STEaMINg : semantic time evolving models for industry 4.0

Franco Giustozzi

Normandie Université

THÈSE

Pour obtenir le diplôme de doctorat

Spécialité Informatique

Préparée au sein de l'Institut National des Sciences Appliquées Rouen Normandie

STEaMINg
Semantic Time Evolving Models for Industry 4.0

Présentée et soutenue par

Franco Giustozzi

Thèse soutenue publiquement le 14 décembre 2020 devant le jury composé de		
Anne HÅKANSSON	Professeure, UiT The Arctic University of Norway	Rapportrice
Davy MONTICOLO	Professeur, ERPI, Université de Lorraine	Rapporteur
Nathalie PERNELLE	Professeure, LIPN, Université Sorbonne Paris Nord	Examinatrice
Edward SZCZERBICKI	Professeur, Gdansk University of Technology	Examinateur
Habib ABDULRAB	Professeur, LITIS, INSA Rouen Normandie	Examinateur
Julien SAUNIER	Maître de conférences, LITIS, INSA Rouen Normandie	Co-encadrant
Cecilia ZANNI-MERK	Professeure, LITIS, INSA Rouen Normandie	Directrice de thèse

Thèse dirigée par Cecilia Zanni-Merk et Julien Saunier

INSA ROUEN NORMANDIE

litis

Abstract

Industry 4.0 aims to improve the production and associated services through the digitalization and automation of manufacturing processes. A key characteristic of factories in Industry 4.0 is that assets and machines in plants are equipped with sensors which collect data for effective equipment condition based monitoring. This monitoring of all the equipment involved in a manufacturing process allows early detection of undesirable behaviors or situations that could lead to failures, which affect the performance, energy-use and reliability of manufacturing processes. Through the early detection of these situations, proactive decisions can be made to avoid production downtime. In order to detect these situations the data collected by the sensors must be interpreted. This is a challenging task as it requires the integration and processing of heterogeneous data coming from different sources, with different temporal resolutions and different underlying meanings. Moreover, to create added value out of these data, it must be combined with domain knowledge containing resource specification and planning information. In this context, semantic web technologies are increasingly considered as key technologies to improve data integration. In particular, ontologies emerge as a pertinent method to represent knowledge of manufacturing in a machine-interpretable way through the construction of semantic models. These models provide a virtual representation of the resources as well as the modeling of relevant situations from expert knowledge. In a real factory resources execute their tasks over time and under different conditions (contexts). It is necessary that the model represents in which situation(s) the resource is performing its tasks to allow a more informed decision making, since the actions to take and rules that manage the task may vary according to the current situation. In other words, the model must evolve in order to represent in which situation(s) the resource is concerned during the execution of its tasks.

The main objective of this thesis is to address the evolution of semantic models in Industry 4.0. To this end, firstly we propose a semantic model (ontology) for the manufacturing domain that represents the resources and processes that are part of a factory, with special emphasis on modeling the context of these resources and processes. Relevant situations that combine sensor observations with domain knowledge are also represented in the model. Secondly, an approach that uses stream reasoning to detect these situations that lead to potential failures is introduced. This approach enriches data collected from sensors with contextual information using the proposed semantic model. The use of stream reasoning facilitates the integration of data from different data sources, different temporal resolutions as well as the processing of these data in real time. This allows to derive high-level situations from lower-level context and sensor information. Detecting situations can trigger actions to adapt the process behavior, and in turn, this change in behavior can lead to the generation of new contexts leading to new situations. These situations can have different levels of severity, and can be nested in different ways. Dealing with the rich relations among situations requires an efficient approach to organize them. Therefore, we propose a method to build a lattice, ordering those situations depending on the constraints they rely on. This lattice represents a road-map of all the situations that can be reached from a given one, desirable or undesirable. This helps in decision support, by allowing the identification of the actions that can be taken to correct the abnormality avoiding in this way the interruption of the manufacturing processes. Finally, an industrial application scenario for the proposed approach is described. Through it, we verify that (1) the proposed semantic model is generic and extensible to accommodate a wide spectrum of manufacturing processes. Furthermore, its modular architecture allows

to describe manufacturing capabilities of manufacturing resources in different levels of modularity; (2) the use of stream reasoning together with classical reasoning approaches allow to detect abnormal situations as well as their possible causes in a suitable way; and (3) the exploitation of the lattice helps in decision making tasks, by allowing the identification of the actions that can be taken to correct the abnormality.

Keywords: *Semantic Technologies; Ontology; Context Modeling; Industry 4.0; Stream Reasoning; Condition Monitoring.*

Résumé

L'Industrie 4.0 vise à améliorer la production et les services associés grâce à la numérisation et à l'automatisation des processus de fabrication. Une caractéristique clé des usines de Industry 4.0 est que les actifs et les machines des usines sont équipés de capteurs qui collectent des données pour une surveillance efficace basée sur l'état des équipements. Cette surveillance de tous les équipements impliqués dans un processus de fabrication permet la détection précoce de comportements indésirables ou de situations pouvant conduire à des défaillances, qui affectent les performances, la consommation d'énergie et la fiabilité des processus de fabrication. Grâce à la détection précoce de ces situations, des décisions proactives peuvent être prises pour éviter les arrêts de production. Afin de détecter ces situations, les données recueillies par les capteurs doivent être interprétées. C'est une tâche difficile car elle nécessite l'intégration et le traitement de données hétérogènes provenant de différentes sources, avec des résolutions temporelles et des significations sous-jacentes différentes. De plus, pour créer une valeur ajoutée à partir de ces données, il faut les combiner avec des connaissances de domaine contenant des informations sur la spécification et la planification des ressources. Dans ce contexte, les technologies du web sémantique sont de plus en plus considérées comme des technologies clés pour améliorer l'intégration des données. En particulier, les ontologies apparaissent comme une méthode pertinente pour représenter la connaissance de la fabrication d'une manière interprétable par les machines grâce à la construction de modèles sémantiques. Ces modèles fournissent une représentation virtuelle des ressources ainsi que la modélisation de situations pertinentes à partir de connaissances d'experts. Dans une usine réelle, les ressources exécutent leurs tâches au fil du temps et dans différentes conditions (contextes). Il est nécessaire que le modèle représente dans quelle(s) situation(s) la ressource exécute ses tâches pour permettre une prise de décision plus éclairée, car les actions à entreprendre et les règles qui gèrent la tâche peuvent varier en fonction de la situation actuelle. En d'autres termes, le modèle doit évoluer afin de représenter dans quelle(s) situation(s) la ressource est concernée lors de l'exécution de ses tâches.

