approval of several substantial clients located in USA, UK, France, Germany, Italy and China, associated with various segments such as pharmaceutical, automotive, HVAC and food and beverage.

**Keywords:** failure diagnostics; industry 4.0; industrial internet of things (IIoT); knowledge graph; machine maintenance; semantic web

## 1. Introduction

Industrial Internet of Things (IIoT) has emerged as an enabler of the rapid integration of advanced technologies in the industrial world [1]. Factories are becoming fully connected and smart, thereby allowing manufacturers to improve process efficiency, sustainability, and safety while decreasing costs. Many industries are making heavy investments in smart manufacturing and production systems. In return, they expect optimal and sustainable production with minimum maintenance efforts. This makes maintenance one of the most important aspects of industrial process activities. Formerly considered as part of general enterprise costs, it has become a real source of data and critical for business continuity and performance [2].

Systems such as Computerized Maintenance Management System (CMMS), Manufacturing Execution System (MES) and Enterprise Resource Planning (ERP) are used to perform maintenance activities in several industries [3]. These systems provide features such as predictive and preventive maintenance, maintenance planning, scheduling, execution, monitoring and traceability. However,

31 these systems have two inherent drawbacks. First, they are intended to optimize the maintenance  
32 process at given location (a factory or a site). This means that two different factories (or sites) cannot  
33 share the details of a specific machine's maintenance operations without a human expert in the loop.  
34 Such sharing is useful when machines share the same characteristics and perform the same operations  
35 regardless of their locations. While cloud-based offers are a potential remedy, several customers still  
36 do not want to have their data on multiple vendor cloud platforms.

37 The second drawback of these systems is that they are not interoperable at the semantic level.  
38 Schneider Electric works with several Original Equipment Manufacturers (OEMs) who design,  
39 build and ship machines for their customers around the world. In our experience, there is no easy way  
40 to align the data coming out of these maintenance systems and to thereby get a uniform understanding.  
41 This issue is complicated by the heterogeneity of these systems and the associated silos since each  
42 business segment and customer is unique and operates under different regulatory and geographic  
43 constraints. Despite these challenges, our customers are increasingly demanding better visibility about  
44 the performance of their assets, reductions in maintenance costs and downtime, improved productivity  
45 and more agility in their end-to-end processes.

46 In this paper, we present the early outcomes of our work, SemKoRe: how we use it to construct  
47 knowledge graphs of machine failures and exploit it to address various issues. SemKoRe is a vendor  
48 agnostic solution that uses formal, shared and explicit models to capture the details of machine  
49 domains, the failures of these machines and the applied repairing procedures. SemKoRe is designed  
50 to speed-up the maintenance process and to allow for quick recovery from failures. When a new  
51 machine is installed in a factory today, there is no existing knowledge of its failures. In any given  
52 factory, each failure is only discovered at its first occurrence. The maintenance process includes  
53 diagnostics to determine the reasons for a failure, its impact, and to define and apply the correct repair  
54 procedures. This process is repeated at different locations for the same machines having the same  
55 failures. In reality, failure details are usually captured manually, e.g., using spreadsheets (e.g., Excel).  
56 This approach is not fault proof as each person filling out the sheet cannot be expected to provide  
57 all the required information and even if that information is given, there will be semantic mismatch,  
58 e.g., one person describes issue as "abnormal rotation speed" while another person describes the same  
59 issue as "irregular spinning rate". Both mean the same but use different semantics (we provide more  
60 details on it based on interactions with our actual customers in Section 2).

61 SemKoRe helps to avoid this semantic mismatch, and captures all the machine, failure and  
62 maintenance data as a knowledge graph, allowing several actors to benefit (Sections 4 and 5).  
63 For example, Operators and Technicians can benefit from the knowledge provided by other operators at  
64 different sites to address their issues. This will also reduce the risk of mismanipulation of machines by  
65 incompetent operators, which is a considerable industrial threat in reality [4]. OEM Machine Builders  
66 can improve the next generations of the machines they build, thanks to the knowledge captured in  
67 SemKoRe that helps them to know why certain machines have higher failure rates. Analytic teams can  
68 improve their work, as SemKoRe will provide a global view of machine failures and the background  
69 of a variety of contexts.

70 We implement our SemKoRe system using a triplestore called "GraphDB by Ontotext" (as  
71 cloud/gateway triplestore), IBM Node-Red (flow-based development tool), Microsoft Azure (for cloud  
72 service), Azure IoT Hub (for IoT connectivity), and Docker (to package SemKoRe services) (Section 6).  
73 To support the needs of different actors, SemKoRe is developed using the semantic web and ontologies  
74 [5]. This approach helps to accommodate future requirements and to provide a clear separation of  
75 concerns between the application needs and the domain knowledge which in this case is machine  
76 domain and failure domain knowledge. We adopted a distributed architecture in which knowledge  
77 collection is performed on the edge layer. The collected knowledge is shared with other actors and  
78 machines through a cloud-based instance.

79 We elaborate the lessons learned in Section 7.1, and offer future research directions in Section 7.2  
80 and conclude in Section 7.3.

## 81 2. Motivating Scenario and Requirements

82 In this section, we talk about some key motivating scenarios, with practical examples, that would  
83 help to envision the core idea of the problem domain. Then we present the set of customers  
84 requirements that guided us during the elaboration of our solution.

### 85 2.1. Motivating Scenario

86 The following scenario is based on our interactions with real customers who want to improve  
87 their existing maintenance process. Let us consider three actors: Bob the machine operator, Alice the  
88 maintenance technician and Joe the OEM machine builder. On a given day, Bob is working on factory  
89 floor operating several machines when suddenly one machine stops working. Bob spends some time  
90 to fix the issue himself but is unable to do so, since Bob's main job is to operate the machine. He might  
91 be able to fix small issues due to his experience but he is supposed to call a qualified technician for  
92 anything major. He then calls Alice to come to factory floor to check on the machine. When Alice checks  
93 the machine she finds that she is also not able to solve the issue so she calls the OEM or Schneider  
94 Electric service bureau, where a machine expert guides her through the repair process. Finally, Alice is  
95 able to fix the issue and the machine starts working.

96 The whole process took a long time and while Bob is now able to operate his machine, if the same  
97 issue occurs in a similar type of machine located in a different city the same process would likely be  
98 repeated because only Alice knows how to quickly solve this particular issue. However, if Alice can  
99 describe what she learned from the service bureau and share her experience with the technicians in  
100 other sites by using some appropriate mechanism, they could all benefit from this common knowledge.

101 Another beneficiary of this common knowledge is Joe. Today, when Joe gets reports about the  
102 issues with his machines from different customers, he has no easy way to get the finer details that can  
103 only come from the technicians like Alice. These details could be useful and help him to understand  
104 why some of his machines are facing particular issues. This can help him to improve the design and  
105 engineering process of his machines, especially in the case of hundreds or thousands of machines  
106 being used worldwide, the scale of problem and timely action in resolving the issue becomes hugely  
107 difficult. Another benefit is that using the insights from customer A, Joe can help customer B to quickly  
108 respond to machine issues while respecting of privacy and sensitive nature of the information, if both  
109 customers have the same type of machines. The importance of the quick resilience after failure aspect  
110 is discussed in details by Alcaraz et al. in [6].

### 111 2.2. Requirements

112 Based on the motivating scenario described above, we now present the following set of  
113 requirements. The first requirement is that the proposed solution should make it easy to capture and  
114 share knowledge among various actors. The second requirement is that the proposed solution should  
115 be usable both on cloud (public or private) and on-premise systems. Indeed many customers are  
116 willing to connect their machines and factories to the cloud, while others choose to fully isolate their  
117 factories in order to protect their industrial property and to keep their private data locally. The third  
118 requirement is that the solution should have built-in mechanism to protect the sensitive information  
119 about the processes and the business. During our interactions with customers, this requirement came  
120 up as the make or break point for them. The fourth requirement is that the solution should support  
121 root cause analysis and make it easy to identify the component(s) that cause the failures. The fifth  
122 requirement is that the solution should be platform-independent and thus should not depend on any  
123 particular hardware or software platform. The sixth and last requirement is that the proposed solution  
124 should be open and extensible to cover the current as well as future needs. These requirements are also  
125 thought to avoid introducing security issues or affecting the machines' performances in the customers  
126 sites [7].

### 3. Related Works

E. Kharlamov et al. present one of the earlier works, from a major industrial company on capturing industrial information models using W3C standards [8]. This work proposes an application front-end to allow non-semantic experts to develop ontologies. The front-end is a modified Web-Protege [9], that hides the complexity associated with the desktop Protege version. The work highlights the benefits of involving domain experts to capture the domain knowledge and to create different services using it. However, the solution does not cover our main requirements.

N. Zaini et al. [10] propose a generic online tool for building collaborative ontology without prior deep knowledge of the domain. An initial ontology is built, populated and enriched by multiple participants in a collaborated manner. The goal is to simplify the ontology based modeling of domain knowledge for the users without ontology expertise. However, the ontology concepts are not sufficiently abstracted, as these concepts are simply renamed. This means that despite simplification, substantial semantic expertise is needed to make the necessary modifications to the ontology. Another important missing element is that there are no checks to ensure consistency which can be an issue in a multiple user environment.

There is a large pool of work on industrial maintenance. D. L. Nunez et al. [11] created a taxonomy of the Prognostics and Health Management in manufacturing. They propose a formal ontology for failure prognostics based on industrial ISO standards for failure mode analysis, failure diagnostics and prognostics (e.g., ISO 13372, ISO 13379, ISO 13381 and others). Failure knowledge is described in ontologies from ISO standards. Semantic Web Rule Language (SWRL) [12] is used to define rules in order to generate warning messages in case of abnormal states. However no approach is described to share the acquired failure knowledge among different users.

In [13], M. Melik-Merkumians et al. used ontologies for fault diagnosis for industrial control applications. They utilized reasoning capability to check the model consistency over the time and to raise early-alarms for critical failures.

L. Palacios et al. [14] propose an ontology based support for fault diagnosis for aircraft maintenance operations. An aircraft maintenance ontology is modeled and fed by the (manual) alignment of several existing ontologies related to the avionics domain to discover the relations between the causes and failure symptoms, explain the failures and any unscheduled maintenance requirements as well as the possible procedures that can be applied to each situation.

An ontology-based approach is adopted by H. Peng et al. [15] for the fault diagnostics of conveyors. The knowledge about fault symptoms, fault causes and fault solutions was modeled with multiple ontologies. The resulted ontologies were mapped together based on a mathematical formulation of conveyor fault diagnostics. Some reasoning rules were defined in order to infer additional relations between faults, symptoms and potential causes.

In [16], R. Chen et al. used ontologies to model the knowledge of fault diagnosis for rotating machines. Their proposed ontology model describes fault diagnosis knowledge considering the vibration characteristics as main fault factor. The model's reasoning capability is considered by defining some SWRL rules for fault diagnostics.

F. Xur et al. [17] also relied on ontologies to design a loader fault diagnosis system. It aims to help users find the fault causes, locations and fault maintenance measures of loaders in a reasonable amount of time. Ontology is used to model the loader information and describe the relative failures. This work uses the Condition Based Reasoning (CBR) method to diagnose loader faults by finding similar corresponding situations in the past. When no corresponding case is found, CBR fails and the (SWRL-based) Rule Based Reasoning (RBR) approach is proposed for fault diagnosis.

In [18], S. Wan et al. developed a Collaborative Maintenance System Planning that allows many stakeholders to collaborate to ensure maintenance process quality. An ontology-based approach is adopted to model a large field of knowledge: the machine domain model, failure knowledge and stakeholders knowledge are modeled together to ensure the interoperability between their systems, the maintenance planning and Resources and Constraints knowledge. However, this centralized

177 solution focuses mainly on preventive maintenance planning. In addition, the managed failure  
 178 knowledge is relatively basic and does not consider root causes or symptoms.

179 All the works mentioned above use ontologies to model machine data models and failure  
 180 knowledge. However, none of these works satisfy all of the requirements that we identified from  
 181 our motivating scenario. These works developed various ontology models and some exhaustively  
 182 described potential failures and their characteristics. However, to the best of our knowledge no  
 183 machine failure ontology is available for reuse or for extension. Furthermore, neither of our two major  
 184 requirements, i.e., knowledge sharing and data confidentiality have been considered.

185 Furthermore, many cloud-based solutions are proposed to enhance the maintenance process  
 186 for different domains like smart grids, shop-floors, etc. Different aspects were analyzed:  
 187 such as remote maintenance [19], fault detection [20], machines monitoring [19,21], preventive  
 188 maintenance scheduling [22], predictive maintenance [23], or data confidentiality [24]. However,  
 189 none of these studies considered sharing experiences or knowledge between different actors for  
 190 maintenance purpose.

#### 191 4. Architecture and Ontology Models

192 In this section, we discuss our contributions and our proposed architectures, along with the  
 193 developed ontology models.

##### 194 4.1. High-Level Architecture

195 As mentioned before, our customers require different deployment options, and so we divided  
 196 them into three categories and developed three architectures. In the first category, customers prefer  
 197 to not connect their machines to the cloud and some even do not want to connect to the Internet due  
 198 to the sensitive nature of their business, and to protect their data. The architecture proposed for this  
 199 category is shown in Figure 1a. The second category of customers opted for an entirely connected  
 200 architecture, in which the machines/gateways in their factories are directly connected to the cloud.  
 201 For this category, we proposed the architecture shown in Figure 1b. The third architecture targets the  
 202 customers who refuse to connect their machines to the cloud, but are ready to deploy a local on-premise  
 203 server between the cloud and their machines. For this use case, we proposed a hybrid architecture  
 204 Figure 1c, where most of the collected data stays in the local server, and only an anonymized part of  
 205 the data are transmitted to the cloud.

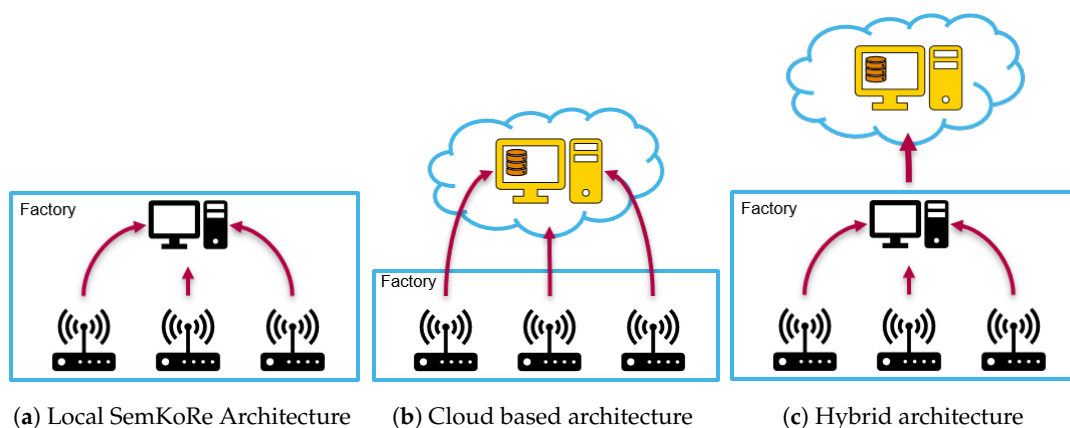




Figure 1. SemKoRe architectures;  IoT gateway connected to a real machine:  Local server.

206 In all these cases, each machine is connected to an industrial IoT gateway, e.g., Modicon M262 <sup>1</sup>  
 207 to collect the run-time data of the machine and the information provided by maintenance personnel

<sup>1</sup> <https://www.se.com/ww/en/product-range/65771-modicon-m262/>

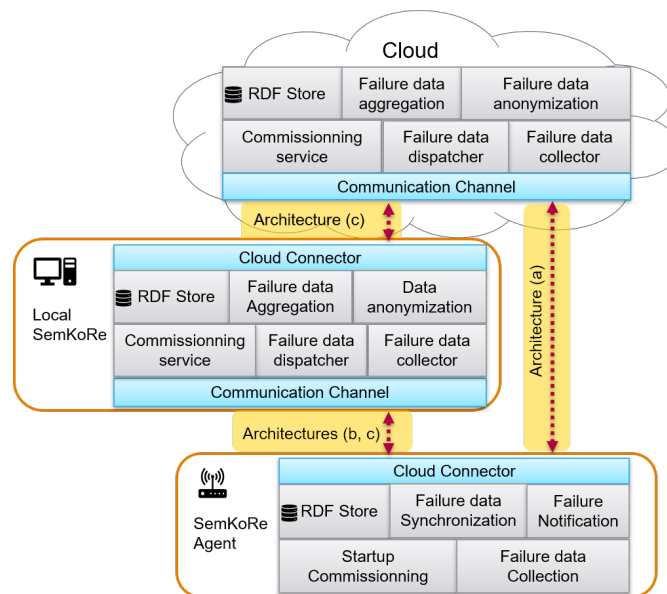
and/or operators. Each gateway is connected to a central entity (either a Local SemKoRe or a SemKoRe Server) which collects the data from all the gateways and then aggregates and shares with the gateways connected to the same type of machine.