L'objectif principal de cette thèse est d'aborder l'évolution des modèles sémantiques dans l'industrie 4.0. À cette fin, nous proposons tout d'abord un modèle sémantique (ontologie) pour le domaine de la fabrication qui représente les ressources et les processus qui font partie d'une usine, en mettant particulièrement l'accent sur la modélisation du contexte de ces ressources et processus. Les situations pertinentes qui combinent les observations des capteurs avec la connaissance du domaine sont également représentées dans le modèle. Ensuite, une approche qui utilise le raisonnement par flux pour détecter ces situations qui conduisent à des défaillances potentielles est introduite. Cette approche enrichit les données collectées par les capteurs avec des informations contextuelles en utilisant le modèle sémantique proposé. L'utilisation du raisonnement en flux facilite l'intégration de données provenant de différentes sources, de différentes résolutions temporelles ainsi que le traitement de ces données en temps réel. Cela permet de dériver des situations de haut niveau à partir d'informations contextuelles et de capteurs de niveau inférieur. La détection de situations peut déclencher des actions pour adapter le comportement du processus, et à son tour, ce changement de comportement peut conduire à la génération de nouveaux contextes conduisant à de nouvelles situations. Ces situations peuvent avoir différents niveaux de gravité et peuvent être imbriquées de différentes manières. La gestion des relations riches entre les situations nécessite une approche efficace pour les organiser. C'est pourquoi nous proposons une méthode pour construire un treillis, en ordonnant ces situations en fonction des con-

traintes sur lesquelles elles reposent. Ce treillis représente une feuille de route de toutes les situations qui peuvent être atteintes à partir d'une situation donnée, souhaitable ou indésirable. Cela aide à l'aide à la décision, en permettant l'identification des actions qui peuvent être prises pour corriger l'anomalie en évitant ainsi l'interruption des processus de fabrication. Enfin, un scénario d'application industrielle pour l'approche proposée est décrit. À travers celui-ci, nous vérifions que (1) le modèle sémantique proposé est générique et extensible pour s'adapter à un large éventail de procédés de fabrication. De plus, son architecture modulaire permet de décrire les capacités de fabrication des ressources de fabrication à différents niveaux de modularité ; (2) l'utilisation du raisonnement en flux avec les approches de raisonnement classiques permet de détecter les situations anormales ainsi que leurs causes possibles de manière appropriée ; et (3) l'exploitation du treillis aide à la prise de décision, en permettant l'identification des actions qui peuvent être prises pour corriger l'anomalie.

Mots-clés : *Technologies sémantiques ; Ontologie ; Modélisation du contexte ; Industrie 4.0 ; Raisonnement en flux ; Surveillance des Processus.*

To my sister and my parents.

Acknowledgements

In these lines I would like to express my most sincere gratitude to all those people who have supported me and helped me in the development of this thesis.

I would like to thank the reviewers for the care with which they have evaluated this manuscript and the consideration they have given to my thesis work. Their constructive comments and questions have provided me with new and very insightful perspectives on my research works.

I wish to express my deepest appreciation to my thesis supervisors Cecilia and Julien for their patience, dedication, guidance and motivation. They convincingly guided and encouraged me to face every challenge that arose during the development of this thesis even when the road got tough. It has been a privilege to be able to count on their guidance and help. Not only have they helped me in my research work by giving me advice, inspiring ideas and enriching feedback but they have also contributed a lot to my personal development.

I would also like to pay my special regards to all the staff at LITIS and INSA Rouen for the friendly and comfortable atmosphere. It is a pleasure to work in such an environment. I want to thank Brigitte, Sandra, Fabienne and Mathieu for their patience and always willing to help me with all the administrative tasks and paperwork. I also want to thank all my colleagues. I am sure I am forgetting someone, but thanks to Jean Baptiste, Antoine, Benjamin, Rachel, Usman, Mathieu, Caterine, Qiu, Sandratra, Maël, Cyprien, Ismaila, Henrique and Marwa. During these years we have shared many great moments and have certainly made the everyday life more enjoyable. With many of them a great friendship was born. A special thanks to Jean Baptiste for receiving me in France. His support was fundamental for me to integrate and adapt to the daily life in Rouen. I also want to thank Antoine, who helped me during my first year in France.

I also want to extend my thanks to the people who have accompanied me outside the professional environment.

I wish to express my deepest gratitude to Manon and her family. Their friendship and kindness was a fundamental support for me. They have welcomed me into their home and always treated me with great kindness. They have made me feel at home.

Moreover, my appreciation goes to Sofia for her support and understanding. Beyond the constant *distance* that always separates us and the dynamism of life she has been an inexhaustible source of strength and admiration.

Last but not least, I want to express my heartfelt thanks to my family. I would like to recognize the invaluable and unconditional support that they all provided over the years and in every decision I have made. They have always encouraged me to pursue my goals. This achievement would not have been possible without them.

Thank you. Merci. Gracias.

Table of Contents

List of Figures

List of Tables

Chapter 1

Synthèse de la thèse en français

1.1 Introduction

Contexte

Historiquement, les révolutions industrielles ont changé la façon dont la société vit et travaille. La première révolution industrielle (du 18ème au 19ème siècle) a utilisé l'énergie de l'eau et de la vapeur pour mécaniser la production [DD79]. La deuxième révolution industrielle (1870 à 1914) a utilisé l'énergie électrique pour permettre la production de masse de biens [Mok98]. La troisième révolution industrielle (1980 à 2010) a utilisé l'électronique et les technologies de l'information pour automatiser la production [Rif11]. Aujourd'hui, une quatrième révolution industrielle, également connue sous le nom d'Industrie 4.0, s'appuie sur la troisième. L'industrie 4.0 se caractérise par une fusion des technologies qui brouille les lignes entre les sphères physique et numérique. Parmi ces technologies, on trouve l'intelligence artificielle (IA), la robotique, l'Internet des objets (IoT) et les systèmes cyberphysiques. La figure 1.8 illustre ces révolutions industrielles et les technologies clés associées à chacune d'entre elles.

L'objectif principal de l'industrie 4.0 est donc d'améliorer la production et les services associés par l'utilisation de technologies innovantes [SMB+17, WWLZ16]. Pour garantir une productivité, une disponibilité et une efficacité élevées des processus de fabrication, la détection de situations anormales dans les lignes de production est une question cruciale pour les fabricants [Has11]. Afin de résoudre ce problème, les usines s'appuient sur la surveillance de l'état du système (*condition monitoring*). Il s'agit de surveiller tous les équipements impliqués dans un processus de fabrication afin de détecter rapidement les comportements ou situations indésirables qui pourraient conduire à des anomalies, qui affectent leurs performances, leur consommation d'énergie ou leur fiabilité. Grâce à la détection précoce de ces situations, des décisions proactives peuvent être prises pour éviter les arrêts de production. Ces décisions peuvent impliquer, par exemple, de modifier les paramètres du processus pour en adapter le comportement.

Une caractéristique importante de la production industrielle dans l'industrie 4.0 est que les éléments physiques tels que les capteurs, les actionneurs et les machines de l'entreprise sont connectés entre eux et à l'internet. Dans cet environnement, les capteurs

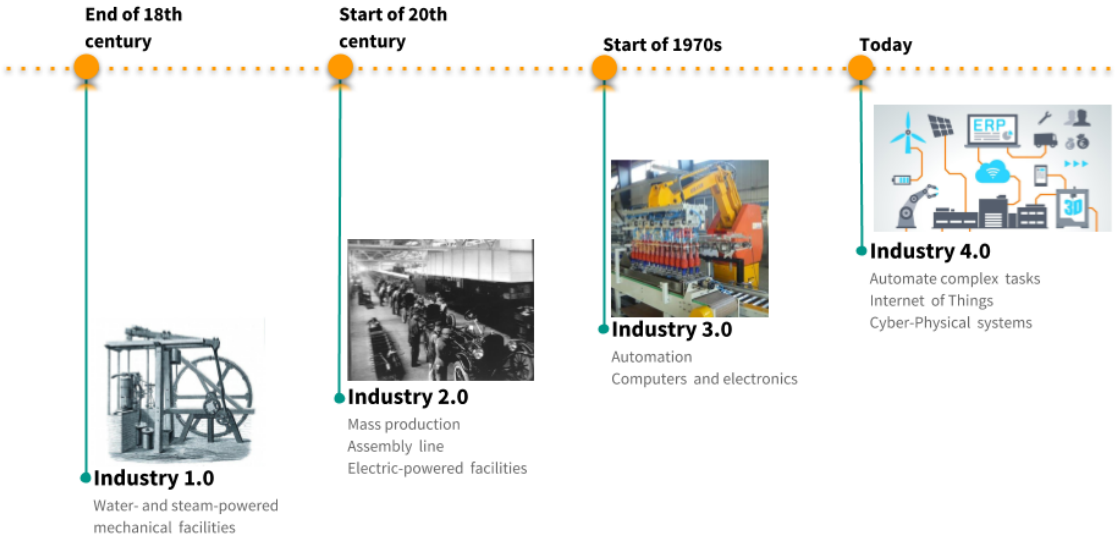

Figure 1.1 – Histoire des révolutions industrielles.[1]

génèrent une quantité croissante de données qui peuvent être utilisées pour une surveillance efficace de l'état des équipements. Afin de détecter des situations anormales, les données collectées par les capteurs doivent être interprétées. C'est une tâche difficile car elle nécessite l'intégration et le traitement de données hétérogènes provenant de différentes sources, avec des résolutions temporelles et des significations sous-jacentes différentes. De plus, un point essentiel à considérer est que la surveillance des processus industriels devrait dépendre non seulement de leur état interne et des interactions des utilisateurs, mais aussi du contexte de leur exécution. La prise en compte du contexte des processus industriels permet de fournir des informations à valeur ajoutée pour améliorer leurs performances. Le contexte est toute information qui peut être utilisée pour caractériser la situation d'une entité [DA00]. Il est généralement considéré comme un mélange de données géospatiales, de capteurs environnementaux, de descriptions de services, entre autres. Dans le contexte de l'industrie, un modèle de contexte dynamique doit non seulement prendre en compte le contexte des outils, des machines, des pièces et des produits, mais aussi la planification des processus de fabrication, la spécification des ressources et la configuration du système de contrôle. Les données contextuelles sont soumises à des changements constants et peuvent être très hétérogènes.

Les technologies du web sémantique ont prouvé leur efficacité pour traiter la question de l'intégration des données. Le web sémantique est une extension du web mondial qui combine l'ingénierie de la connaissance et les méthodes d'IA pour représenter, intégrer et raisonner sur des données et des connaissances à travers des ontologies et des règles. En informatique, une ontologie est considérée comme une spécification explicite d'une conceptualisation pour un domaine d'intérêt. Les ontologies apparaissent comme une méthode pertinente pour représenter les connaissances de toute nature (en parti-

[1] adapté de: https://www.i-scoop.eu/industry-4-0/

culier, les connaissances de fabrication) d'une manière interprétable par les machines. Les modèles sémantiques ainsi construits fournissent une représentation virtuelle des ressources d'une usine de fabrication ainsi que des situations pertinentes qui leur sont associées. De plus, le raisonnement sur les ontologies, grâce à ses extensions basées sur des règles, permet de transformer les observations brutes collectées par les capteurs en abstractions de plus haut niveau, telles que des situations d'intérêt, qui sont significatives pour les humains et permettent de mieux comprendre le monde physique pour soutenir les tâches de prise de décision. Par exemple, les observations des capteurs peuvent être utilisées pour optimiser la consommation d'énergie dans une chaîne de production afin d'éviter les pannes. Ce processus de transformation des données est illustré à l'aide de la célèbre "hiérarchie des connaissances" [Row07] illustrée dans la Figure 1.9.