Though, the Achilles heel of our proposal remains the case of the first category of customers, i.e., who do not want to connect their factories to the cloud. The unique technical solution to share the maintenance knowledge consists of using physical supports (e.g., USB keys, CDs, ...) with human intervention. However, the internal business case analysis is still in progress. We believe that once the business case is finalized, the technical adaptation of the SemKoRe services will be trivial and will not require lot of efforts.

In this paper, we focus on the cloud-based architecture because it covers all the constraints and features of the other architectures. This architecture is also implemented in a proof-of-concept to demonstrate the feasibility (see Section 6).

#### 4.2. Detailed Architecture

Figure 2 shows the detailed architecture of the SemKoRe.



**Figure 2.** SemKoRe detailed architecture.

The SemKoRe consists of three entities:

1. A SemKoRe Agent: Runs on industrial IoT gateways connected to the machines. It collects data when failures occur in the connected machines. According to the chosen architecture, the collected data are then shared with either the SemKoRe Server or Local SemKoRe. The SemKoRe Agent is designed to fit in all architectures (see Figure 1).
2. A SemKoRe Server: Running on the cloud, it manages several failure data producers, i.e., Local SemKoRes or SemKoRe Agents. The collected failure data are validated by an expert and aggregated and afterwards shared with the SemKoRe Agents and/or Local SemKoRes.
3. Local SemKoRe: Lightweight instance of the SemKoRe Server deployed on a local server to manage the machines located in a site or factory. It collects data produced by the SemKoRe Server (if available) and the SemKoRe Agents on local gateways. The data aggregation is done locally, and only aggregated data are shared with the SemKoRe Server. The cloud then merges its aggregations with the Local SemKoRe aggregation, and pushes back the updates to the corresponding entities.



236 We use in the cloud a message broker to connect the machine gateways to the SemKoRe Server  
237 for bi-directional data sharing. We also developed a REST (Representational State Transfer) interface  
238 on the SemKoRe Server side to directly call remote services, e.g., commissioning service.

### 239 4.3. Machine Failure Ontology Model

240 The first part of our work is collecting information on machine failures to be able to answer the  
241 following questions:

- 242 • What are the failure symptoms? Symptoms reflect the perceptible aspects of failures whether they  
243 are visual, sonic, odor or heat related.
- 244 • What is the impact of the failure? This may or may not be detected easily. Each failure impact is  
245 relative to a machine or to one of its components.
- 246 • What are the root causes of a particular failure? This question is difficult to answer, since it assumes  
247 prior knowledge about the cause-effect relations specific to each machine type. Answering this  
248 question requires the knowledge of a machine domain expert.
- 249 • After knowing the failure type and impact, how can we repair the machine?
- 250 • After knowing the root causes of a specific failure, is there a preventive maintenance procedure  
251 that can help us to avoid that failure?

252 To answer all of these questions, we defined the data model of the machine failures using  
253 Semantic Web standards because of the schema-less nature of RDF (Resource Description Framework),  
254 RDFS (RDF Schema), OWL (Web Ontology Language) and the explicit formalism supported by these  
255 languages. The failure ontology is created by interacting with the machine builders and by using the  
256 initial set of requirements described in Section 2.2. It acts as a common data model and will be enriched  
257 with new concepts by domain experts over time. Progressively, our design, engineering, configuration  
258 and maintenance tools will use this ontology to create the knowledge about the failures and allow us  
259 to develop different services over it. SemKoRe targets the industry automation business, in which  
260 the failure knowledge can be significantly different from one customer to another. The concept uses a  
261 flat ontology model, containing only the most common general concepts required for current needs.  
262 The subsequent specialization concepts will be easily and naturally added by domain experts as the  
263 knowledge collection progresses.

264 To develop the machine failure ontology presented in Figure 3, we adopted the Seven-step method,  
265 developed by the Medical Information Center of Stanford University [25]. Its seven steps are as follows:

- 266 1. **Determine domain and scope:** This work focuses on industrial machine failures.
- 267 2. **Consider reusing existing ontologies:** No existing machine domain or machine failure ontology  
268 was found for reuse, therefore we developed both for this work. Regarding upper-level ontologies,  
269 there are several candidates like Basic Formal Ontology (BFO), ISO-15926, Gist, and Suggested  
270 Upper Merged Ontology (SUMO) but we still need to finalize one.
- 271 3. **List important terms in the ontology:** After interactions with the machine domain experts,  
272 the following important terms were identified Failure, Symptom, Impact, Root cause, Solving  
273 Procedure, among others.
- 274 4. **Define classes and class hierarchy:** Several classes were created including the important terms  
275 listed above. However, since no specialization concept will be introduced, the ontology is flat.  
276 Only the classes relative to types (e.g., Failure Type, Symptom Type) are grouped as sub-classes  
277 of the Types class.
- 278 5. **Define object properties of classes:** We defined a set of object properties that link all the defined  
279 classes together. For example, the property *hasSymptoms* links a Symptom to a specific Failure.  
280 The complete list is illustrated in Figure 4b
- 281 6. **Define data properties of classes:** We also defined several of the data properties of classes,  
282 with cardinality and type constraints, as shown in Figure 4c.

283 7. **Create instances and check exceptions:** Instances of SemKoRe ontology are divided into two  
 284 parts. The first part, defined by experts during the design time, concerns the generic concepts that  
 285 will be used for most industrial use cases, such as Severity level (Catastrophic, Critical, Moderate,  
 286 Low). The second type of instances concern the data provided by the users during the runtime  
 287 of the SemKoRe. Users instances include details about the failures and symptoms. To check for  
 288 exceptions, we used Pellet reasoner [26] to verify the correctness of our ontology model.



Figure 3. Failure ontology.

289 In addition to the failure ontology, we also created a machine domain ontology (Figure 4a) to  
 290 describe the machine components to satisfy our requirement to link failures to specific machine  
 291 components when and where they occur, as simply knowing about a failure is not useful on its  
 292 own. We also need to identify the components that caused a failure or that show failure symptoms,  
 293 so that they can be identified as candidates for repair or inspections. For simplification, we only  
 294 described two types of machines in our machine domain ontology, a Tray Sealers Machine and a  
 295 Packaging Machine. Each machine type is composed of many components (PLC, Drive, Actuator,  
 296 Sensor, and others), connected through different communication buses. To integrate both ontologies,  
 297 we created OWL Class *FailureAsset*, to associate failures to the corresponding components in the  
 298 machine domain ontology.

299 However, we faced another issue to identify the exact component of a machine that has failed or  
 300 is impacted by the failure. For example, consider that a machine has two Servo Drives of the same type.  
 301 These Servo Drives are described in the machine domain ontology as two instances (SDA and SDB)  
 302 of the “ServoDrive” class and are associated with the instance of a machine. When a failure occurs

303 in SDA, we should be able to identify it through ontology. Such detailed identification is especially  
 304 useful when the failure knowledge is shared with the other sites using the same machine type, as it  
 305 will help them to recognize the exact component responsible or impacted by the failure.

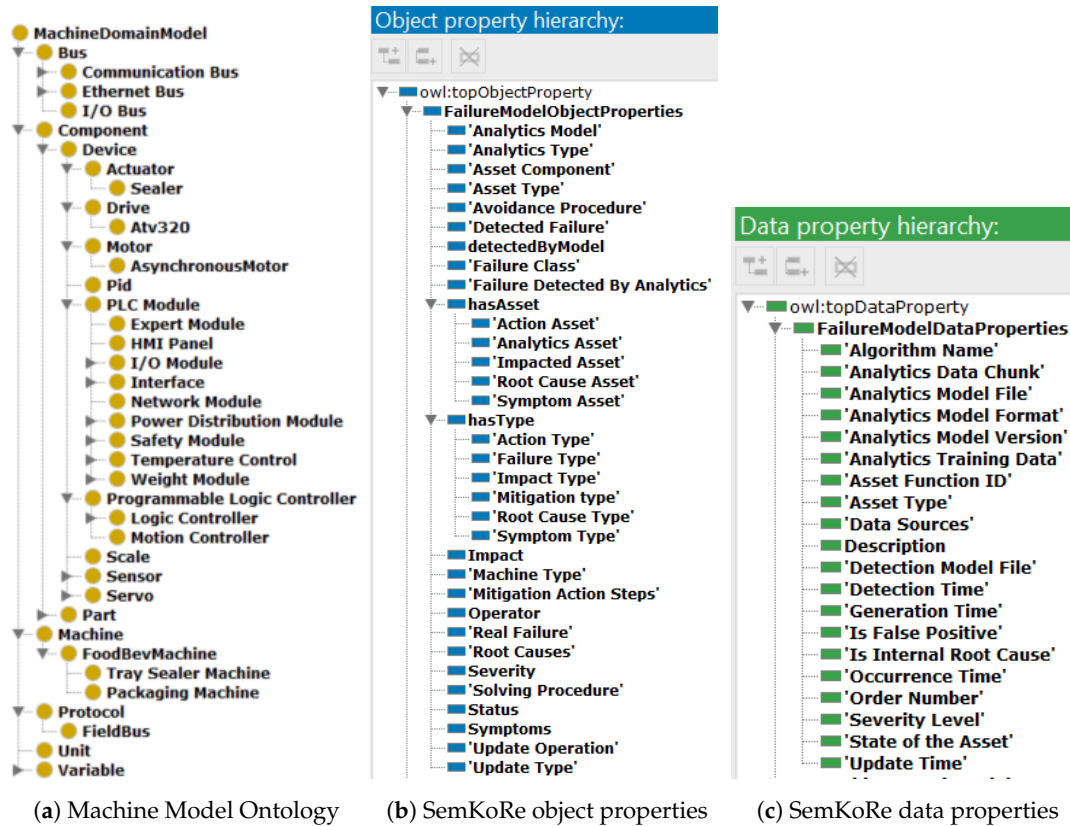


Figure 4. SemKoRe ontology design.

306 To address this issue, each component in the machine domain ontology has a unique number  
 307 “FunctionID”, to distinguish its role in the machine compared to other components of the same type.

308 We used both desktop Protégé [27] and Web-Protégé [9] to create our ontologies. The latter  
 309 allowed us to include machine domain experts in the ontology development process and to gather and  
 310 organize their feedback.

311 It is important to mention that our main focus in this work has been on the validation of the  
 312 idea to the OEMs, that formalized knowledge about machines and their failures can be useful for  
 313 quick resolution of failures and to improve Overall Equipment Effectiveness (OEE). The ontology,  
 314 in its current form, will be extended to cover the needs of the OEMs. To guarantee the ontology  
 315 generality, a potential extension basis could be the adoption of taxonomy of ISO standards for failure  
 316 mode analysis (similarly to [11]). Relevant actors can be involved to extend these ontologies with new  
 317 concepts and relationships by using appropriate tools like the ones mentioned in Section 7.1.

## 318 5. SemKoRe Process

319 Figure 5 shows the SemKoRe process of failure data collection and sharing. The process  
 320 is distributed on two layers: on the edge with the SemKoRe Agent, and on the cloud with the  
 321 SemKoRe Server.

322 The failure data collection starts when a machine failure occurs. The failure information collection  
 323 service generates the Human Machine Interface (HMI) for the user (Bob or Alice) to offer the details of  
 324 the failure Figure 6. Through the survey, we first try to know if the failure has really occurred or it  
 325 was only a false positive case triggered by some failure detection service. Then the user is asked to

326 provide details about the symptoms of the failure by selecting known symptoms or by creating new  
327 ones, when necessary.

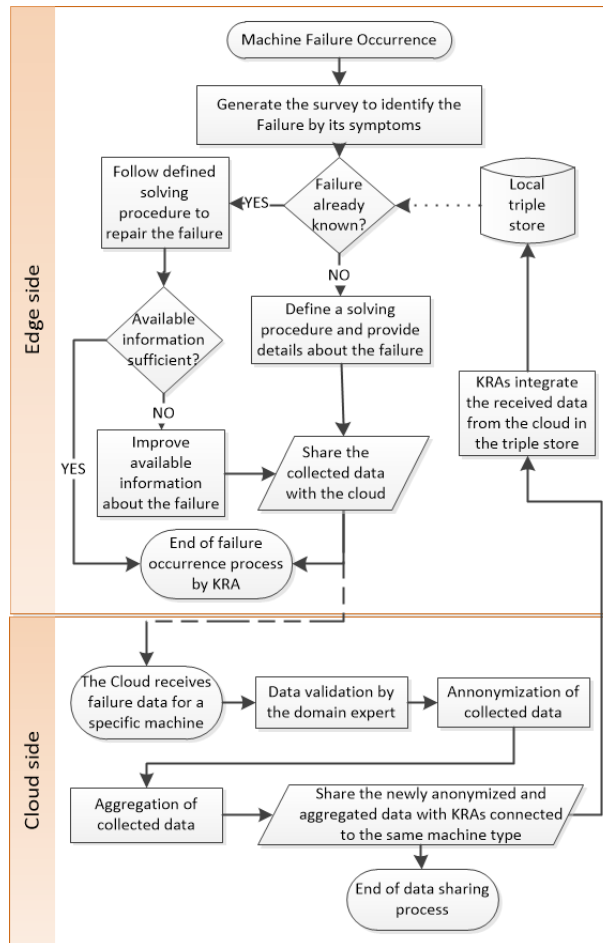


Figure 5. SemKoRe process.

**Failure detected**  
A failure is detected by our algorithm in the **ServoDrive 1** of the **Machine XYZ1**, please fill this survey

**Is it a real failure ?**  
 Yes  
 No, Everything is OK

**Did you observe any symptoms for this failure ?**  
 Yes  
 No

**What is the nature of the Symptom ?**  
 Abnormal Mechanical Behaviour

**Can you identify the component showing that symptom ?**  
 ----- ServoDrive1

**Can you describe concisely this symptom? (use simple sentences)**  
 High Vibration

**Are there other identified symptoms ?**  
 Yes  
 No

**Which severity level you estimate for this failure ?**  
 Low  
 Medium  
 Critical

**Is the Failure one of these choices?**  
 Servo Drive Electrostatic Discharge  
 Servo Drive Bearing Failure  
 Other

**Figure 6.** Screenshot of the failure survey.

328 The user checks if the identified failure is already known by the SemKoRe before providing  
 329 additional details. If the failure already exists, the user follows the instructions to repair the machine.  
 330 Otherwise, the failure will be documented by Alice or by machine domain experts as shown in Figure 5.  
 331 Sometimes, the impacts of a failure may differ from one machine to another. So, existing repair  
 332 procedures might be adapted or new procedures might be created to repair the machine. This process  
 333 ends, in the edge level, by sharing the collected data with the SemKoRe Server in the cloud.

334 When the SemKoRe Server receives failure data from a SemKoRe Agent instance, the data must  
 335 be validated by a machine domain expert before it is integrated into the SemKoRe Knowledge Graph.  
 336 After the validation process, data are anonymized to protect the data and customer privacy and  
 337 business sensitive information of the customers (see Section 6.3). The anonymized data are then  
 338 aggregated (see Section 6.4) in order to get insights about the occurrence frequency of failures, their  
 339 impacts and the most adapted/used repairing procedures .

340 The aggregated data are then shared with the SemKoRe Agents instances connected to the same  
 341 machine type. On the other side, each SemKoRe Agent instance integrates the data it receives from the  
 342 cloud into the local triplestores (Graph databases).

## 343 6. SemKoRe Implementation

344 In this section, we describe the services implemented for the SemKoRe. It must be noted that  
 345 the services deployed on Local SemKoRes are the same as the ones deployed on a SemKoRe Server.  
 346 As stated previously, we only focus on the architecture where gateways are directly connected to the  
 347 cloud (since its an overarching architecture and covers all the scenarios we described in Figure 1).

### 348 6.1. Startup Commissioning

349 Knowing that our Failure ontology will evolve, it is only deployed on the SemKoRe Server.  
 350 On-premises, the gateways must run the commissioning service in order to get the latest failure model  
 351 corresponding to the type of the connected machine. The startup commissioning service retrieves two  
 352 types of information from the SemKoRe Server:

- 353 • The machine failure T-Box, containing the concepts defined in the failure ontology; and,
- 354 • A-Box data, containing the instances of the T-Box concepts. Knowing that the SemKoRe Server  
355 manages data relative to several types of machines, the A-Box retrieved by a gateway contains  
356 only the information relative to the machines connected to it.

357 The startup commissioning service sends a request to the SemKoRe Server with the identity and  
358 the type of the connected machines. The SemKoRe Server runs then a Construct SPARQL query to  
359 create a sub-graph containing all the data (T-Box and the A-Box) related to the provided machine type.  
360 The resulting sub-graph is sent back to the gateway.

## 361 6.2. Failure Data Collection

362 This service runs exclusively in the SemKoRe Agent and is used during or after the maintenance  
363 phase. It consists of the following two parts:

### 364 6.2.1. Failure Survey

365 This part collects the failure information using an ordered set of predefined questions. It facilitates  
366 the collection of perceptible symptoms of the failures as well as the reporting of new symptoms and  
367 failures. During our interactions with machine domain experts, we found that the content of the survey  
368 is strongly correlated with the machine types and the kinds of failures they encounter. Our on-field  
369 interactions with the operators, technicians and experts highlighted the importance of an intuitive  
370 user interface.

### 371 6.2.2. Failure Ontology Instantiation

372 We use the Model Driven Interfaces (MDI) to dynamically generate the user interfaces using  
373 the Failure ontology. This procedure has two advantages: One, the UIs allow instantiation of the  
374 Failure ontology by non-technical users without any knowledge about the Semantic Web or ontologies;  
375 and Two, the UIs enforce the constraints defined in the ontology model and ensure that all the inputs are  
376 valid. These UIs rely on the annotations defined in the ontology such as, *@rdfs:label* and *@rdfs:comment*.  
377 The former is used as a human readable label of the input fields shown to the user, while the latter  
378 is displayed to explain the nature of the field and the expected input. We defined an additional  
379 annotation *@semkore:hidden* to hide a field on the UI in case it should be defined automatically or  
380 exclusively by an expert.

381 The ontology model incorporates two types of constraints: value and cardinality constraints,  
382 as described below.

#### 383 6.2.2.1 Value Constraints

384 In the W3C OWL reference [28], a value constraint is used to enforce restrictions on the range  
385 of a property when applied to a particular class description. Value constraints can be applied to  
386 data properties, for which the value is a data literal, and object properties, for which the value is an  
387 individual. We handle each type of property differently:

- 388 • Data properties: Users can input a value in the text box which will be validated to make sure that  
389 the data type is correct as per the defined constraints; and
- 390 • Object properties: A select box is provided with the list of all possible values. For example, for the  
391 object property *Machine hasComponent AllValuesFrom Component*, will lead to a select box with  
392 the list of all available **Component** instances. With this approach, the probability of getting an  
393 invalid input is eliminated altogether.

394 Only the *owl:someValuesFrom* constraint was managed differently from the W3C standard [28]  
395 definition, as it defines a constraint that is applied to at least one value, which means that the property  
396 could have other values without any restrictions. Since our UIs are targeting non-expert users, and to

397 guarantee the consistency of our model, we considered *owl:someValuesFrom* as being similar to the  
398 *owl:allValuesFrom* constraint in our implementation.

#### 399 6.2.2.2 Cardinality Constraints

400 In the W3C OWL reference [28], a cardinality constraint restricts the (min, max, or exact) number  
401 of values a data or object property can have. To satisfy the cardinality, single/multiple input fields  
402 are generated for each property, allowing to the user to provide the correct number of values for  
403 each property.

#### 404 6.3. Anonymization Service

405 Privacy protection is a very important concern for our customers. They do not want to share any  
406 machine and process-related data in a way that could potentially expose sensitive business information.  
407 To address this concern, we implemented a simple service (described below), in the SemKoRe Server  
408 in the cloud, to anonymize the collected data before sharing it with other sites or locations.

409 When a failure occurs, the gateway creates an instance of “Failure Occurrence Class”, containing  
410 information about the failure, e.g., symptoms, impact, root causes if known, and the failure context,  
411 which includes the machine ID, its location, timestamp when failure occurred, and a snapshot of the  
412 current parameters. The whole process consists of three steps:

- 413 1. The SemKoRe Server removes the machine ID, location, and owner-related information and does  
414 not share this information.
- 415 2. A human expert reviews and validates all of the failure information before integrating  
416 it into the SemKoRe Knowledge Graph. This additional check helps to protect sensitive  
417 business information.
- 418 3. Finally, all the validated failure information is aggregated and then shared with the connected  
419 gateways. This process ensures that no one can deduce the origin of the data, the failure location  
420 or the ownership details.

421 This service is a subject for future SemKoRe versions. The goal is to automate this process so that  
422 little to no human involvement is required.

#### 423 6.4. Failure Data Aggregation

424 Hosted in SemKoRe Server, this service and aggregates failure data collected from different  
425 gateways in order to produce deep insights on the machine failures and their characteristics including  
426 symptoms, impacts, and root causes. Once the aggregation is done, the data are shared by the SemKoRe  
427 Server with the connected gateways that need it. We have defined a list of simple aggregations that are  
428 applied to the failure data:

- 429 1. For each machine type, get the list of all failures and their frequency;
- 430 2. For each failure, compute the list of all possible symptoms with the frequency of each symptom;
- 431 3. For each failure, compute the list of all possible impacts with the frequency of each impact;
- 432 4. For each failure, compute the list of all possible root causes with the frequency of each root  
433 cause; and
- 434 5. For each failure, get the list of solutions with the number of times each solution was successfully  
435 used to repair that failure.

436 For each of these aggregations, a dedicated SPARQL query is executed and the results are injected  
437 into the Knowledge Graph. The aggregation service executes after every new data collection to keep  
438 the Knowledge Graph up-to-date.

### 439 6.5. Failure Data Sharing

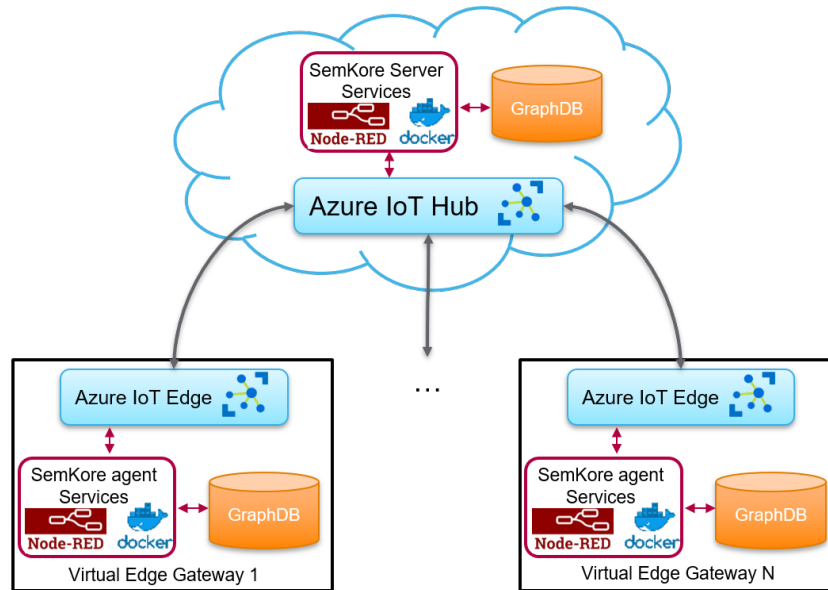
440 The failure data sharing is done through the SemKoRe Server's message broker. Each gateway  
441 subscribes to the topic ".../failure\_updates/{machine\_type}", where {machine\_type} is the type of machine  
442 to which the gateway is connected. When failure data are sent to the SemKoRe server (through the  
443 REST interface), it is validated, anonymized, aggregated and then published on the message topic  
444 corresponding to the right machine type. The gateways receiving these data will simply update the  
445 locally stored graph data.

### 446 6.6. Implementation Details

447 To demonstrate the feasibility of the SemKoRe approach, we developed a proof-of-concept  
448 (Figure 7) using the following technologies:

- 449 • GraphDB: We used GraphDB [29] as triplestore in the cloud and in the gateway. It provides high  
450 performance and scalability in addition to the reasoning capabilities.
- 451 • Node-Red : Node-Red was used to develop all above-mentioned services for the SemKoRe  
452 Server and the SemKoRe Agent. Node-Red is a flow-based development tool for visual  
453 programming originally developed by IBM for wiring together hardware devices, APIs  
454 (Application Programming Interfaces) and online services as part of the Internet of Things  
455 [30]. Node-Red is gaining popularity for rapid application development in Schneider's Industrial  
456 Automation business.
- 457 • Microsoft Azure: The SemKoRe Server services are hosted on Microsoft Azure, which provides  
458 high-performance cloud services. The server uses Azure IoT Hub [31] to connect the IoT  
459 devices to the cloud using several communication protocols, including the MQTT (Message  
460 Queuing Telemetry Transport) messaging protocol [32], which we used to simplify the data flow  
461 transmission between our SemKoRe agent and server services. For the edge, the Azure IoT Edge  
462 service was used to easily connect the edge gateways to the cloud via Azure IoT Hub as shown in  
463 Figure 7.
- 464 • Docker: This is a popular container-based virtualization [33] tool. Docker supports many  
465 operating systems and hardware architectures, and allows self contained applications to be  
466 packaged and executed with a high level of portability and reproducible results [34]. We used  
467 Docker to package all of the services of SemKoRe Server and SemKoRe Agent. In reality  
468 SemaKoRe agent will reside with many other components. Docker allows us to have the flexibility  
469 of deploying different components easily and manage/extend them without impacting others.





**Figure 7.** SemKoRe implementation setup. Simulated virtual edge gateways with SemKoRe Agents managed by a SemKoRe server instance hosted on the Microsoft Azure cloud.

470 As we are still in early stage, we were not able to implement the SemKoRe in real conditions with  
 471 SemKoRe Agent running on Industrial Gateways connected to real machines. The main obstacle was  
 472 that the current hardware available for this work had ARM architecture and so it would require a  
 473 significant effort to port the triplestore and other software components. That effort was beyond the  
 474 scope of our work. However, after the successful PoC, the business team decided to use an industrial  
 475 PC as a gateway with enough RAM (8GB), processing (Intel Atom) and storage (64GB) capability.  
 476 The next iteration of this work will use this industrial PC when it is ready for commercialization. In the  
 477 meantime, we evaluated our implementation by simulating many virtual SemKoRe Agent instances  
 478 on a PC equipped with an Intel(R) Core(TM) i7-7820HQ processor, and 16Gb of RAM. We used a single  
 479 GraphDB server to manage separate triplestores for all the SemKoRe Agent instances. We randomly  
 480 defined a set of three machine types {Packaging Machine, Palletizing Machine, Pasteurization Machine}  
 481 that we associated to the active SemKoRe Agents. Each machine type was associated with at least  
 482 two SemKoRe Agent instances which were then connected to the cloud SemKoRe Server. Our current  
 483 implementation choices will facilitate an easy transition to industrial PCs in future.

484 We generated random machine failure data by defining a set of potential failures for each machine  
 485 type. Each failure was then associated to a set of potential characteristics: symptoms, impacts,  
 486 root causes and solving procedures. For each machine failure occurrence, we randomly picked one of  
 487 the potential failures of the machine, and then picked a random number of the failure characteristics  
 488 from the predefined sets.

489 We were able to demonstrate that the failure data were collected by each SemKoRe Agent and  
 490 successfully shared with the SemKoRe Server. The data were aggregated and shared back with the  
 491 SemKoRe Agent instances connected to the same machine type.

## 492 7. Conclusions and Future Work

### 493 7.1. Learned Lessons

494 Conducting this study helped us to learn several lessons. The first lesson is that the use of  
 495 semantic web technologies to solve complex industrial problems is still a largely unexplored area.  
 496 Even today most of the solutions on the market focus on the enterprise and IT side than on the  
 497 operational side of large industries. This means that there are mature solutions that use semantic web  
 498 technologies to bridge siloed enterprise data in RDBMS (Relational Database Management System)

499 and unstructured data like documents but there is no mature solution that can do the same for data  
500 described in operation technology protocols, e.g., OPC-UA(Open Platform Communications - Unified  
501 Architecture) [35].

502 The second lesson is that technologies such as triplestore are not easily adoptable to typical  
503 industrial use cases. Almost all triplestores are focused on big data and huge numbers of triples but,  
504 as our work demonstrates, there are several use cases where an efficient solution is needed for typical  
505 industrial gateways. Industrial PCs are an option but they are expensive and can only be used by  
506 large companies whereas small devices have a very large user base. While machine failures are reality,  
507 they do not occur every minute, and so there is no need to use a complex solution that supports billions  
508 of triples. Outside the vendor space, the open source community has some options like RedStore [36]  
509 but most are not in active development.

510 The third lesson is that the development of industrial grade ontologies is still a herculean task  
511 and the existing tool set continues to act as a barrier to entry. In our experience, experts want to  
512 formalize their domain knowledge but they have no motivation to learn complex tools such as Protege  
513 that do not support collaborative ontology development. WebProtege is a possibility but lacks query,  
514 visualization and documentation capabilities. New efforts such as Modom.io [37] and Zazuko [38] take  
515 a more simplified approach for non-experts to create ontologies but they are still works in progress.

516 Regarding ontology governance, there is no standard framework that can be applied to design and  
517 develop modular ontologies on an industrial scale. The evolution of ontologies is another area where  
518 no clear recommendations and no industrial tools are available to manage the required documentation,  
519 evaluation, release and versioning. While some academic works such as [39] exist, they are not mature  
520 and often not easy to deploy and use in industrial settings. The Semantic Web community as a whole  
521 needs to address these points and improve the developer experience in order to mainstream these  
522 useful technologies.

## 523 *7.2. Future Potential*

524 We have also identified several avenues as the future potential to continue this work. They are  
525 mentioned here without any order of priority.

### 526 *7.2.1. Machine Learning for Data Anonymization*

527 The first item is to explore the use of machine learning for content anonymization services.  
528 In this work, we used a simple approach with validation by a human expert. However, a far more  
529 efficient approach would be to investigate the use of artificial intelligence and machine learning to  
530 anonymize data based on several contexts. The state-of-the-art anonymization techniques achieve  
531 good precision scores (up to 98%), which will make it unnecessary to involve humans. One might  
532 consider to collect anonymous data from the beginning to avoid the anonymization overhead and the  
533 complexity. However, with this approach we will miss important data that might be useful for things  
534 like audit (machine id, maintenance history, configuration) and other applications.

### 535 *7.2.2. GraphDB Multi-Tenant Support*

536 The second item is that currently, one GraphDB tenant is used in the cloud to collect the data of all  
537 customers, which could become an issue for scalability and data privacy. A potential solution may be  
538 to have separate GraphDB tenants for each customer and then create a common GraphDB instance to  
539 collect anonymized and aggregated data from the other customers' instances. Managing these tenants  
540 and synchronizing them will be big challenges.

### 541 *7.2.3. Lightweight Triplestores*

542 The third area would be to work on lightweight triplestores for small industrial devices.  
543 Many triplestores for embedded/small platforms exist. Most of them are based on the Redland

544 RDF Libraries [40], and are using SQLite as backend storage (e.g., RedStore [36]). However, all these  
545 solutions lack reasoning engines and do not support SWRL rules.

#### 546 7.2.4. Knowledge Graphs Synchronization

547 Ensuring the knowledge synchronization between the SemKoRe Server and SemKoRe Agent is  
548 the fourth area. As mentioned before, not all customers are keen to have a cloud connection or can  
549 have always-on connection. Therefore, it is necessary to define a synchronization process to ensure  
550 that there is no inconsistent knowledge.

#### 551 7.2.5. UI Enhancement

552 The fifth item of future work is that today UIs (User Interfaces) are used to report machine failures  
553 through manual input from persons like Bob or Alice. This process can be enhanced by using AI/ML  
554 algorithms that observe the symptoms and prefill the UI form with accurate details. This can be further  
555 extended to automatically fetch the repair instructions from a SemKoRe graph before failures occur.

#### 556 7.2.6. Ontology Extension by Non-Expert Users

557 In this work, we target a large set of customers from various domains and with different needs.  
558 We are not expected to create or to modify ontology for each and every customer. Therefore, the sixth  
559 item is to develop a framework along with a tool suite and set of services for non-experts to allow  
560 them to create and extend their ontology models. We will also need to address more advanced topics  
561 like ontology matching, alignment, and conflict resolution to ensure consistency.

#### 562 7.2.7. Use of Upper-Level Ontologies

563 Another future item to consider is the use of upper-level ontologies as a basis of the failure and  
564 the machine domain ontologies. There is a separate on-going work to decide on the right ontology  
565 to provide maximum data integration for the future. For example, today the discussion revolves  
566 around using either BFO<sup>2</sup> or ISO15926<sup>3</sup> or Gist<sup>4</sup> as upper-level ontology. Some customers may also  
567 be interested in using domain ontologies like SSN<sup>5</sup>. Our view on this point is that once a decision is  
568 made, our current ontology can be easily refactored.