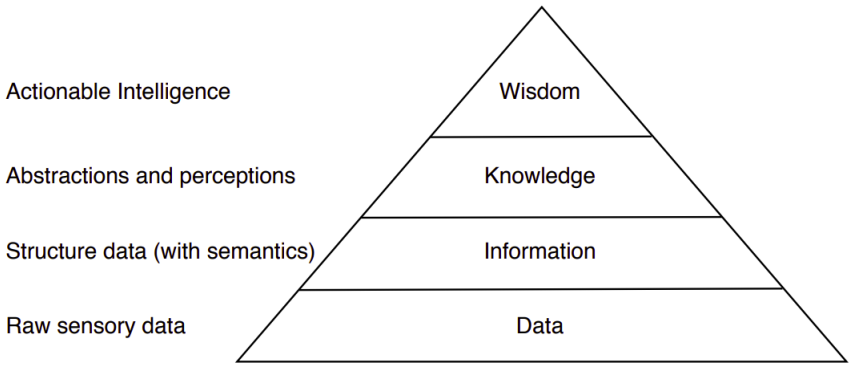

Figure 1.2 – Hiérarchie de la connaissance (pris de [Row07])

Les solutions actuelles de raisonnement sur les ontologies ont été traditionnellement développées pour des données statiques ou à évolution lente. La nature hautement dynamique des données dans le domaine industriel introduit de nouvelles questions. Pour y faire face, un certain nombre de travaux récents proposent d'unifier le raisonnement et le traitement des flux de données, donnant naissance au domaine de recherche du raisonnement sur les flux de données (*stream reasoning*). Ce domaine s'intéresse aux systèmes de décision basés sur le traitement continu des flux de données ainsi que sur des connaissances de fond riches [SCDV+19].

Dans ces circonstances, le modèle sémantique lui-même doit évoluer afin de représenter dans quelle(s) situation(s) se trouve(nt) la ou les ressources pendant l'exécution de ses tâches pour soutenir la prise de décision.

Par conséquent, cette thèse étudie l'utilisation des méthodes de représentation des connaissances, en particulier les technologies du web sémantique, pour construire un modèle sémantique évolutif qui représente le domaine industriel, en mettant l'accent sur la modélisation du contexte pour fournir la notion de situation.

Motivation

Comme mentionné dans la section précédente, les modèles sémantiques sont utiles pour la représentation des connaissances sur un domaine particulier, en l'occurrence l'industrie 4.0. Cependant, ces modèles sémantiques sont plutôt statiques, ce qui signifie qu'ils ne

répondent pas à la dynamique des processus de fabrication. Pour que le modèle puisse représenter les situations dans lesquelles les processus sont exécutés, il est nécessaire que le modèle sémantique évolue afin de représenter ce qui se passe dans le monde réel.

Le modèle sémantique peut évoluer pour différentes raisons. La première est liée à un changement dans la structure du modèle lui-même, c'est-à-dire l'ajout ou la suppression de concepts et de relations. Ce type de changement est étudié dans le domaine de l'évolution des ontologies [SMMS02, Sto04, NK04, PDTC07] et n'est pas abordé dans cette thèse. La deuxième possibilité est liée à l'ajout d'instances à des concepts déjà définis et à l'ajout de relations sur des instances existantes. Un exemple peut être l'ajout d'une ressource physique à l'usine qui serait reflétée comme une nouvelle instance du concept de ressource correspondant dans le modèle sémantique. Un autre exemple peut être l'ajout d'une situation détectée et le fait qu'une certaine ressource est impliquée dans cette situation (c'est-à-dire un nouveau lien entre deux instances à travers une relation existante).

Les situations d'intérêt dans le domaine industriel dépendent des données des capteurs et de la connaissance du domaine. Outre les informations statiques sur les entités industrielles, des informations dynamiques, telles que la manière dont les processus industriels sont exécutés, doivent être décrites. Par conséquent, les données collectées par les capteurs déployés dans une usine doivent être explicitement représentées pour décrire l'état du processus. C'est la première exigence pour que le modèle sémantique soit construit.

Une autre exigence est la représentation des relations temporelles entre les processus et des relations spatiales entre les entités et les emplacements dans l'usine. Ces relations sont utilisées pour identifier différentes situations d'intérêt. Cela implique de traiter des informations qui évoluent dans le temps, comme les changements de situation(s) qu'une machine peut traverser ou les changements de valeurs d'un paramètre de la machine en fonction des différentes décisions prises. Par conséquent, le modèle sémantique doit fournir suffisamment d'expressivité pour saisir des connaissances de haut niveau, telles que des situations, à partir de données annotées dans le temps provenant de capteurs.

Plus précisément, les exigences auxquelles le modèle sémantique doit répondre sont les suivantes :

Intégration de données. L'intégration des informations provenant de différentes sources est une caractéristique essentielle. De plus, l'intégration de données provenant de capteurs (flux de données) avec des connaissances de base décrivant le domaine d'application est également nécessaire (par exemple, les données de flux collectées par une machine sont combinées avec des connaissances de base sur les différentes contraintes qui indiquent un comportement anormal).

Représentation temporelle. Le temps joue un rôle central dans cette thèse. Cela exige une représentation adéquate du temps, où les données peuvent être annotées avec leur heure d'occurrence et leur validité. Outre le traitement des données en continu, il est également nécessaire de gérer des données historiques, c'est-à-dire des informations annotées dans le temps sur les états précédents de la ressource analysée (par exemple, l'historique des relevés de température d'une certaine machine). Ces informations peu-

vent être utilisées, par exemple, pour identifier des tendances, pour extraire des statistiques ou pour comparer des informations antérieures et actuelles afin d'identifier de nouveaux types d'anomalies ou de situations.

Efficience. En ce qui concerne la détection des situations, l'efficacité du traitement est liée à la capacité de faire face à une grande quantité de données collectées pour générer de nouveaux résultats en temps utile (par exemple, les situations anormales doivent être détectées rapidement pour permettre des actions de résolution).

Aide à la décision. Une fois qu'une situation susceptible de conduire à des défaillances est détectée, une action doit être entreprise pour la contrer. Les actions modifient l'état du système et conduisent à d'autres situations. Afin d'aider à la prise de décision et de choisir l'action la plus appropriée, il est nécessaire de comprendre les relations entre les situations.

Ces exigences motivent la proposition d'un nouveau cadre pour traiter l'évolution des modèles sémantiques dans l'industrie 4.0. Le cadre proposé utilise les technologies sémantiques pour représenter le domaine de la fabrication en mettant l'accent sur la modélisation du contexte des ressources impliquées dans une usine. il vise à détecter les situations qui peuvent conduire les processus de fabrication dans l'industrie 4.0 à des défaillances et leurs causes possibles. Grâce à la détection de ces situations et de leurs causes, des décisions appropriées peuvent être prises pour éviter l'interruption des processus de fabrication.

Contributions

Cette thèse propose un cadre pour aborder l'évolution des modèles sémantiques dans l'industrie 4.0. Un aperçu du cadre est présenté dans la figure 1.10, les rectangles en pointillés bleus représentent les principales contributions de cette thèse qui sont décrites ci-dessous.

La première contribution est liée au développement du modèle sémantique ; et les autres contributions sont liées à la gestion de l'évolution du modèle sémantique.

- Un modèle de connaissance ontologique pour le domaine de l'industrie est proposé pour fournir une représentation déclarative et abstraite des ressources, des processus, des capteurs et des relations entre eux. En outre, le modèle est fortement orienté vers la modélisation du contexte des ressources impliquées dans un processus de fabrication. Les situations pertinentes qui combinent les observations des capteurs et la connaissance du domaine y sont également représentées.

- Une approche qui utilise le raisonnement par flux pour détecter les situations pertinentes qui conduisent à des défaillances potentielles est introduite. Cette approche enrichit les données collectées par les capteurs avec des informations contextuelles en utilisant le modèle sémantique proposé. L'utilisation du raisonnement sur les flux de données facilite l'intégration de données provenant de différentes sources, de différentes résolutions temporelles ainsi que le traitement de ces données en

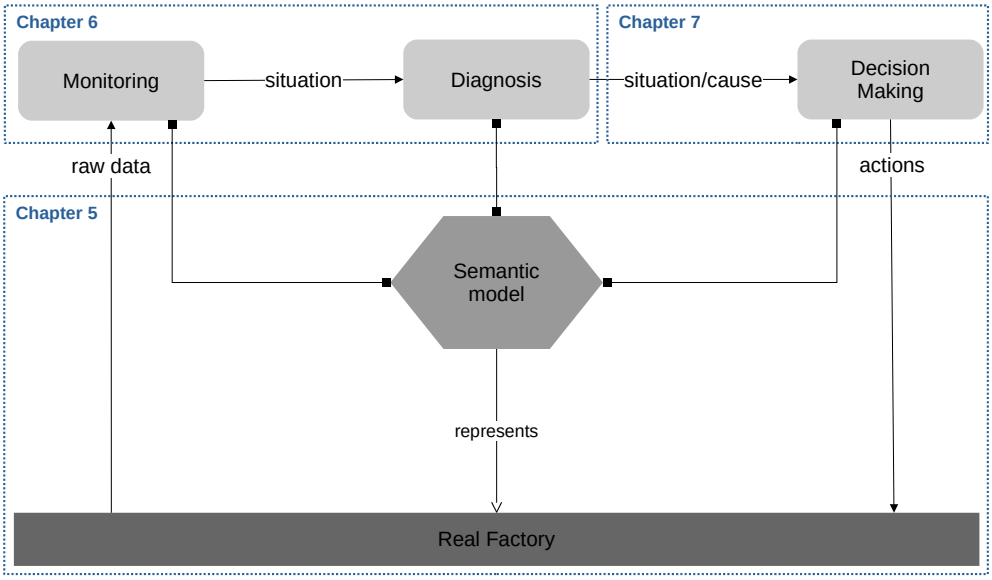

Figure 1.3 – Aperçu du schéma proposé.

temps réel. Cela permet d'identifier des situations à partir de flux de données contextuelles et de capteurs de bas niveau combinés à des connaissances de fond. En outre, notre proposition utilise des approches de raisonnement classiques pour déterminer les causes possibles qui génèrent la situation lorsque les résultats du raisonnement sur les flux ne sont pas suffisants.

- Les situations détectées peuvent déclencher des actions visant à adapter le comportement du processus, et ce changement de comportement peut à son tour entraîner la création de nouveaux contextes conduisant à de nouvelles situations. Pour organiser les situations en fonction des contraintes sur lesquelles elles reposent, la construction d'un treillis est proposée. Ce treillis représente une feuille de route de toutes les situations, désirables ou non, qui peuvent être atteintes à partir d'une situation donnée. Cela aide à l'aide à la décision, en permettant l'identification des actions qui peuvent être prises pour corriger le comportement du processus.

Enfin, un scénario d'application industrielle pour l'approche proposée est étudié. Au moyen de celui-ci, nous vérifions que (i) le modèle sémantique proposé est générique et extensible pour s'adapter à un large spectre de processus de fabrication, et que son architecture modulaire permet la description de ressources de fabrication avec différents niveaux de modularité ; (ii) l'utilisation du raisonnement en flux avec les approches de raisonnement classiques permet la détection de situations anormales ainsi que de leurs causes possibles de manière appropriée ; et (iii) l'exploitation du treillis aide à la prise de décision, en permettant l'identification des actions qui peuvent être prises pour corriger le comportement anormal du processus.

1.2 État de l'art

Dans cette section, la première sous-section présente les concepts de base des systèmes intelligents. L'architecture KREM est décrite ainsi que ses quatre composantes. La deuxième sous-section décrit les approches existantes pour la surveillance de l'état des systèmes dans l'Industrie 4.0. Nous nous concentrons en particulier sur les modèles ontologiques et leurs extensions basées sur des règles qui sont pertinentes pour la surveillance du système. Enfin, à la fin de chaque sous-section, un résumé est présenté ainsi que quelques conclusions pertinentes.

1.2.1 Aperçu des systèmes intelligents

Un système intelligent est un programme informatique qui utilise des techniques d'intelligence artificielle pour résoudre des problèmes complexes qui seraient normalement traités par une personne ayant une expertise [Mil08]. Ces systèmes ont un modèle de calcul d'un certain domaine d'intérêt dans laquelle les symboles servent de substituts aux artefacts du domaine du monde réel, tels que les objets, événements et relations, etc [Sow99]. Une base de connaissances stocke ces symboles et le système effectue un raisonnement en manipulant ces symboles.

Les applications peuvent fonder leurs décisions sur des questions posées à la base de connaissances. Elles sont programmées pour résoudre des problèmes de manière similaire à celle d'un expert, par exemple faire des conclusions basées sur le cas courant et adopter la bonne stratégie pour résoudre le problème; ils peuvent traiter des informations incomplètes en demandant des informations complémentaires; ils disposent d'une interface utilisateur qui leur permet de faire des demandes d'information appropriées et d'expliquer comment ils sont parvenus à leurs réponses; et enfin ils peuvent être développés en réutilisant des structures, des règles et des méthodes de résolution de problèmes génériques.