### 569 7.3. Conclusions

570 In this paper, we proposed a knowledge graph-based approach, SemKoRe, to enhance the  
571 maintenance process for the customers of Machine Builder OEMs. The idea consists of collecting  
572 machine failure data generated by different machines owned by many customers in different locations  
573 and in different business segments. The SemKoRe approach helps reduce the maintenance costs by  
574 sharing maintenance experiences between OEM customers. Based on this early work, our customers  
575 showed an interest in using the SemKoRe approach to enhance their industrial maintenance processes.  
576 Furthermore, by using the SemKoRe approach, the overall machine building process can be optimized.  
577 The machine design phase can benefit from the maintenance feedback to identify any weaknesses  
578 of a machine and can improve its design. Furthermore, the collected statistics will allow the  
579 performance comparison of a particular machine working in different locations and contexts. Thus,  
580 additional services and recommendations can be proposed to the customers in order to optimize  
581 their manufacturing process. Some customers also feel that our approach can help them to build

---

<sup>2</sup> <https://basic-formal-ontology.org/>

<sup>3</sup> <https://15926.org/home/>

<sup>4</sup> <https://www.semanticarts.com/gist/>

<sup>5</sup> <https://www.w3.org/2005/Incubator/ssn/ssnx/ssn>

582 Digital Twins to monitor the performance and efficiency of their machines. As mentioned in the Future  
583 Potential section, we plan to investigate on several work items to improve the features of SemKoRe.

584 **Author Contributions:** Conceptualization, H.H., A.T.-I.; methodology, H.H., I.K.; software, H.H.; validation,  
585 H.H., I.K. and A.T.-I.; investigation: H.H.; writing—original draft preparation, H.H., I.K., M.A.; writing—review  
586 and editing, I.K., M.A., N.C.; supervision: I.K., A.T.-I., N.C.; project administration, A.T.-I., I.K. All authors have  
587 read and agreed to the published version of the manuscript.

588 **Funding:** This research received no external funding.

589 **Conflicts of Interest:** The authors declare no conflict of interest.

## 590 References

- 591 1. Aazam, M.; Zeadally, S.; Harras, K.A. Deploying fog computing in industrial internet of things and  
592 industry 4.0. *IEEE Trans. Ind. Inform.* **2018**, *14*, 4674–4682.
- 593 2. Cachada, A.; Barbosa, J.; Leitão, P.; Gcraldcs, C.A.; Deusdado, L.; Costa, J.; Teixeira, C.; Teixeira, J.;  
594 Moreira, A.H.; Miguel, P.; et al. Maintenance 4.0: Intelligent and Predictive Maintenance System Architecture.  
595 In Proceedings of the IEEE 23rd International Conference on Emerging Technologies and Factory Automation  
596 (ETFA), Turin, Italy, 4–7 September 2018; Volume 1, pp. 139–146.
- 597 3. Swanson, L. An information-processing model of maintenance management. *Int. J. Prod. Econ.* **2003**, *83*,  
598 45–64.
- 599 4. Lopez, J.; Alcaraz, C.; Roman, R. Smart control of operational threats in control substations. *Comput. Secur.*  
600 **2013**, *38*, 14–27.
- 601 5. Berners-Lee, T.; Hendler, J.; Lassila, O. The semantic web. *Sci. Am.* **2001**, *284*, 34–43.
- 602 6. Alcaraz, C.; Wolthusen, S. Recovery of structural controllability for control systems. In Proceedings of  
603 the International Conference on Critical Infrastructure Protection, Arlington, VA, USA, 17–19 March 2014;  
604 Springer: Berlin/Heidelberg, Germany, 2014; pp. 47–63.
- 605 7. Alcaraz, C.; Lopez, J. Analysis of requirements for critical control systems. *Int. J. Crit. Infrastruct. Prot.* **2012**,  
606 *5*, 137–145.
- 607 8. Kharlamov, E.; Grau, B.C.; Jiménez-Ruiz, E.; Lamparter, S.; Mehdi, G.; Ringsquandl, M.; Nenov, Y.; Grimm, S.;  
608 Roshchin, M.; Horrocks, I. Capturing Industrial Information Models with Ontologies and Constraints.  
609 In Proceedings of the International Semantic Web Conference, Kobe, Japan, 17–21 October 2016.
- 610 9. Horridge, M.; Gonçalves, R.S.; Nyulas, C.; Musen, M.A. WebProtege: A Cloud-Based Ontology Editor. In  
611 Proceedings of the WWW'19: Companion 2019 World Wide Web Conference, San Francisco, CA, USA, 13–17  
612 May 2019.
- 613 10. Zaini, N.; Omar, H. An online system to support collaborative knowledge acquisition for ontology  
614 development. In Proceedings of the 2011 IEEE International Conference on Computer Applications and  
615 Industrial Electronics (ICCAIE), Penang, Malaysia, 4–7 December 2011; pp. 543–548.
- 616 11. Nuñez, D.L.; Borsato, M. An ontology-based model for prognostics and health management of machines.  
617 *J. Ind. Inf. Integr.* **2017**, *6*, 33–46.
- 618 12. Horrocks, I.; Patel-Schneider, P.F.; Boley, H.; Tabet, S.; Grosz, B.; Dean, M. SWRL: A semantic web rule  
619 language combining OWL and RuleML. *W3C Memb. Submiss.* **2004**, *21*, 1–31.
- 620 13. Melik-Merkumians, M.; Zoitl, A.; Moser, T. Ontology-based fault diagnosis for industrial control applications.  
621 In Proceedings of the 15th Conference on Emerging Technologies & Factory Automation (ETFA 2010),  
622 Bilbao, Spain, 13–16 September 2010.
- 623 14. Palacios, L.; Lortal, G.; Laudy, C.; Sannino, C.; Simon, L.; Fusco, G.; Ma, Y.; Reynaud, C. Avionics maintenance  
624 ontology building for failure diagnosis support. In Proceedings of the 8th International Joint Conference on  
625 Knowledge Discovery, Knowledge Engineering and Knowledge Management (IC3K 2016), Porto, Portugal,  
626 9–11 November 2016; Volume 2, pp. 204–209.
- 627 15. Peng, H.; Chang, D.; Wang, Y. Fault Diagnosis of Conveyor Based on Ontology *Sens. Transducers* **2013**, *157*,  
628 338–345.
- 629 16. Chen, R.; Zhou, Z.; Liu, Q.; Pham, D.T.; Zhao, Y.; Yan, J.; Wei, Q. Knowledge modeling of fault diagnosis  
630 for rotating machinery based on ontology. In Proceedings of the IEEE 13th International Conference on  
631 Industrial Informatics (INDIN), Cambridge, UK, 22–24 July 2015; pp. 1050–1055.

- 632 17. Xu, F.; Liu, X.; Chen, W.; Zhou, C.; Cao, B. Ontology-based method for fault diagnosis of loaders. *Sensors*  
633 **2018**, *18*, 729.
- 634 18. Wan, S.; Li, D.; Gao, J.; Roy, R.; Tong, Y. Process and knowledge management in a collaborative maintenance  
635 planning system for high value machine tools. *Comput. Ind.* **2017**, *84*, 14–24.
- 636 19. Mourtzis, D.; Vlachou, A.; Zogopoulos, V. Cloud-based augmented reality remote maintenance through  
637 shop-floor monitoring: A product-service system approach. *J. Manuf. Sci. Eng.* **2017**, *139*, 061011.
- 638 20. Xenakis, A.; Karageorgos, A.; Lallas, E.; Chis, A.E.; González-Vélez, H. Towards distributed IoT/cloud based  
639 fault detection and maintenance in industrial automation. *Procedia Comput. Sci.* **2019**, *151*, 683–690.
- 640 21. Mourtzis, D.; Vlachou, E.; Milas, N.; Xanthopoulos, N. A cloud-based approach for maintenance of machine  
641 tools and equipment based on shop-floor monitoring. *Procedia Cirp* **2016**, *41*, 655–660.
- 642 22. Mourtzis, D.; Vlachou, E. A cloud-based cyber-physical system for adaptive shop-floor scheduling and  
643 condition-based maintenance. *J. Manuf. Syst.* **2018**, *47*, 179–198.
- 644 23. Schmidt, B.; Wang, L. Cloud-enhanced predictive maintenance. *Int. J. Adv. Manuf. Technol.* **2018**, *99*, 5–13.
- 645 24. Alcaraz, C.; Agudo, I.; Nunez, D.; Lopez, J. Managing Incidents in Smart Grids à la Cloud. In Proceedings of  
646 the 2011 IEEE Third International Conference on Cloud Computing Technology and Science, Athens, Greece,  
647 29 November–1 December 2011; pp. 527–531, doi: 10.1109/CloudCom.2011.79.
- 648 25. Noy, N.F.; McGuinness, D.L. *Ontology Development 101: A Guide to Creating Your First Ontology*; Technical  
649 Report KSL-01-05 and SMI-2001-0880, Stanford Knowledge Systems Laboratory and Stanford Medical  
650 Informatics, 2001.
- 651 26. Pellet Reasoner. Available online: <https://www.w3.org/2001/sw/wiki/Pellet> (accessed on 24 April 2020).
- 652 27. Noy, N.F.; Sintek, M.; Decker, S.; Crubézy, M.; Ferguson, R.W.; Musen, M.A. Creating semantic web contents  
653 with protege-2000. *IEEE Intell. Syst.* **2001**, *16*, 60–71.
- 654 28. Web Ontology Language Reference. Available online: <https://www.w3.org/TR/owl-ref> (accessed on  
655 24 April 2020).
- 656 29. GraphDB. Available online: <http://graphdb.ontotext.com> (accessed on 24 April 2020).
- 657 30. Node-Red. Available online: <https://nodered.org> (accessed on 24 April 2020).
- 658 31. Azure IoT Hub. Available online: <https://azure.microsoft.com/en-us/services/iot-hub> (accessed on  
659 24 April 2020).
- 660 32. Banks, A.; Gupta, R. MQTT Version 3.1.1. *OASIS Stand.* **2014**, *29*, 89.
- 661 33. Docker: Empowering App Development for Developers. Available online: <https://www.docker.com>  
662 (accessed on 24 April 2020).
- 663 34. Boettiger, C. An introduction to Docker for reproducible research. *ACM SIGOPS Oper. Syst. Rev.* **2015**, *49*,  
664 71–79.
- 665 35. Bruckner, D.; Stănică, M.; Blair, R.; Schriegel, S.; Kehrer, S.; Seewald, M.; Sauter, T. An introduction to OPC  
666 UA TSN for industrial communication systems. *Proc. IEEE* **2019**, *107*, 1121–1131.
- 667 36. Haslhofer, B.; Roochi, E.M.; Schandl, B.; Zander, S. *Europeana RDF Store Report*; Austrian National Library,  
668 University of Vienna, Vienna, Austria, 2011.
- 669 37. Modom.io. Available online: <https://modom.io> (accessed on 24 April 2020).
- 670 38. Zazuko Ontology Manager. Available online: <https://zazuko.com/products/ontology-manager>  
671 (accessed on 24 April 2020).
- 672 39. Alobaid, A.; Garijo, D.; Poveda-Villalón, M.; Santana-Pérez, I.; Fernández-Izquierdo, A.; Corcho, Ó.  
673 Automating ontology engineering support activities with OnToology. *J. Web Semant.* **2019**, *57*, 100472.
- 674 40. Redland RDF Libraries. Available online: <http://librdf.org> (accessed on 24 April 2020).

Annex **B**

Paper II

Department: Head  
Editor: Name, xxxx@email

# Data Anonymization for Maintenance Knowledge Sharing

**Hicham Hossayni**

Schneider Electric, Grenoble, France

**Imran Khan**

Schneider Electric, Grenoble, France

**Noel Crespi**

Institut Polytechnique de Paris, Paris, France

**Abstract**—Formerly considered as part of general enterprise costs, industrial maintenance has become critical for business continuity and a real source of data. Despite the heavy investments made by companies in smart manufacturing, traditional maintenance practices still dominate the industrial landscape. Maintenance knowledge sharing between industries can significantly optimize the maintenance activity and improve the processes efficiency. Different international standards and initiatives are promoting such approach. However, this trend failed to gain ground in the manufacturing industry. In this paper, we present the results of our investigation about the real roadblocks that obstruct the progress of the maintenance knowledge sharing approach. We determined that the knowledge graphs and, more importantly, the automated data anonymization techniques can facilitate the development of general-purpose solutions to share the maintenance knowledge among concerned actors.

**Index Terms:** Data anonymization, industrial maintenance, knowledge sharing

■ **INTRODUCTION** Over the past few decades, knowledge sharing has become both democratized and essential for individuals and professionals. It is a go-to approach in many domains like in computer software through open-source software

and tools, open data-sets, crowd-sourcing and collective universal encyclopedia (Wikipedia is just one example). However, this trend has not gained traction in the manufacturing industry yet, especially for the maintenance activity. Typically,

industrial maintenance knowledge is inaccessible due to industry policies and practices, with little to no motivation to share it with others. Today, when a new machine is installed in a factory, there is no existing knowledge of its failures unless there is a human expert who has dealt with similar machines before. In any given factory, each failure is only discovered at its first occurrence and requires usually a long and costly curative maintenance operation. The maintenance process includes diagnostics to determine the reasons for a failure, its impact, and to define and apply the correct repair procedures. Therefore, the same failure of a machine, installed in many factories, could lead to high costs and big production losses because every factory, or even every maintenance operator deals with the failure individually, from the diagnostics phase to the problem resolution. A potential solution that could help improve the situation would be to enable the sharing of maintenance knowledge and experiences between the industries or factories that own and operate the same types of machines. The shared knowledge could be used by the machine operators as a guide to reduce the diagnostic and repair times, target the failures' root cause(s), and improve the machines' efficiency. Another important effect of maintenance knowledge sharing is that it will transfer maintenance expertise between experts and novices. This will lead to much faster handling of machine failures, training of new staff and to reduce the dependency on external maintenance companies. Even more importantly, it will reduce the number of accidents related to maintenance activities.

### Existing maintenance data exchange standards & projects

There are many international standards and consortium agreements that promote and adopt the maintenance knowledge sharing approach. Some of the most well-known ones are:

- The OREDA project [1]: Offshore and Onshore Reliability Data for oil and gas industries, which led to the ISO 14224 international standard [2]. It is a project organization with 7 to 11 oil and gas companies as members, managing data of 292 installations and 18000 equipment units. Running for more than 35

years, OREDA has led to significant cost savings in the development and operation of platforms <sup>1</sup>, and has helped has helped the participating oil companies to save \$70M. <sup>2</sup>.

- SPARTA [3]: System Performance, Availability and Reliability Trend Analysis adopted by 9 wind energy companies in the UK. It manages and exchanges data of 19 wind farms and 1256 turbines in 2020. It covers more than 60% of all offshore wind power generation in the UK. Thanks to SPARTA, the average number of crew transfers per turbine in the UK fell by 50% between 2014 and 2018, to around six trips per year <sup>3</sup>.
- WInDPool [4]: stands for Windenergy-InformationData-Pool, for wind energy industries in Germany. Adopted by 7 members to exchange the maintenance data for a fleet of 640 wind turbines onshore and 297 wind turbines offshore with an expectation of more than 1M€ of costs savings on the maintenance activity <sup>4</sup>.
- The Configuration Data Exchange [5]: launched by General Electric Aviation for the world-wide aviation industry. \$4 billion savings are anticipated across the sector thanks to the configuration data exchange solution <sup>5</sup>.
- OPDE [6]: The International Pipe Failure Data Exchange project; and
- ISO 6527 [7]: A Reliability Data Sharing standard for nuclear energy producers.

Through these standards and applications, the maintenance knowledge sharing concept has proven its efficiency in the improvement of equipment reliability, in the enhancement of maintenance processes and in the reduction of production costs, leading to savings over a machine's life cycle. However, each of these standards or applications targets a specific domain and has

<sup>1</sup><http://www.datsi.fi.upm.es/~rail/new/WP2/OREDA-history.htm> (accessed 17 Feb, 2021)

<sup>2</sup><https://www.oreda.com/join-us> (accessed 17 Feb, 2021)

<sup>3</sup><https://ore.catapult.org.uk/wp-content/uploads/2018/11/SPARTA-Portfolio-Review-201718-1.pdf> (accessed 17 Feb, 2021)

<sup>4</sup>[https://wind-pool.iec.fraunhofer.de/opencms/export/sites/WInD-Pool/img/WInD-Pool-Business-Case\\_ENG.pdf](https://wind-pool.iec.fraunhofer.de/opencms/export/sites/WInD-Pool/img/WInD-Pool-Business-Case_ENG.pdf) (accessed 17 Feb, 2021)

<sup>5</sup><https://www.businesswire.com/news/home/20161115005705/en/GE-Aviation-Launches-Configuration-Data-Exchange-Reduce> (accessed 17 Feb, 2021)



been adopted by only a handful of participants who exchange their data under multilateral agreements.