Dans la suite, nous présentons les fondements théoriques et les concepts de base des systèmes intelligents à l'aide de l'architecture KREM (Knowledge, Rules, Experience, and Meta-Knowledge) [ZM15]. Les quatre composantes du KREM sont :

La composante Connaissance (*Knowledge*)

La composante *Knowledge* contient des connaissances sur un certain domaine au moyen d'un modèle conceptuel formel. Parmi les modèles conceptuels existants, les ontologies ont été largement utilisées pour formaliser la connaissance d'un domaine. En informatique, la définition la plus citée d'une ontologie est la suivante: *Une ontologie est une spécification formelle et explicite d'une conceptualisation partagée d'un domaine d'intérêt* [Gru93].

Le Web sémantique est une initiative du World Wide Web Consortium [2] (W3C) pour une nouvelle génération du web mondial. L'idée a été présentée à l'origine par Tim Berners-Lee comme une solution pour améliorer l'exploitation des connaissances stockées sur le

[2] https://www.w3.org/

web. Les technologies du web sémantique offrent un ensemble de solutions pour gérer la sémantique des données. Dans le web sémantique, les ontologies sont utilisées comme modèles de représentation des connaissances et le langage utilisé pour écrire ces modèles est le Web Ontology Language (OWL). OWL est basé sur le Resource Description Framework (RDF) et le Resource Description Framework Schema (RDFS), qui sont superposés au langage de balisage extensible (XML). OWL fournit un vocabulaire supplémentaire ainsi qu'une sémantique formelle pour faciliter une meilleure interprétation automatique du contenu Web que celle supportée par XML, RDF et RDF Schema [MVH+04].

La composante Règles (*Rules*)

La composante *Rules* d'un système intelligent permet différents types de raisonnement sur les instances des ontologies contenues dans la composante *Knowledge*. Nous considérons le raisonnement comme la capacité à générer des conclusions non triviales à partir de prémisses ou d'hypothèses [ARFS12]. Les faits affirmés connus sont appelés connaissances explicites, tandis que les faits déduits sont des connaissances implicites. Le Semantic Web Rule Language (SWRL) étend les axiomes OWL avec des règles similaires à IF-THEN. Ces règles logiques sont exécutées par un raisonneur, qui est un logiciel capable de déduire des conséquences logiques à partir d'un ensemble de faits ou d'axiomes affirmés [Abb12]. Certains des moteurs de règles les plus connus dans la communauté du web sémantique sont Jess[3], Jena[4] et Drools[5].

Les solutions actuelles de raisonnement sur les ontologies ont été développées pour des données statiques ou à évolution lente, ce qui n'est pas le cas pour la nature dynamique des données dans le domaine de l'Industrie 4.0 où les machines de fabrication sont équipées de capteurs qui collectent des données en continu. Les données des capteurs sont un exemple de données en flux continu qui changent rapidement. Ces données doivent être rapidement consommées et traitées. Par exemple, si les valeurs d'une propriété particulière baissent par rapport à son seuil autorisé, ces informations doivent être récupérées rapidement et une décision appropriée doit être prise.

Pour combler cette lacune, un certain nombre de travaux récents proposent d'unifier le raisonnement et le traitement des flux, ce qui a donné naissance au domaine de recherche du raisonnement sur les flux de données (Stream Reasoning - SR).

La composante Expérience (*Experience*)

La composante *Experience* permet la capitalisation et la réutilisation des connaissances antérieures. Elle améliore le modèle de connaissance en le complétant progressivement avec l'expérience acquise lors des interventions des experts humains.

[3]https://jess.sandia.gov/
[4]https://jena.apache.org/
[5]https://www.drools.org/

La composante Méta-Connaissance (*Meta-Knowledge*)

La composante *Meta-Knowledge* comprend les connaissances sur les trois autres composantes, selon l'application. Les méta-connaissances sont des connaissances sur la connaissance du domaine, sur les règles et sur l'expérience. Cette composante fournit notamment une représentation formelle du contexte, afin d'identifier celui dans lequel les actions doivent être prises. Dans le domaine industriel, la méta-connaissance peut contrôler le déclenchement de différents ensembles d'actions ou de règles en fonction de différents contextes. Par exemple, les règles de détection de l'écart de température d'une machine au-dessus de son seuil peuvent changer en fonction de l'humidité de l'environnement de la machine. De cette manière, l'utilisation de la méta-connaissance peut orienter l'exécution de systèmes intelligents. Le contexte a été étudié dans des domaines tels que les systèmes basés sur la connaissance et omniprésents, soit pour traiter dynamiquement des connaissances complexes [GMD12], soit pour fournir des interfaces humaines intelligentes [DAS01].

Il existe diverses représentations du contexte dans différents domaines de recherche. Selon [SAW94b], la connaissance du contexte est la capacité d'un système à rassembler des informations sur son contexte ou son environnement à un moment donné et à adapter son comportement en conséquence en fournissant des services appropriés. La définition du contexte la plus largement citée est la suivante : *le contexte est toute information pouvant être utilisée pour caractériser la situation d'une entité. Une entité est un utilisateur, un lieu, ou un objet physique ou informatique qui est considéré comme pertinent pour l'interaction entre une entité et une application, y compris l'entité et l'application elles-mêmes* (traduit de [DA00]).

Ces entités et applications sont capables de réagir spécifiquement à leur emplacement actuel, au temps et à d'autres attributs de l'environnement et d'adapter leur comportement sans intervention explicite de l'utilisateur, visant ainsi à accroître la convivialité et l'efficacité. Dans cette section, nous étudions les modèles existants pour le traitement du contexte et analysons leurs avantages et inconvénients par rapport aux exigences de la version 4.0 de l'industrie.

Conclusion

La façon dont la connaissance du domaine est formalisée détermine la manière dont les règles sont exprimées. L'expérience vient compléter les connaissances et les règles disponibles. Enfin, la méta-connaissance interagit directement avec les règles et l'expérience pour indiquer quelles règles doivent être executées en fonction du contexte du problème à résoudre. Les méta-connaissances sont des connaissances sur la connaissance du domaine, sur les règles ou sur l'expérience. Cette méta-connaissance peut prendre la forme d'un contexte, d'une culture ou de protocoles. Le contexte est l'information qui caractérise une situation en relation avec l'interaction entre les êtres humains, les applications et leur environnement [DA99].