## Survey of maintenance knowledge-sharing challenges

To understand the perspectives of different actors involved in industrial maintenance operations, we conducted a survey through a set of direct interviews. Plant manager, production manager, machine operators, maintenance engineers, and different other industrial profiles were interviewed. We targeted 30 companies from various domains (Engineering, pharmaceuticals, automotive, HVAC, Food & Beverage, ...etc.) and from 6 countries (UK, Spain, France, Switzerland, China and US). The survey was limited to manufacturing industries for which the equipment maintenance is a major activity. On one hand, the survey brought-forward several issues that are faced by these actors in their daily routine. We found real interest in a maintenance knowledge sharing solution and these actors believed that it will be beneficial in the short term as well as in the long term. On the other hand, the survey allowed us to identify the various challenges that could impede the development of generic maintenance knowledge sharing solutions. Some of the most important ones are:

- 1) **Business culture:** Sharing knowledge is not a common activity in most organizations. It has been demonstrated that individuals are rewarded mostly for what they know, and not what they share [8]. The competitive environment that encourages individual instead of collective productivity has taught employees to consider their knowledge as their own property, and that to deepen and defend their knowledge is the main way to keep their jobs within the organization.
- 2) **Maintenance data collection:** The collection of good quality maintenance data<sup>6</sup> is a mandatory step before being able to extract and share useful maintenance knowledge<sup>7</sup>

<sup>6</sup>Maintenance data refers to maintenance reports established to document a maintenance operation. These may contain data about the machine's components, description of the failure and details of the maintenance procedure.

<sup>7</sup>By maintenance knowledge we refer to the result of an aggregation of maintenance data collected by many entities that is all relative to the same machine type.

from it. It is a long-term activity, requiring the involvement of competent and well-trained personnel to guarantee the quality and usefulness of the collected data. However, this is usually seen as a marginal and costly process instead of being considered as a solid investment for the future.

- 3) **Human factors:** Many people may be reluctant to write a detailed report when they are not confident in their own expertise and want to avoid being judged. Also, when failure frequency is high, many maintenance operators believe there is no time to write a detailed report and thus may produce minimal reports with little or no transferable knowledge.
- 4) **Common maintenance taxonomy:** Collecting good quality maintenance data is not enough to make it shareable. Entities exchanging their maintenance data must have the same understanding of the shared data and hence, use the same vocabulary or taxonomy.
- 5) **Legal aspects:** Every maintenance report is a potential legal liability. In fact, a report's producer is legally responsible for material damages that may be caused by the application of his or her instructions. The responsibility could be penal in the case of human damages. Due to this issue of responsibility, some large companies destroy their maintenance reports beyond a certain legal period (e.g., 2 years in Europe) to avoid any future problems associated with their reports.
- 6) **Sensitive data disclosure:** Maintenance data may contain sensitive information that might compromise or negatively impact a company's activity. Thus, companies often choose to keep all their data secret to ensure business stability.

All the standards and projects presented above try to address aspects of these challenges. The existence of international standards that rule the data sharing activity may, itself, has a positive impact on challenges 1,2, and 3, since the existing successful experiences are sufficient to influence the business culture and processes and indirectly reduce the human factors risk. Challenge 4 is ex-

Explicitly handled by the definition of standard exhaustive taxonomies that describe the equipment components, failure details, and the maintenance procedures of the maintenance key performance indicators (KPIs). This is already possible where the standards are defined for specific industry fields (e.g., oil & gas) with well-known types of equipment. However, this solution cannot be adopted to cover all industries. Consortium agreements are a solution for the legal aspect (challenge 5), as data is contractually agreed to be shared by declining all responsibility relative to its use. Nevertheless, this approach is not scalable and not efficient when targeting a large set of companies. Finally, the sixth challenge is being addressed by the cited standards by their recommendations to apply data anonymization. This process aims to remove all sensitive data from the maintenance knowledge before it is shared. In practice, the anonymization of maintenance data is usually done manually, as no automated approach has been adopted or promoted by the standards and applications.

### SemKoRe for maintenance data sharing

In our previous work we proposed "SemKoRe" [9], a technical solution for maintenance data sharing, with which we aimed to tackle challenges 4, 5, and 6. SemKoRe targets the Original Equipment Manufacturers (OEMs) that produce and sell machines to other industries. An OEM can offer SemKoRe as a supplementary service, connected to a machine, that simplifies the collection and reuse of maintenance data, and then to share the growing body of maintenance knowledge with customers that own the same machines types. Our customers showed an interest in using the SemKoRe approach to enhance their industrial maintenance processes. Moreover, the overall machine building process can be optimized. The machine design phase can benefit from the maintenance feedback to identify any weaknesses of a machine and can improve its design. Furthermore, the collected statistics will allow the performance comparison of a particular machine working in different locations and contexts. Thus, additional services and recommendations can be proposed to the customers in order to optimize their

manufacturing process.

Technically, instead of proposing an exhaustive taxonomy for all types of equipment, machines or failures for a specific industry field, our approach defines individual taxonomies for each machine type. In fact, SemKoRe defines an ontology-based taxonomy to describe the invariant concepts in maintenance activity. This taxonomy is then extended by the OEMs to build machine-specific taxonomies that describe the exhaustive vocabulary relative to each machine. Moreover, SemKoRe proposes ontology-based adaptive UIs to simplify the filling of failures or maintenance details with more predefined inputs than free texts, which may help to increase the quality of the collected data. To tackle the challenges 5 and 6, we recommend the adoption of an automated anonymization approach [10]. It consists of applying automated tools to remove the sensitive data within maintenance reports so that they can be shared securely between different companies. The removal of sensitive data also removes all information about its origin, and thus the resulting anonymized data assures that no legal responsibility could be engaged against its producer (challenge 5). In the literature, existing automated anonymization solutions for structured or text data achieve high accuracy scores (around 98% [11]). This level is judged to be sufficient [12] to address the 6<sup>th</sup> challenge (sensitive data disclosure) as the remaining 2% of non-anonymized entities cannot be statistically distinguished from the anonymized ones. In the next section, we describe the automated data anonymization approach in detail. This solution has proven to be the most important enabler of SemKoRe for OEMs and their customers.

### Automated anonymization of maintenance data

In the digital era, big data and massive processing capabilities allow us to explore new possibilities. They have become essential for the optimization of our activities and resources, and help us to discover new opportunities. However, the ubiquity of personal information or PII (personally identifiable information) makes this task very challenging [13]. Data anonymization originated from the need to depersonalize processed information while guaranteeing the usefulness

and knowledge contained in that information. Around the world, multiple regulations require the anonymization of data before it can be processed, especially in the financial services or the medical field, which is the leader in terms of privacy preservation needs [14]. However, to the best of our knowledge, no initiative for industrial maintenance data anonymization has emerged to date. Several data anonymization techniques can be found in the literature [15], ranging from simple dictionary searches to Deep Learning-based techniques. The main objective of these techniques is to detect sensitive information in a text and classify it according to its nature. Machine learning techniques have shown their absolute superiority in this task. Other methods of pattern-matching or dictionary lookups are complementary and allow to simplify and focus learning models in difficult cases. It is very important to consider the nature of the data to be anonymized. Data can be classified as structured and unstructured (i.e., text data). Structured data is already tagged data that can be stored in a relational database, or in a structured file such as: JSON, XML, CSV, spreadsheets, ... etc. Unstructured text data might contain any type of text information. In our case, industrial maintenance data is usually a mix of structured data (e.g., dates, machine identifiers, sites, addresses, maintenance operator identities) and unstructured text data (e.g., diagnostic steps, solving procedure, and maintenance operators' observations). Both types of data could potentially contain sensitive information relative to the manufacturing process, production details, machine configuration, or other sensitive data.

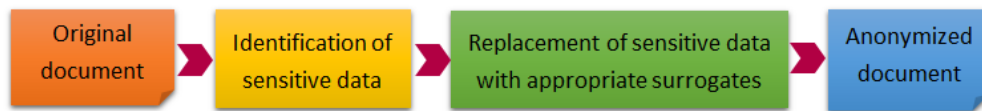
#### Data sensitiveness determination

An important step for data anonymization is to determine sensitive data and decide if it needs to be removed/replaced or kept without any change. Maintenance reports may contain two types of sensitive data: Personal data and Business data. Personal data or Personally Identifiable Information (PII) includes all data that may refer to a person directly (their name, email address, phone number, social insurance number, etc.), or indirectly, known as quasi-identifiers (such as their birth date, zip code, profession, or gender). Only one direct identifier is needed to identify a person,

while a combination of two or more indirect identifiers is required for the same purpose. Business-sensitive data is any information that poses a risk to the company if discovered. Examples of such data include financial data, supplier and customer information, manufacturing secrets, etc. Our survey showed that judging a data item as being business-sensitive varies from one company to another. For example, some users considered that the configuration of their machines was not sensitive, while others (e.g., a flavor manufacturing company, or a tire company) believe that their machine configuration is a manufacturing secret and thus a part of their industrial property, and so it cannot be shared with others. In the industrial context, deciding which data is sensitive and which is not is a challenging task. Consensus on business data sensitiveness among all the actors appears to be impossible to achieve. This situation reinforces the need for an anonymization system with a large set of capabilities to cover all customers' needs, and that allows users to define their sensitive data types as configuration.

#### Structured data anonymization

The purpose of structured data anonymization methods is to prevent the re-identification of sensitive data from an anonymized dataset by means of matching with external data sources [16]. These methods are classified into two categories: perturbative and non-perturbative methods [17]. Perturbative methods alter the data set to introduce uncertainty around the true values, while non-perturbative methods reduce the details in data by imposing generalization or by suppressing certain values without distorting the data structure. Some of the most well-known structured data anonymization techniques are: Local suppression, Micro-aggregation, Noise-addition and Swapping. These techniques try to satisfy some statistical properties, such as K-Anonymity, l-diversity or t-closeness. The validation of these properties does not mean that the privacy is 100% preserved, but they are a good indicator of the quality of the anonymization process. Some examples of structured data anonymization tools are IBM's ARX Data Anonymization Tool, Amnesia, the Cornell Anonymization Toolkit (CAT), and Aircloak Insights.



**Figure 1.** Text anonymization process.

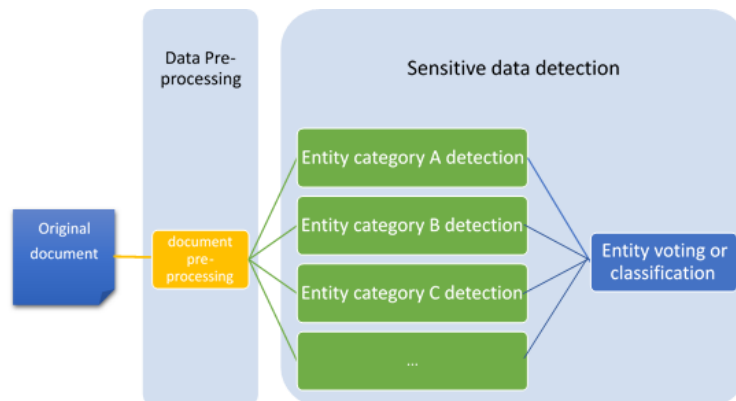
Text anonymization process overview

**Figure 1** shows an overview of the process of text anonymization. It is composed of two main tasks. The first consists of identifying and classifying the sensitive data within a text. Both personal and business sensitive data are detected during this phase. The second step determines the appropriate surrogates with which to replace the sensitive data detected during the previous phase, thereby producing an anonymized document that still contains valuable data. The next sections present some tools and techniques that are used for both tasks.

Identification of sensitive data process

Different approaches are used for sensitive data detection within a text, all of which are based on the text analysis and rely mainly on Natural Language Processing (NLP) techniques. As shown in **Figure 2**, this process follows the standard schema of an ML Ensembling pipeline. It begins with the pre-processing of the input data before it is simultaneously fed to different classifiers. Each classifier is responsible for detecting a specific type of sensitive data.

This approach is adopted by the Named Entity Recognition (NER) tools [18]. NER consists of



**Figure 2.** NLP-based sensitive data detection process

**Table 1. Examples of sensitive data surrogates.**

Sensitive data	Examples of surrogates
<b>Date</b>	In many cases, it may not be sensitive data. Otherwise, replace it by the week number or by applying a random shift on [-15days, +15days]; this will maintain the seasonal information that may have an implication in the failure occurrence.
<b>Person names, employee numbers</b>	Replace this information by a person’s role within the company, such as: operator, , maintenance manager, etc. When possible, adding information about a person’s expertise level could be worthwhile and will contribute to the credibility of a report.
<b>Location, address</b>	Replace all the addresses and locations by “the factory”. Add the country if there are at least 10 or more companies that own the same type of machine in that country.
<b>Product names</b>	Replace all product names by the keyword “the Product”, except for the names of the machine components that are common to all customers. For example <b>Schneider Lexium Servo Drive</b> is a product name, but it may be common to all the machine owners, making it non-sensitive. This means that this surrogate could behave differently for two different machine types.
<b>Machine configuration(s)</b>	The surrogate of this data type will change depending on the machine type. If the machine configuration is sensitive, then this data will simply be dropped from the content. However, if it is not sensitive but should not be shared as is, then all numeric values must be replaced by adding a random value, for example in the range of [-20%,+20%]. This should give an insight on the range of acceptable values.

identifying, within a text, the entities (words or group of words) that are relative to real-world objects with associated names, and classifying them into pre-defined categories such as person names, organizations, locations, time, quantities, etc. There are many NER tools and APIs, with different capabilities and variable accuracy. As an example, the CLARIN<sup>8</sup> (Common Language Resources and Technology Infrastructure) website references several NER tools, along with their different capabilities and supported languages. Some famous NER python libraries like NLTK<sup>9</sup>, SpaCy<sup>10</sup> or Flair<sup>11</sup> are provided with built-in NER features that support many languages and offer state-of-the-art accuracy and performance. Actually, the well-known NER tools are most commonly used for personal data identification. Regarding the maintenance data anonymization, additional classifiers should be developed and trained for the detection of sensitive business data, following the same approach in Figure 2. Such classifiers can use different machine learning techniques like: Neural Networks, Recurrent Neural Networks (RNN), Conditional Random Fields (CRF), or transfer learning by using transformer models such as Google's BERT [19]. In practice, the above NLP-based techniques are often used together with the rule-based techniques, that are efficient in the detection of entities with well-known and usually static formats or names. These techniques rely on two approaches: pattern matching using regular expressions to detect entities with specific formats (such as dates, phone numbers, credit cards, social/employee numbers), and dictionary lookups to detect entities with well-known names, including countries, cities, streets, organizations, and products.

#### Replacement of sensitive data with appropriate surrogates

The generation of surrogates is one of the most challenging problems in automated data anonymization. Unlike the abundant research about sensitive data detection techniques, little progress has been made in surrogate generation

<sup>8</sup><https://www.clarin.eu/resource-families/tools-named-entity-recognition> (accessed 8 Jan, 2021)

<sup>9</sup><https://www.nltk.org/> (accessed 8 Jan, 2021)

<sup>10</sup><https://spacy.io/> (accessed 8 Jan, 2021)

<sup>11</sup><https://github.com/flairNLP/flair> (accessed 8 Jan, 2021)

[20]. Only some obvious pre-defined substitutions can be found in the literature, such as replacing a person's name by another random name of the same gender, or replacing dates, ages and street numbers by random values. Some works remove all sensitive information and replaces it with their category name (e.g., "Mr. Jack" becomes <PERSON>) instead of using a suitable replacement. However, the risk with such solutions is the loss of vital information and they cannot be generalized and adapted for other use cases, as well as the introduction of confusing lack of detail. Table 1 shows some examples of (generally) appropriate sensitive data surrogates. Ensuring the semantic correctness and usefulness of the anonymized data is a key challenge of surrogate generation [20]. The simplistic solutions, removing the sensitive data or using just the data category as a surrogate, cannot adequately satisfy that requirement. More sophisticated approaches are needed, with a deep dive into each sensitive data type to determine the adequate surrogate for each usage context.

#### Conclusions and perspectives

There are many standards for maintenance data sharing. Despite their limited application scope, their positive impact on the enhancement of the maintenance activity makes the generalization of their approach more attractive. Given this perspective, we developed our solution, 'SemKoRe', to simplify the collection, reuse and sharing of maintenance data through powerful services relying on semantic web technologies. Before adopting the maintenance data sharing approach, our customers had expressed the need for automated anonymization features to avoid sensitive data disclosure. This requirement came up as the make or break point. However, we've found that the application of data anonymization for manufacturing industries is still a challenging task. On one hand, a consensus about the sensitiveness of data seems unreachable for the different industrial actors, since each data fragment could be considered as sensitive for some and non-sensitive for others, which makes a one-solution-fits-all approach impossible. On the other hand, the replacement of sensitive data with appropriate surrogates is still an open challenge. Tackling these challenges is our cur-

## Department Head

rent priority for the SemKoRe project. We are working on a novel solution, combining the efficiency of the ML techniques and the semantic web technologies to reconcile the data protection and context-awareness. Furthermore, collecting diversified maintenance data is another issue that needs to be considered, since a single source of data (e.g. Schneider-Electric) is not representative enough to cover all the needs and specificities of the various industrial domains.

## ■ REFERENCES

1. SINTEF Technology and Society, Norges teknisk-naturvitenskapelige universitet, OREDA. Offshore Reliability Data Handbook, OREDA participants, 2002.
2. ISO 14224: 2016. Petroleum, petrochemical and natural gas industries – Collection and exchange of reliability and maintenance data for equipment. Third edition, Geneva, Switzerland.: International Organization for Standardization (ISO).
3. Portfolio Review 2016, System Performance, Availability and Reliability Trend Analysis (SPARTA), Northumberland, UK, 2016.
4. "WInD-Pool: Wind-energy-Information-Data-Pool," (Online). Available: <http://www.wind-pool.de>. [Accessed 18 11 2020].
5. GE Aviation, "GE Aviation launches Configuration Data Exchange to reduce maintenance costs," [Online]. Available: <https://www.geaviation.com/press-release/systems/ge-aviation-launches-configuration-data-exchange-reduce-maintenance-costs>. [Accessed 18 11 2020].
6. J. R. Bengt Lydell, "OPDE—The international pipe failure data exchange project," Nuclear Engineering and Design, vol. 238, pp. 2115-2123, 2008.
7. Nuclear power plants - Reliability data exchange - General guidelines, International standard ISO 6527, 1982: International Organization for Standardization.
8. K. Dalkir, "Knowledge Management In Theory And Practice," Oxford, Elsevier Inc: Jordan Hill., 2005, p. 132–133.
9. H. Hossayni, I. Khan, M. Aazam, A. Taleghani-Isfahani and N. Crespi, "SemKoRe: Improving Machine Maintenance in Industrial IoT with Semantic Knowledge Graphs," Applied Sciences, vol. 10, no. 6325, 2020.
10. Mamede, N., Baptista, J., & Dias, F. Automated anonymization of text documents. In 2016 IEEE congress on evolutionary computation (CEC), 2016, p. 1287-1294.
11. Marimon, M., Gonzalez-Agirre, A., Intxaurreondo, A., Rodriguez, H., Martin, J. L., Villegas, M., & Krallinger, M. (2019, September). Automatic De-identification of Medical Texts in Spanish: the MEDDOCAN Track, Corpus, Guidelines, Methods and Evaluation of Results. In IberLEF@ SEPLN (pp. 618-638).
12. Personal Data Protection Commission, Singapore, "Guide To Basic Data Anonymisation Techniques", 2018 (Online). Available: [https://www.pdpc.gov.sg/-/media/Files/PDPC/PDF-Files/Other-Guides/Guide-to-Anonymisation\\_v1-\(250118\).pdf?la=en](https://www.pdpc.gov.sg/-/media/Files/PDPC/PDF-Files/Other-Guides/Guide-to-Anonymisation_v1-(250118).pdf?la=en) [Accessed 16 02 2021].
13. P. M. Heider, J. S. Obeid, and M. Meystre, "A Comparative Analysis of Speed and Accuracy for Three Off-the-Shelf De-Identification Tools," AMIA Summits Transl. Sci. Proc., pp. 241–250, 2020.
14. Kuschner, "Anonymizing and Sharing Medical Text Records," Physiol. Behav., vol. 176, no. 3, pp. 139–148, 2017.
15. Goswami, P., & Madan, S. (2017, May). Privacy preserving data publishing and data anonymization approaches: A review. In 2017 International Conference on Computing, Communication and Automation (ICCCA) (pp. 139-142). IEEE.
16. Scaiano, M., Middleton, G., Arbuckle, L., Kolhatkar, V., Peyton, L., Dowling, M., ... & El Emam, K. (2016). A unified framework for evaluating the risk of re-identification of text de-identification tools. Journal of biomedical informatics, 63, 174-183.
17. Domingo-Ferrer, J., Sánchez, D., & Soria-Comas, J. (2016). Database anonymization: privacy models, data utility, and microaggregation-based inter-model connections. Synthesis Lectures on Information Security, Privacy, & Trust, 8(1), 1-136.
18. Nadeau, D., & Sekine, S. (2007). A survey of named entity recognition and classification. Lingvisticae Investigationes, 30(1), 3-26.
19. Mao, J., & Liu, W. (2019). Hadoken: a BERT-CRF Model for Medical Document Anonymization. In IberLEF@ SEPLN (pp. 720-726).
20. Yogarajan, V., Pfahringer, B., & Mayo, M. (2020). A review of automatic end-to-end de-identification: Is high accuracy the only metric?. Applied Artificial Intelligence, 34(3), 251-269.

**HICHAM HOSSAYNI** R&D Engineer on Semantic Web & Industrial IoT at Schneider Electric, Grenoble, France. Previously, he worked in Orange Applications for Business (OAB), Grenoble as Embedded Software Engineer, and in CEA-Leti, Grenoble as a Cryptography and Embedded software R&D Engineer. He received M.S. Eng. in networks and distributed systems from ENSEIRB-Bordeaux (France) and ENSIAS-

Rabats (Morocco), and M.S. in Cryptography, Security and Information Coding from Joseph-Fourier, Grenoble. His current research interests are Internet-of-Things, Semantic Web, Data Mining and Cloud & Edge computing. ([hicham.hossayni@se.com](mailto:hicham.hossayni@se.com))

**IMRAN KHAN**, Senior Member IEEE, is currently working as Senior Principal Architect at Schneider Electric France, leading Ontology Program for the IoT Platform of Schneider Electric. Previously, he worked as Innovation Project Leader working on efficient data and information management in Industrial IoT space. He received Ph.D. degree from Institut Mines-Télécom, Télécom SudParis jointly with UPMC Paris VI, France, M.S. degree from M.A. Jinnah University, Pakistan and B.S. degree from COMSATS Institute of IT, Pakistan. During his Ph.D., he worked as collaborating researcher at Concordia University, Montreal, Canada to lead a 3 year Cisco funded project. He was also involved in several European research projects funded by ITEA2 and H2020. During M.S., he was member of Center of Research in Networks and Telecom (CoReNeT) and worked on projects funded by French Ministry of Foreign Affairs and the Internet Society (ISOC). He has number of publications in peer reviewed conferences & journals and patents. He has also contributed to IETF standardization activities. His current research interests are IoT, Knowledge Graphs, Data & Information Management, Cloud & Edge Computing and Intelligent Systems. For additional details: <http://www.imrankhan1984.com>, ([imran@ieee.org](mailto:imran@ieee.org)).

**NOEL CRESPI** Prof. Noel Crespi holds Masters degrees from the Universities of Orsay (Paris 11) and Kent (UK), a diplôme d'ingénieur from Telecom Paris-Tech, and a Ph.D and an Habilitation from UPMC (Paris-Sorbonne University). From 1993 he worked at CLIP, Bouygues Telecom and then at Orange Labs in 1995. He took leading roles in the creation of new services with the successful conception and launch of Orange prepaid service, and in standardization (from rapporteurship of the IN standard to the coordination of all mobile standards activities for Orange). In 1999, he joined Nortel Networks as telephony program manager, architecting core network products for the EMEA region. He joined Institut Mines-Telecom SudParis in 2002 and is currently Professor and Program Director at Institut Polytechnique de Paris, leading the Service Architecture Lab. He coordinates the standardization activities for Institut Mines-Telecom at ITU-T and ETSI. He is also an adjunct professor at KAIST (South Korea), an affiliate professor at Concordia University (Canada), and

a guest researcher at the University of Goettingen (Germany). He is the scientific director of ILLUMINE, a French-Korean laboratory. His current research interests are in Data Analytics, the Internet of Things and Softwarization. <http://noelcrespi.wp.tem-tsp.eu/>, ([noel.crespi@telecom-sudparis.eu](mailto:noel.crespi@telecom-sudparis.eu)).

Annex **C**

## Paper III



# Privacy-preserving Sharing of Industrial Maintenance Reports in Industry 4.0

Hicham Hossayni  
*Institut Polytechnique de Paris,*  
*Schneider Electric,*  
38000 Grenoble, France  
hicham.hossayni@se.com

Imran Khan  
*Schneider Electric,*  
38000 Grenoble, France  
imran2.khan@se.com

Noel Crespi  
*Institut Polytechnique de Paris, IMT*  
91011 Evry Cedex, France  
noel.crespi@it-sudparis.eu

**Abstract**—Knowledge sharing has proven its worth in many domains as an enabler for optimized processes and improved organizational agility. It makes problem-solving and decision-making much faster for the stakeholders. Knowledge sharing has also proven its effectiveness in the industrial maintenance domain as a key vector for the improvement of maintenance processes, that are still dominated by traditional practices. However, sensitive data disclosure remains one of the major roadblocks that prevent this trend from gaining ground in the manufacturing industry. In this paper, we present a new approach to provide remedies for sensitive data disclosure problems during the knowledge sharing activity. Relying on the semantic web, rule-based, and natural language processing techniques, our approach helps to detect and identify the potentially sensitive data in maintenance reports before being shared with other actors.

**Index Terms**—Industrial maintenance, Knowledge sharing, Natural Language Processing, Semantic web, Sensitive data disclosure.

## I. INTRODUCTION

Knowledge sharing has become commonplace in various domains. It is adopted by several organizations to increase operational efficiency and staff productivity [1]. It makes the expertise of specialists accessible to a large community which helps for individual growth and development. Despite the existence of many standards and international consortiums that promote knowledge sharing for industrial maintenance, this trend has not gained traction in the manufacturing industry yet, where traditional practices are still in force, such as paper-based maintenance reports or the maintenance reports that are never reused. Also, current industry policies and practices make the maintenance knowledge inaccessible with little nay no motivation to share it with others.

In our previous work, we proposed SemKoRe [2], a vendor-agnostic solution that enables maintenance knowledge sharing with Semantic Knowledge Graphs. SemKoRe gathers all failure’s related data in the knowledge graph and shares it among all connected customers in order to easily solve future failures of the same type. It received the approval of several important clients of Schneider-Electric located in several countries and from various segments.

During our work on the SemKore Project, we conducted an interview campaign with several experts in the field of

industrial maintenance. The results were unequivocal; sharing maintenance knowledge between manufacturers would have a positive impact on the optimization of maintenance routines and on manufacturing productivity. However, a major problem that would prevent such solutions from becoming a reality, is the presence of sensitive information in maintenance reports. It is therefore essential to guarantee data privacy-preserving and avoiding sensitive data disclosure before sharing any maintenance report.

In this paper, we propose a new approach to avoid sensitive data disclosure during maintenance knowledge sharing through the SemKoRe solution. Our approach relies on Semantic Web ontologies combined with different techniques, usually used for data anonymization [3], such as: Named Entity Recognition or rule-based sensitive data detection.

We tackled a couple of challenges in this work. The first challenge is the lack of datasets with a reasonable number of real maintenance reports containing confidential data. Fortunately, the interviews campaign allowed us to understand the nature of sensitive data in maintenance reports and some ways to recognize it. The second challenge is that sensitive data detection techniques, especially the NLP-based ones, require annotated data corpus with a considerable number of samples, while we have almost no maintenance report with sensitive data. So, for the proof of concept needs, we implemented a generic solution to collaboratively collect, annotate, and construct our own data corpus.

The collected data corpus was used to train and evaluate three different Named Entity Recognition (NER) models. Then, we deployed and used on-premise all the trained models on an Edge gateway. The results are promising, they show that our approach can be used for on-premise detection of potentially sensitive data in maintenance reports. We also identified several areas of improvements, as future work, to make our solution usable in real use cases.

This paper is organized as follows: the related work is presented in section II, followed by the requirements imposed by our customers and the SemKoRe project. Section IV presents the main contributions of the paper. Section V details the

challenges and the future works before concluding in Section VI.

## II. RELATED WORK

There are many international standards and consortium agreements that promote and adopt the maintenance knowledge sharing approach. Some of the well-known ones are:

- The OREDA project [4]: Offshore and Onshore Reliability Data for oil and gas industries, which led to the ISO 14224 international standard [5].
- ISO 6527 [6]: A Reliability Data Sharing standard for nuclear energy producers.
- SPARTA [7]: System Performance, Availability and Reliability Trend Analysis for wind energy industries in UK.
- WInDPool [8]: stands for Windenergy-Information Data-Pool, for wind energy industries in Germany.
- OPDE [9]: The International Pipe Failure Data Exchange.
- The Configuration Data Exchange [10]: launched by GE Aviation for the worldwide aviation industry.

All these standards and projects recommend applying data anonymization and removing sensitive data before sharing maintenance reports with others. They, however, don't propose any solution for that purpose. And to the best of our knowledge, no existing work on detecting sensitive information in industrial maintenance reports has been published. Several reasons justify this situation, the first one is that most industrial users have no intention to share their maintenance data with others as they feel that they have experienced persons who can take care of the maintenance activities without external help. However, the situation is rapidly changing due to the decline in the working force in industrial countries like Germany and Japan [11]. The second reason is that majority of the industries are closed source in nature. They never had the concept of collaboration or inclination to learn from the experiences of other teams. Another reason is that no maintenance datasets can be found on the internet or open data repositories to help researchers and scientists explore this field.

On the other side, sensitive data detection in textual or unstructured data has been studied in several works in the literature. Most of them target the medical field for which data anonymization is imposed by regulatory rules such as HIPAA (Health Insurance Portability and Accountability Act) [12]. Various techniques are used and range from simple rule-based techniques to more advanced natural language processing (NLP) techniques [13].

Named Entity Recognition (NER) remains one of the most used NLP techniques for sensitive data detection. It consists of identifying, within a text, the entities (words or group of words) that are relative to real-world objects with associated names. There are also many hybrid solutions relying on both, rule-based and NER techniques [14] that usually offer good

performances with high accuracy scores, exceeding sometimes 98% [15] which is judged to be sufficient for data privacy purpose [16].

## III. REQUIREMENTS

As discussed previously, this project is considered as a feature that will be proposed to our SemKoRe's customers. Thus, a set of requirements needs to be fulfilled in order to satisfy our customer's needs and to make the solution easily integrable within SemKoRe's ecosystem.

The *first* requirement is that the solution must not be restricted to specific types of machines or specific manufacturing domains. Rather, it should support almost any type of machine used in any manufacturing field. The *second* requirement is that the solution should be customizable to cover the different needs of our customers. In fact, our interviews campaigns showed that each manufacturer has its own data sensitivity evaluation criteria, and the needs of a customer are sometimes, different from others even for the same standard machines.

The *third* requirement is that the solution needs to be deployable and executable on an edge gateway. In fact, most of SemKoRe's services are running on an edge gateway directly connected to the machine. The goal is to offer on-premise services to our customers so that they can be used without needing to be permanently connected to the cloud. (please refer to our previous paper [2] for more details).

## IV. CONTRIBUTION

### A. Survey on maintenance reports and sensitive attributes

To understand the nature of sensitive data within a maintenance report, we reached out to several domain experts from various industries who are involved in/related to the maintenance activities in their organizations. Several questions were asked during our interviews campaign. The goals were mainly to understand the structure of maintenance reports and to know which data or attributes can be judged as sensitive. Hereafter, we present the list of questions and summarized answers we received during the interviews. For more details on these interviews like statistical validity and global conclusions please refer to our previous work [17].

#### 1) Q1- What is the structure of a maintenance report?:

Almost all maintenance reports are composed of two parts: the *first part* is a structured section (usually heading) that recalls the context of the maintenance operation such as the maintenance date and time, location, maintenance operator's name or identifier, machine Identifier. The *second part* is the content of the report, it is an unstructured text written by the maintenance operator to describe the reasons for the maintenance operation and the actions applied during the maintenance or repair of the machine.

#### 2) Q2- Which data can be considered sensitive in a maintenance report?:

We can distinguish mainly three types of sensitive data in a maintenance report:

- Personal data: all information relative to a specific person within the company. E.g.: Personally Identifiable Information (PII) like employee’s name or number, role, addresses, phone number.
- Business data: every information about the company or its activity. This includes the manufacturer’s name, products, location, customers, or subcontracting companies.
- Manufacturing data: especially information about the manufacturing process that may leak details about the manufactured products or the trade secrets, e.g. secret recipe of Coca-Cola. Also, some industries believe that the machine configuration is sensitive since it is part of the competitive know-how making it part of the company’s industrial property.