Dans cette section, nous donnons un aperçu des fondements théoriques des systèmes intelligents. Nous encadrons notre approche par l'architecture KREM, qui comporte qua-

tre composantes. Dans la composante *Knowledge*, le concept d'ontologie est introduit ainsi que les concepts clés du Web sémantique et les technologies associées utilisées dans cette thèse. Dans la description du composant *Rules*, nous présentons la notion de raisonnement dans des scénarios statiques et dynamiques. SWRL est un langage puissant pour le raisonnement hors ligne, qui est devenu le langage de règles standard de facto dans le Web sémantique. Le raisonnement sur les flux de données, qui combine le traitement continu des flux de données et le raisonnement avec de riches connaissances de fond, peut être utilisé pour traiter des domaines dynamiques tels que l'Industrie 4.0. Cette thèse ne traite pas de la composante *Experience* de KREM. Dans la section sur la composante *Meta-knowledge*, la définition du contexte est présentée. La représentation du contexte et le raisonnement contextuel dans le domaine de l'Industrie 4.0 peuvent être traités par les technologies sémantiques.

Dans la section suivante, nous passons en revue les approches existantes pour la surveillance de l'état des processus dans l'Industrie 4.0. Nous accordons une attention particulière à celles qui utilisent les technologies du web sémantique pour faciliter les tâches de maintenance prédictive dans l'industrie.

1.2.2 Approches existantes pour la surveillance de l'état du système dans l'Industrie 4.0

La surveillance est une activité cruciale dans l'industrie. Toute défaillance imprévue d'une machine peut dégrader ou même interrompre les processus de fabrication d'une entreprise. Il est donc fondamental de développer une stratégie de surveillance efficace pour prévenir les arrêts de production imprévus, améliorer la fiabilité et réduire les coûts d'exploitation.

Dans cette section, nous présentons les connaissances de base sur l'Industrie 4.0, en détaillant ses quatre éléments clés : l'Internet des objets (IoT), les systèmes cyberphysiques (CPS), l'informatique ubiquitaire (*Cloud Computing*), et les technologies d'analyse de grandes données (*Big Data*). Ensuite, nous décrivons les stratégies de maintenance existantes dans le domaine industriel en mettant l'accent sur les stratégies de maintenance prédictive pour l'Industrie 4.0. Les approches basées d'apprentissage automatique, et les approches basées sur la connaissance des experts pour la surveillance de l'état du système sont également discutées.

Nous nous concentrons particulièrement sur les modèles fondés sur les ontologies et leurs extensions basées sur des règles qui sont pertinentes pour la surveillance des processus de fabrication.

Éléments principaux de l'Industrie 4.0

L'Industrie 4.0 suit et étend la notion de technologie d'automatisation introduite lors de la troisième révolution industrielle en appliquant un ensemble de technologies pour l'échange et le traitement automatique de données dans les usines de fabrication [WSJ17].

Les systèmes cyberphysiques, l'internet des objets, le cloud computing et les tech-

niques d'analyse des grandes données sont les éléments clés de l'Industrie 4.0. Ils permettent l'interconnexion automatique et l'échange de données entre les entités manufacturières. La collecte et l'analyse de ces données peuvent améliorer la productivité et la fiabilité des systèmes de production [LBK15].

L'Industrie 4.0 vise à transformer les usines de fabrication traditionnelles en "usines intelligentes" (*smart factories*) en équipant l'usine de capteurs, d'actionneurs et de systèmes autonomes [DGP19]. De cette façon, les machines de fabrication peuvent atteindre des niveaux élevés d'auto-optimisation et d'automatisation. L'utilisation conjointe des IoT, CPS, Cloud Computing et technologies de Big Data peut aider les systèmes de production à accéder eux-mêmes à leur état pour prendre des décisions intelligentes afin d'éviter des défaillances potentielles [ZTL15].

Stratégies de maintenance dans le contexte industriel.

Les stratégies de maintenance dans le domaine industriel entrent généralement dans l'une des trois catégories suivantes, chacune avec ses problèmes et ses avantages :

- La Maintenance Réactive [Mob02, Swa01] est une méthode de gestion de la maintenance en cas de panne. L'action de maintenance pour la réparation d'un équipement n'est effectuée que lorsque l'équipement est en panne. Une entreprise qui utilise la maintenance réactive ne dépense pas d'argent pour la maintenance tant qu'une machine ou un système ne fonctionne pas. Cependant, le coût de réparation ou de remplacement d'un composant serait potentiellement supérieur à la valeur de production obtenue en le faisant fonctionner jusqu'à la panne. De plus, lorsque les composants commencent à vibrer, à surchauffer et à se casser, des dommages supplémentaires peuvent survenir sur l'équipement, ce qui peut entraîner d'autres réparations coûteuses.

- La Maintenance Préventive [Mob02, WTL+17, Ger13] prévoit des activités de maintenance régulières sur des équipements spécifiques afin de réduire la probabilité de défaillances. La maintenance est effectuée même lorsque la machine fonctionne encore et dans des conditions normales de fonctionnement, de manière à éviter les pannes imprévues avec les temps d'arrêt et les coûts qui y sont associés. Presque tous les programmes de gestion de maintenance préventive sont fondés sur le temps [Mob02, AK12]. Le processus général de maintenance préventive peut être présenté en deux étapes : La première étape consiste à étudier statistiquement les caractéristiques de défaillance de l'équipement en se basant sur l'ensemble des données chronologiques collectées. La deuxième étape consiste à décider des politiques de maintenance optimales qui maximisent la fiabilité et la disponibilité du système aux coûts de maintenance les plus bas. La maintenance préventive réduit les coûts de réparation et les temps d'arrêt imprévus, mais peut entraîner des réparations inutiles si elle est effectuée trop tôt ou des pannes si elle est effectuée trop tard.