3) *Q3- How sensitive data appears in a maintenance report’s content?:* Personal information can only be found in the report’s heading, it is very rare or almost impossible to find personal data in a maintenance report’s content.

Regarding the business data, the critical data (like financial data or strategy) are never shared with the maintenance technicians, therefore it has no chance to appear in a maintenance report. However, the names of the manufacturer, products, customers, or other companies might likely appear in the report to describe the maintained machine’s context.

Finally, manufacturing data can consist of multiple details but the two most important or common ones are:

- The machine configuration: corresponds to the settings of the different machine components, e.g. motor rotation speed, pressure, temperature.
- The manufacturing process: corresponds to a succession of granular steps and actions needed to produce the finished goods.

4) *Q4- How to recognize sensitive data in maintenance reports?:* The heading of a maintenance report is usually considered sensitive, since it contains information about the company, the location, the maintenance operator, in addition to different identifiers and references with internal significance only. Discarding the maintenance report’s heading from the shared information is a wise decision.

Also, the business data differs from one company to another. It is mostly a set of proper nouns and well-known entities such as the names of products, customers, companies, etc...

On the other hand, there is a common pattern of the machine configuration and manufacturing process in the majority of maintenance reports. Actually, for every machine failure, the maintenance technicians usually describe how a machine component has been used or has behaved, or what configuration it had when the failure occurred. So, after the analysis of some maintenance reports, we found that such information is most of the time relative to a machine component since they are the most representative landmarks in a machine to describe a failure or a maintenance operation. This hypothesis was

approved by the domain experts to be the most representative of the manufacturing data commonly found in the maintenance reports. Nevertheless, they do not exclude other rare forms not covered in this paper and that will be part of future works.

Finally, it is important to note that for standard machines, most of the machine configurations or manufacturing processes are provided by the OEM or the machine builder, and therefore they are not sensitive nor confidential. The exceptions are more relative to the customization applied on the standard machines such as adding new components or using a secret configuration that requires high engineering expertise. In both cases, only a domain expert can evaluate the sensitivity of each manufacturing data.

## B. Our approach

1) *Overview:* To detect sensitive data in maintenance reports, we adopted a hybrid approach relying on rule-based techniques and Named Entity Recognition. In addition to these techniques, we used Semantic Web ontologies to guarantee the customizability, portability, and useability by different customers for different machine types.

For the detection of business data such as the manufacturer name, products names, customers, and subcontracting companies; we use a dictionary lookup technique. We ask the customer to provide a list of all proper names (e.g. products names, people, companies) typically used in the company. This list can be updated whenever new entities are introduced in the company.

In this paper, we focus on the manufacturing data and within it to the identification of the machine components in the text. Since, as presented above, they are the most common form of machine configuration or manufacturing process leaks. For that purpose, we will use NER techniques.

Finally, as discussed previously, recognizing the sensitive nature of a piece of data can only be done (for the moment) by a human expert. Therefore, to provide a highly customizable solution, we propose an ontology-based approach to allow the domain experts to specify the machine components that they judge as being potentially sensitive.

Figure 1 shows a detailed overview of our approach. It requires four different inputs:

- 1) A machine components taxonomy bringing together the components’ names, their synonyms, and abbreviations.
- 2) An annotated text corpus that will be used for the training of the NER module. This corpus should be composed of numerous text entries with tagged machine components. As a requirement, all tagged machine components must be part of the machine taxonomy.
- 3) A specific machine ontology that describes a physical machine and all its components. This ontology must be restricted to the components defined in the machine taxonomy. In this way, we guarantee the synchronization

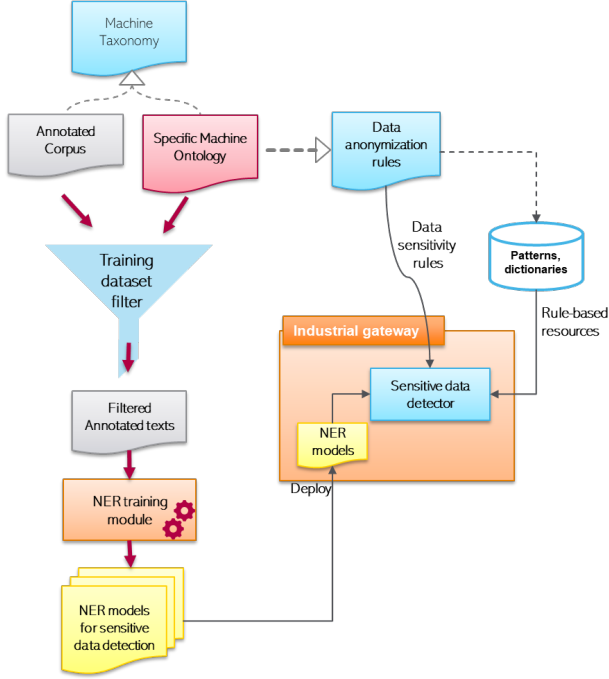


Fig. 1. Detailed overview of sensitive data detection in maintenance reports

between the components identified by the NER model and the components referenced in the machine ontology.

- 4) A data anonymization rules ontology, which specifies the different ways used to identify the various types of sensitive data. It is imported in the machine ontology to allow domain experts to specify the sensitivity rules or flags for each machine component, i.e. flag a component as potentially sensitive or not sensitive. This ontology also references the rule-based resources (e.g. dictionaries) that are used during the sensitive data search phase.

Note that the outcome of this approach is a custom tool for the detection of (potentially) sensitive data in the maintenance reports of a specific target machine. However, our approach is applicable to a variety of machines from different domains.

Once all the inputs are provided, the *training dataset filter* uses the *specific machine ontology* to filter the *annotated corpus*. It also ignores all tagged machine components that are not part of the target machine. We get a *filtered annotated corpus*, as a result, that only contains samples tagged with components of the target machine. The reason for this choice is that we consider the annotated corpus of general-purpose and that might reference several thousands of machine components, while only a few dozens of components are part of the target machine.

The *filtered annotated corpus* is then used to train a *NER model* for the machine components detection. It is known that reducing the training datasets helps in accelerating the training phase. Also, this allows the NER models to focus on efficiently recognizing a limited number of entities instead of trying to recognize all existing machine components. This step provides

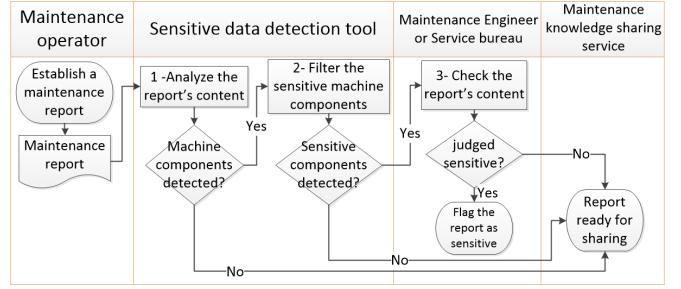


Fig. 2. Sensitive data detection flow for maintenance data sharing

a *custom NER model* trained to detect only the components of the machine for which the maintenance reports are drawn up.

To satisfy the last requirement, we deploy the trained NER model on the industrial gateway connected to the target machine. Then, the *Sensitive data detector* service will use on-premise: 1) the trained NER model, 2) the data sensitivity rules, and 3) the rule-based resources during the analysis of the maintenance operators' inputs for the recognition of (potentially) sensitive data.

2) *Sensitive data detection flow in maintenance reports*: The Figure 2 describes the flow of sensitive data detection in maintenance reports, it consists of three main steps:

- 1) The first step consists of automatically analyzing the maintenance report's text and identifying all machine components in it.
- 2) Once we have a list of machine components detected by the tool, it is filtered in order to keep only the items judged potentially sensitive. When sensitive components are found in the text, the maintenance report is flagged as containing potentially sensitive data and must be verified by a domain expert such as a maintenance engineer or service bureau.
- 3) Finally, the domain expert decides if the report can be judged sensitive or not. All reports identified as non-sensitive are tagged as ready for sharing and are transferred to the maintenance knowledge sharing service.

In the following, we will describe the design of all these components to provide a customizable sensitive data detection tool for industrial maintenance reports.

### C. Data corpus collection & preparation: Industrial Machine Data Pool

The major challenge that we had to face in this project was to find the needed datasets with industrial maintenance vocabulary. Traditional sources such as internal documents or open datasets were not helpful and were not satisfying our needs. Therefore, we decided to collect and construct the required data corpus ourselves. Two complementary steps are needed to build our datasets:

- 1) Collect structured data and texts about machine components from internet web pages. The structured data will be used to construct our machine components' taxonomy, and the texts will be used to build the data corpus.

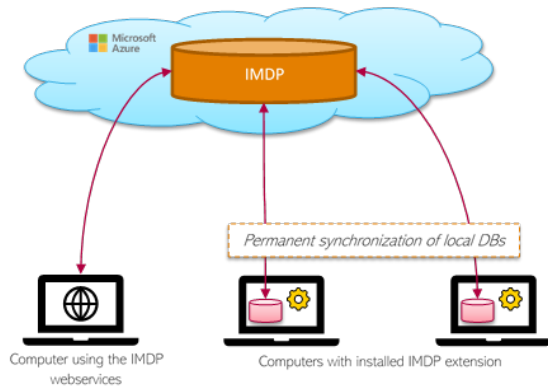


Fig. 3. IMDP's architecture

- 2) Annotate the collected texts using the machine components taxonomy. This step can be divided into two sub-tasks:
  - a) *Automatic annotation* that consists of using tools to annotate automatically the collected texts and to optimize the annotation effort.
  - b) *Manual annotation*: In this task, the user must annotate and correct the entities not detected or incorrectly annotated by the automatic annotation tools.

Thus, we implemented the Industrial Machine Data Pool that provides a collaborative solution with the necessary services for data collection and automatic and manual annotations.

1) *IMDP - Industrial Machine Data Pool*: Several tools are proposed on the internet for web scraping and corpus construction, e.g.: Data Scraper from Data Miner<sup>1</sup>, OpenLink Structured Data Sniffer<sup>2</sup>, or Web Scraper from OpenLink Software<sup>3</sup>. These tools are more designed to automate the collection of structured data, however, the manual effort needed to collect unstructured text data makes them useless. Thus, we decided to develop our own tool to accelerate the data collection task. We designed an extension that can be integrated natively into internet browsers. This allows the user to interact with the data collection tool directly from the visited web pages. In fact, a simple right-click generates a popup form that the user can fill with adequate content, (even with copy and paste) and save it as a new entry in the IMDP. In addition, several users can simultaneously use the extension and participate in the data collection effort. Each instance of the browser extension manages a local database that is permanently synchronized with a central cloud database that is common to all users (see Figure 3). In order to avoid duplicate entries, all the links of the pages that are already collected are highlighted on every visited web page. IMDP server offers also a web service that can be used similarly to the internet extension, but without integration in the visited web pages.

a) *Data sources & data structure*: We configured IMDP to collect data about machine components, and we choose

<sup>1</sup><https://dataminer.io/>

<sup>2</sup><https://osds.openlinksw.com/>

<sup>3</sup><https://webscraper.io/>

Wikipedia as a starting point because of the availability of a large number of pages describing the industrial machines and components. Also, the unified structure of Wikipedia's articles makes it possible to automate the data extraction. From each web page, we gather the following details: The component's type (structural, mechanical, or control element), name, synonyms and abbreviations, URLs, and relative texts.

Afterward, the collected texts must be cleaned and annotated. We split all collected texts into paragraphs with less than 800 characters. The goal is to simplify the text annotation and to attract more colleagues for a collaborative text annotation effort.

b) *Automatic annotation*: We implemented an automatic annotation tool that identifies and automatically tags machine components in the text to accelerate the annotation task. We used three different sources for our automatic annotation:

- Exact keywords and plural forms lookup from the collected machine taxonomy.
- Wikipedia annotations
- Tagme annotations: Tagme<sup>4</sup> is a service that performs on-the-fly semantic annotation of short text via Wikipedia as a knowledge base.

This strategy allows detecting many machine components in the texts, with, however, a non-negligible rate of false-positive annotations. Which requires the involvement of experts for manual annotation.

c) *Manual annotation*: During the manual annotation task, the user needs to check the correctness of automatic annotations and annotate the missed entities. When a new entity is identified by the user, it is added to the machine taxonomy either as a synonym of an existing entity or as a new entity for which the user is asked to provide the different details required by the IMDP. This allows to keep track of all entities in the texts and to know to which entity every annotation refers.

d) *SUMMARY*: At the time of writing these lines, we built a dataset composed of: 193 taxonomy entries with 283 synonyms and abbreviations, 188 articles texts, 3523 cleaned paragraphs, and 333 fully annotated paragraphs containing 1591 sentences.

#### D. Machine ontology definition

As discussed previously, our approach requires the definition of an ontology that describes the physical machine. This ontology should be defined by a domain expert (e.g. machine designer), and an ontology expert (see Fig. 4). For this purpose, we defined a T-Box for the Machine Ontology Model (see Fig. 5) that must be used to create an instance of the physical machine. The T-Box ontology describes the components of the machine and defines the characteristics of each component such as its attributes or its configuration (figure 6). Every machine is described as a hierarchical assembly of the different physical elements (Unit, Sub-unit, Part, and

<sup>4</sup><https://tagme.d4science.org/>

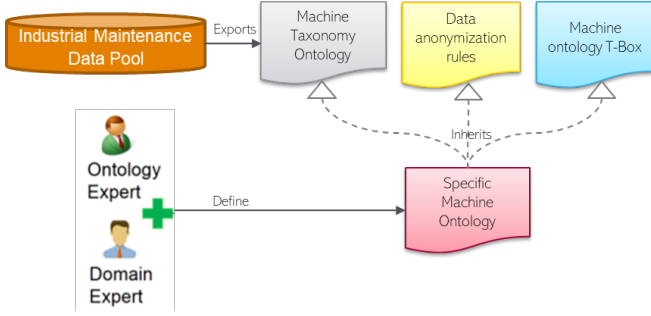


Fig. 4. Machine ontology design

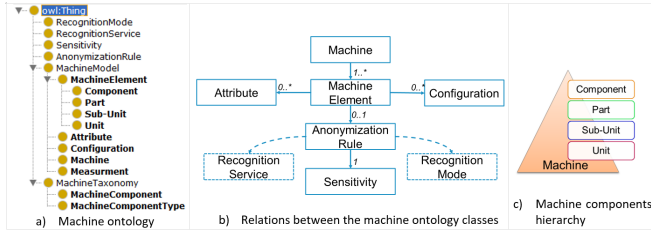


Fig. 5. T-Box & Machine ontology specifications

Component) as shown in the Pyramid in Fig.5.c. One of the Machine T-Box requirements is that all machine components must be referenced in the Machine taxonomy ontology that was generated previously by IMDP.

We also created a Data Anonymization Rules ontology that defines, at this stage, the sensitivity flags relative to each machine component. In other words, it classifies a machine component as “sensitive” or “not sensitive”. The sensitivity flag can also be defined for the other machine elements. In that case, the sensitivity flag is inherited by all the machine components belonging to the flagged element unless a different sensitivity has been defined. This ontology will be extended in the future to cover more data anonymization aspects such as managing multiple sensitive data detection services (e.g. dashed concepts in Fig.5.b) or supporting the sensitive data masking or replacement rules.

1) *Training Named Entity Recognition models for machine components detection:* To train the Name Entity Recognition models, we predefined two categories for entities’ classification: 1) Machine Component and 2) Machine Equipment. These categories represent more than 95% of the tagged entities in the collected texts. So, other categories (Maintenance process, Manufacturing process, Material, and Machine failure) are not included in our study. We trained three different NER models: 1) A custom Spacy NER model, 2) a NER model with CRF (Conditional Random Fields), and 3) a BERT-based NER model.

To train all these models, we first export the annotated texts from IMDP in the needed formats (e.g. BIO, BILUO), then we use the exported data to feed the NER training modules. Each training module applies the pre-processing steps required for the training phase. Finally, a NER model is generated and is ready for use for machine components detection.

a) *Custom Spacy NER model:* Spacy [18] is a very known open-source framework for NLP. It offers several features including an efficient deep CNN-based NER system achieving state-of-the-art performances. Spacy comes with a set of pre-trained NER models for different languages to detect the most common entities: Person, Location, Organization, Values. It is also possible to retrain the pre-trained models to include additional categories or even to train a new model based on an empty NER model. In our case, we chose to train an empty NER model to detect the desired categories only: Machine equipment and Machine component.

b) *CRF NER model:* Conditional random fields are a class of statistical methods designed for the analysis of sequential data (such as text, images, DNA) [19]. They’re often used in pattern recognition and machine learning. One reason for their good performances in the NER task is that they consider the input’s context by taking into account the neighboring or surrounding samples. Therefore, to train a CRF-based NER model, we need to pre-process the annotated texts in order to apply tokenization and extract different features for each token such as POS tags, lemma, shape, and other flag features like: *is uppercase, is capital, is a stop word, is a hyphen*. The features extraction is done within a window of 11 tokens, it includes the features of the current token, and 5 next and 5 previous tokens since some machine components are composed of many tokens such as: “*glass-ceramic-to-metal seal*” that is composed of 8 tokens (5 words and 3 hyphens).

c) *NER model with BERT:* BERT (for Bidirectional Encoder Representations for Transformers) is a deep learning model that has given state-of-the-art results on a wide variety of natural language processing tasks. It has been pre-trained on Wikipedia and BooksCorpus [20]. and requires task-specific fine-tuning [21]. In our case, we applied BERT to build our NER model for machine components detection. To train our NER model, we followed the same approach and implementation proposed in [22] as it matches our problem space. We used a 12-layer BERT model with an uncased vocabulary, and we set the target learning rate to  $2 \cdot 10^{-5}$  with a batch size of 16 and 7 training epochs.

2) *Implementation details:* We used multiple technologies to implement our proof of concept. For IMDP we used:

- Google Chrome as a target for the browser extension as it is quite a popular browser worldwide.
- HTML+CSS and Javascript to implement the web UI, and the different chrome extension services, such as data collection, cleaning, annotation, and data export.
- PouchDB, as a storage database on both, local storage and on the cloud storage. PouchDB is a no-SQL database with native data synchronization features between the local and cloud databases.

The creation of the different T-Box ontologies (Machine Taxonomy, Data Anonymization Rules, Machine Model) was mainly done with the Protégé tool, which is widely used for the creation of semantic web ontologies.

We used the Python programming language to implement the “Training dataset filter” service, and for the NER training module. Specific python libraries were used for the implementation and training of the different NER models:

- Spacy library for the custom spacy NER model.
- pyCRFsuite for the NER model with CRF.
- PyTorch library for NER model with BERT.

For the training of the different NER models, we used a capable laptop setup running, Ms. Windows 10, with 32Gb of RAM and an octa-core i7 processor and a CUDA-enabled GPU (NVIDIA QUADRO M2020) of 12Gb of memory. As an industrial gateway, we used RaspBerry Pi 4, with 4Gb of RAM, ARM-Cortex-A72, and running Linux Ubuntu 5.4. Docker was used as a containerization technology for the different Edge services such as the *Sensitive data detector* service.

Finally, we used Microsoft Azure to host the IMDP server containing the master PouchDB. Currently, the training dataset filter and NER training module services were implemented and tested on a laptop, but we plan to deploy them on a dedicated Ms. Azure server. On the RaspBerry Pi, we deployed the trained NER models and the *sensitive data detector* service.

3) *Evaluation*: To evaluate our approach, we defined a sample machine ontology containing all the components of our machine components taxonomy, and we used the complete annotated dataset, generated by IMDP, to train the NER models. We also defined a dictionary of some names, found in the annotated texts for the rule-based part. We can deduce that the accuracy of our approach is equal to the accuracy of the NER models since the rule-based resources are defined to achieve (biasedly in our PoC) an accuracy of 100%. However, in order to fully evaluate this approach, we need to have much more data than we currently use.

To evaluate the NER models, the current dataset contains 1591 annotated sentences. 70% of the dataset (i.e. 1114 sentences) was used for the training, and 30% (477) was used for the test. We adopted also the F1-score as an evaluation metric. It is defined as follows:

$$F1 = 2 * \frac{precision * recall}{precision + recall}$$

Where the precision is the percentage of named entities found by the learning model that is correct, and the recall is the percentage of named entities present in the corpus that are found by the model.

Table 1 shows the recall, precision, and F1 score of the different models. We can see that even with a relatively small dataset we can achieve good scores exceeding 83%. CRF and BERT-based models achieve a similar F1 score of around 0.84. While the best results were achieved by Spacy’s NER model getting close to 0.9 of F1 score.

TABLE I  
PERFORMANCE RESULTS FOR THE DIFFERENT NER MODELS

Model	Recall	Precision	F1 score
SPACY NER	0.884	0.908	0.896
NER with CRF	0.821	0.857	0.837
NER with BERT	0.858	0.818	0.838

TABLE II  
EXECUTION TIME OF THE NER MODELS ON LAPTOP AND RASPBERRY PI  
(MS/SAMPLE OF 1000 WORDS)

Model	Time on Laptop	Time on Raspberry Pi
SPACY NER	41	183
NER with CRF	8	14
NER with BERT	0.17	9239

We also evaluated the execution times of the different NER models, since our main execution target is an embedded industrial gateway. We found that the time needed to analyze a text and extract the sensitive entities is perfectly linear with the number of words in the text. Table 2 shows the time needed to analyze a sample of 1000 words. BERT showed exceptional performances with the ability to analyze a sample in less than 0.2ms on the laptop. However, it takes more than 9 seconds for the same task on RaspBerry Pi which makes it not suitable for our case. CRF’s model was also fast with 8ms on the laptop and 14ms on the RPI, while Spacy’s NER model takes more execution time on both, the RaspBerry Pi with 183ms and the laptop with 41ms, which also remains reasonable for an on-the-fly text analysis feature. Finally, once a machine component is detected, looking in the ontology if it is sensitive or not takes a negligible time ( $< 10^{-4}ms$ ).

## V. CHALLENGES AND FUTURE WORK

In this paper, we propose a new approach for sensitive data detection in industrial maintenance reports. With the current implementation, we conducted some tests on a limited set of maintenance reports, for which we created a simple machine ontology and found promising results. However, we found several areas of improvement for the future.

The first area of improvement is that we used a restrictive hypothesis as a basis of our approach. In fact, not all sensitive data are relative to machine components. As an example, the maintenance operator could describe a manufacturing step instead of the machine component, e.g. “The packets sealing has small holes”. Also, the machine components might be sometimes described by their use such as using “the milk container” instead of the “liquid tank or reservoir”. This can be improved by adopting digital tools to prepare maintenance reports like a UI based on a set of ontologies for taking input from the operators by suggesting standard terms.

The second challenge is the nature of the language used for real maintenance reports, and that makes the automatic analysis awkward. In fact, maintenance operators do not provide literature texts, they usually use short informal texts, or even use street slang or urban vocabulary with frequent typos and

missing punctuation. Similarly, the operator’s vocabulary understanding may not be coherent with the normative definition, which often results in the use of non-standard abbreviations. Here again, the use of digital tools can be helpful to facilitate the operator to provide details as per normative definitions. Also, using real maintenance reports to train the NLP models could help in capturing the specificities of the maintenance reports’ language.

The third challenge is the use of multiple local languages to create maintenance reports. We focused so far on English maintenance reports. This can be improved by proposing a multi-lingual sensitive data tool. Such a tool will require NER models trained with various maintenance data corpus and taxonomies from each of the supported languages.

Another future work is the simplification of the data annotation process by decoupling the data annotation and the machine taxonomy. Some works like [23] showed that it is possible, for some NLP tasks, to train models on a corpus annotated with a taxonomy different from the one it is designed to output annotations for.

## VI. CONCLUSION

Several standards and international consortiums promote the maintenance data-sharing approach and have proven its efficiency even on limited application scope. However, sensitive data disclosure remains one of the major roadblocks of knowledge sharing in several domains and industrial maintenance is not an exception. For this reason, we proposed a new approach to avoid sensitive data disclosure during the maintenance data sharing activity. We conducted interviews with several domain experts to understand the needs and expectations from such a feature. As a result, the needs differ from one user to another even for the same machines types, and, judging -with certitude- data to be sensitive or not, remains the role of a human expert such as a manufacturing engineer. In our approach, we aim to simplify the work of the human expert by identifying the potentially sensitive data in maintenance reports’ content. This approach relies on Semantic Web technologies to allow the user to customize his solution and specify the items that can be considered as potentially sensitive. In this paper, we evaluated the use of three well-known NER models: Spacy’s pre-trained NER model, CRF-based, and BERT-based NER models. The evaluation results show that even with a small dataset (with less than 1600 samples) these models have F1 scores between 0.84 and 0.90. We were able to deploy them on a Raspberry Pi 4 and we found that Spacy’s NER model and CRF are more suitable for on-premise execution on an edge gateway. Several future work items have also been identified to continue working on this topic.

## REFERENCES

[1] A.-U.-H. Muhammad, S. Anwar. "A systematic review of knowledge management and knowledge sharing: Trends, issues, and challenges," *Cogent Business & Management*, vol. 3, no. 1, p. 1127744, 2016.

[2] H. Hossayni, I. Khan, M. Aazam, A. Taleghani-Isfahani, N. Crespi, "SemKoRe: Improving Machine Maintenance in Industrial IoT with Semantic Knowledge Graphs," *Applied Sciences*, vol. 10, 2020.

[3] M. Nuno, B. Jorge, D. Francisco, Automated anonymization of text documents, *IEEE Congress on Evolutionary Computation*, 2016.

[4] SINTEF Technology and Society, Norges teknisk-naturvitenskapelige universitet, OREDA. *Offshore Reliability Data Handbook*, 2002.

[5] ISO 14224:2016. Petroleum, petrochemical and natural gas industries – Collection and exchange of reliability and maintenance data for equipment. International Organization for Standardization (ISO).

[6] ISO 6527, 1982: Nuclear power plants - Reliability data exchange - General guidelines, International Organization for Standardization.

[7] Portfolio Review 2016, System Performance, Availability and Reliability Trend Analysis (SPARTA), Northumberland, UK, 2016.

[8] *WinD-Pool: Wind-energy-Information-Data-Pool*, [Online].

[9] J. R. Bengt Lydell, "OPDE—The international pipe failure data exchange project," *Nuclear Engineering and Design*, vol. 238, 2008.

[10] GE Aviation, "GE Aviation launches Configuration Data Exchange to reduce maintenance costs,". [Online]. [Accessed 18 11 2020].

[11] Katharina Buchholz, "The Threat of Declining Working Age Populations," [online][Accessed 03 10 2021].

[12] Department of Health and Human Services, "The health insurance portability and accountability act of 1996," T. Report 65 FR, 2000.

[13] V. Veronika, R. Farkas. "De-identification in natural language processing," 37th International Convention on Information and Communication Technology, Electronics and Microelectronics. IEEE, 2014.

[14] Y. Vithya, P. Bernhard, M. Michael, "A review of automatic end-to-end de-identification: Is high accuracy the only metric?," *Applied Artificial Intelligence*, vol. 34, no. 3, pp. 251-269, 2020.

[15] M. Montserrat, G.-A. Aitor, I. Ander, "Automatic De-identification of Medical Texts in Spanish: the MEDDOCAN Track, Corpus, Guidelines, Methods and Evaluation of Results," *IberLEF@ SEPLN*, 2019.

[16] S. Personal Data Protection Commission, "Guide To Basic Data Anonymisation Techniques," 2018. [Online]. [Accessed 29 09 2021].

[17] H. Hossayni, I. Khan, N. Crespi, "Data Anonymization for Maintenance Knowledge Sharing," *IEEE IT-Professional*, vol 23, no 5, 2021.

[18] M. Honnibal, M. Ines . "spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing." *To appear 7.1* (2017): 411-420.

[19] Y. N. LEE, W. Sun, H. L. CHIEU, et al. Conditional random fields with high-order features for sequence labeling. *Advances in Neural Information Processing Systems 22 (NIPS 2009)*, 2009, p. 2196-2204.

[20] Z. Yukun, K. Ryan, Z. Rich. "Aligning books and movies: Towards story-like visual explanations by watching movies and reading books," *IEEE international conference on computer vision*, pp. 19-27, 2015.

[21] D. Jacob, C. Ming-Wei, L. Kenton: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[22] Face, Hugging, *BERT PyTorch GitHub*, 2019. [Online].

[23] S. Soufian, H. Nicolas, M. Emmanuel, "Dialogue act taxonomy interoperability using a meta-model," *International Conference on Computational Linguistics and Intelligent Text Processing*, 2017.





Annex **D**

# Interviews Campaign Questions

# Customer Interview questionnaire

<b>Name</b>	.....
<b>Role</b>	.....
<b>Company</b>	.....
<b>Location</b>	.....
<b>Experience</b>	.....
<b>Date</b>	.....

1. Can you describe the end to end story from the occurrence of the failure until the resolution phase.?

a. How failure is detected?

.....

b. Who signals that failure?

.....

c. What are the steps of resolutions of the failure?

.....

d. Who provides the resolution method?

.....

2. How the machine maintenance information is collected and managed today.?

a. On which support?

Paper ?

Digital ?

If digital:  Raw file?  CMMS? .....

Other? .....

b. Maintenance report format

<input type="checkbox"/> Form with list of questions	<input type="checkbox"/> Text report
.....	
<input type="checkbox"/> One single form for all failures	<input type="checkbox"/> Several forms for different failures
.....	

c. Who fills (or files) a maintenance report?

Only one person

Many

.....

d. What happens when the maintenance procedure is not known in advance?

.....

e. What are the roles of involved people?

.....

f. Who reports what?

.....

g. Who is the receiving part?

.....

h. Is there a validation/transformation phase of the data that occurs during all the maintenance process?

Yes  No

i. The reports are accessible for everyone within the company?

Yes  No

j. b. Who is eligible to read such reports?

.....

---

3. What kind of details are included in the report? please give some examples?

.....

---

4. Are the maintenance reports saved and archived?

a. How

.....

b. During how much time ?

.....

---

5. Are the old maintenance reports reused to solve similar problems?

.....

a. How often do you reuse old reports?

.....

b. Which kind of details are relevant in old reports?

.....

---

6. In your point of view, in case of a company having many factories around the world, do they transfer maintenance reports from a factory to another?

a. How ?

.....

b. What do you think about sharing this information with other factories?

.....

c. What do you think about sharing this information with other companies?

.....

d. Is there currently any maintenance information sharing between companies?

.....

---

7. Are there some sensitive data within the machine maintenance reports?

Yes  No .....

a. Do you keep track of who reported error?

Yes  No .....

b. Details like location; which site, which machine, at what time are part of the report?

Yes  No .....

c. Details about the process or configuration can be included in the report?

Yes  No .....

a. In case of multiple machines are connected, do you include details about other machines?

Yes  No .....

a. If you consider that there is no sensitive data in these reports, Are you ready to make your maintenance reports public?

Yes  No .....

---

8. Can you give some examples of the sensitive data?

.....

---

9. Do you know how data can be anonymized in order to keep the resulted data meaningful and useful for machine maintenance?

.....

---

10. Can you share with us some failure reports? It will be only for research purpose and of course with a non-divulgence agreement.

.....



**Titre :** Le partage des connaissances de la maintenance industrielle via des graphes de connaissances

**Mots clés :** Industrie 4.0, Internet des objets industriel, Maintenance industrielle, Graphes de connaissances, Partage de connaissances, Protection de de données sensibles, Web Sémantique

**Résumé :** Autrefois considérée comme faisant partie des coûts généraux de l'entreprise, la maintenance est devenue critique pour les industriels et une véritable source de données. Malgré les sommes importantes investies dans l'industrie intelligente, les pratiques artisanales en maintenance dominent toujours le paysage industriel. Dans cette thèse, nous étudions le partage des connaissances comme solution potentielle qui peut inverser la tendance et améliorer l'activité de maintenance pour se conformer aux principes de l'Industrie 4.0. Nous considérons les graphes de connaissances comme un outil de collecte et de partage des données de maintenance entre les différents acteurs de l'industrie. Notre approche a été validée par plusieurs experts du domaine et a séduit de nombreux clients de Schneider-Electric. Comme contributions, nous avons conçu et prototypé divers services pour collecter, agréger et partager efficacement les données de maintenance tout en préservant la confidentialité des données sensibles.

**Title :** Enabling Industrial Maintenance Knowledge Sharing by using Knowledge Graphs

**Keywords :** Industry 4.0, Industrial Internet of Things, Machine Maintenance, Knowledge Graphs, Knowledge Sharing, Privacy Preservation, Semantic Web

**Abstract :** Formerly considered as part of general enterprise costs, industrial maintenance has become critical for business continuity and a real source of data. Despite the heavy investments made by companies in smart manufacturing, traditional maintenance practices still dominate the industrial landscape. In this Ph.D., we investigate maintenance knowledge sharing as a potential solution that can invert the trend and enhance the maintenance activity to comply with the Industry 4.0 spirit. We specifically consider the knowledge graphs as an enabler to share the maintenance knowledge among the different industry players. Our approach was validated by several interviewed domain experts and attracted many Schneider-Electric customers. As contributions, we designed and prototyped various services to efficiently collect, aggregate and share the maintenance knowledge while preserving sensitive data privacy.