- La Maintenance Prédictive [WDD94] vise à prévoir quand l'équipement est susceptible de tomber en panne et à décider quelle activité de maintenance doit être ef-

fectuée de manière à obtenir un bon compromis entre la fréquence et le coût de la maintenance. La maintenance prédictive utilise l'état de fonctionnement réel des systèmes et des composants pour optimiser la production. L'analyse prédictive est basée sur les données collectées par des capteurs connectés aux machines et aux outils, telles que les données de vibration, les données ultrasoniques, la disponibilité opérationnelle, etc. Le modèle traite les informations par des algorithmes prédictifs, découvre des tendances et identifie le moment où l'équipement doit être réparé. Plutôt que de faire fonctionner un équipement ou un composant jusqu'à ce qu'il tombe en panne, ou de le remplacer lorsqu'il a encore une durée de vie utile, la maintenance prédictive aide les entreprises à optimiser leurs stratégies en n'effectuant des activités de maintenance qu'en cas de nécessité absolue. Cependant, comparé à la maintenance réactive et à la maintenance préventive, le coût des dispositifs de surveillance de l'état (par exemple, les capteurs) nécessaires à la maintenance prédictive est souvent élevé. Un autre problème est que le système de maintenance prédictive devient de plus en plus complexe en raison de la collecte de données, de l'analyse des données et de la prise de décision.

La Figure 1.4 résume les plans de maintenance de chacun des trois types de maintenance possibles. La maintenance réactive présente le coût de prévention le plus faible grâce à l'utilisation de la gestion des pannes. La maintenance préventive a le coût de réparation le plus faible en raison d'un temps d'arrêt bien planifié, tandis que la maintenance prédictive peut réaliser un compromis entre le coût de réparation et le coût de prévention. Idéalement, la maintenance prédictive permet de réduire au maximum la fréquence de maintenance afin d'éviter une maintenance réactive non planifiée, sans encourir les coûts associés à une maintenance préventive trop importante.

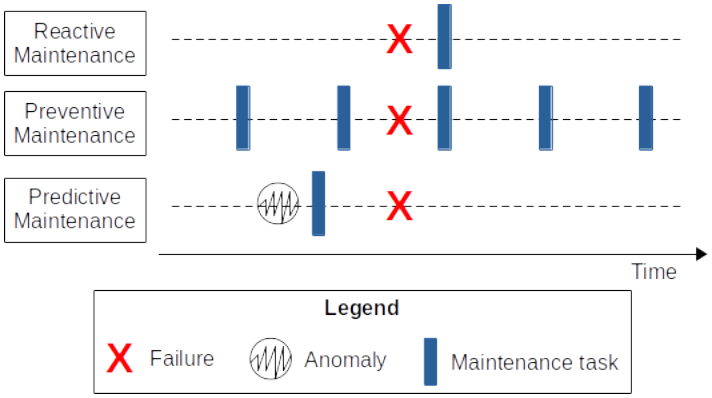

Figure 1.4 – Stratégies de maintenance dans l'industrie (RM - PM - PdM)

L'utilisation des technologies associées à l'Industrie 4.0 rend la maintenance prédictive plus efficace et plus flexible [CBL+18, Wan16]. C'est ce qu'on appelle la maintenance prédictive 4.0. Elle utilise une analyse avancée et en ligne des données collectées pour la détection précoce de l'apparition d'éventuelles défaillances de machines, et soutient les techniciens lors des interventions de maintenance en leur fournissant une aide à la décision. En d'autres termes, la maintenance prédictive 4.0 applique une surveillance en temps réel de l'état des machines, et des alertes sont données sur la base de règles

préétablies ou de niveaux critiques. Au cours des dernières décennies, de nombreux efforts de recherche ont été entrepris pour développer différents modèles de maintenance prédictive industrielle, en utilisant les éléments clés de l'Industrie 4.0.

Dans ce qui suit, nous décrivons les deux principales approches pour mettre en œuvre la maintenance prédictive : les approches fondées sur les données et les approches fondées sur la représentation des connaissances.

Approches basées sur les données pour la surveillance de l'état du système

Avec le développement des techniques liées aux grandes données et la disponibilité toujours croissante des données, la maintenance prédictive basée sur les données est devenue de plus en plus attrayante. Pour extraire des connaissances utiles et aider à prendre des décisions appropriées à partir d'une grande quantité de données, l'apprentissage automatique et les techniques d'apprentissage profond sont considérés comme des solutions puissantes. Les modèles obtenus en appliquant ces techniques fonctionnent comme une boîte noire qui apprend le comportement des machines physiques directement à partir de leurs données d'exploitation [JGZ17]. Ils ont la capacité d'apprendre des caractéristiques, de classer et de prédire les défauts. Dans le cadre d'une approche guidée par les données, les connaissances sur les machines sont extraites en interne des données de fonctionnement des machines, plutôt qu'en externe par des experts du domaine.

Les travaux ont apporté des contributions intéressantes dans le domaine de la détection et du diagnostic des défauts, mais la plupart d'entre elles sont spécifiques à des équipements précis. Ces travaux surveillent le comportement des différentes propriétés d'une machine particulière. Ils sont adaptés et efficaces lors de la consommation de flux de données pour détecter des modèles anormaux dans les valeurs des propriétés d'une machine (par exemple, la température, les vibrations, etc.). Cependant, ils présentent deux inconvénients : (i) la nécessité, à l'avance, d'une grande quantité de données annotées pour l'apprentissage du modèle ; et (ii) l'absence de modèle explicite pour expliquer les décisions. Cela rend difficile l'interprétation des données et complique également l'interopérabilité et la réutilisation des modèles.

Approches basées sur la connaissance pour la surveillance de l'état du système

De nombreux systèmes de surveillance des processus de fabrication et de diagnostic des défauts font appel à des connaissances d'experts a priori et à des processus de raisonnement déductif [PDZ10].

La connaissance experte est exploitée pour construire une ontologie qui décrit formellement les concepts et les relations qui existent dans le domaine industriel. Différentes ontologies ont été construites pour des tâches de surveillance telles que la maintenance prédictive et les systèmes d'évaluation en santé [NB18]. La modélisation basée sur les ontologies permet : (1) le partage des connaissances entre entités informatiques en ayant un ensemble commun de concepts, (2) l'inférence logique en exploitant divers mécanismes de raisonnement logique existants pour déduire des concepts de haut niveau à partir de données brutes, et (3) la réutilisation des connaissances en réutilisant des ontologies bien

définies de différents domaines. Les ontologies peuvent être utilisées pour représenter différents systèmes et dispositifs de machines, et peuvent être combinées avec divers algorithmes de raisonnement à base de règles existants pour réaliser la surveillance et le diagnostic des erreurs. Cela se fait sur la base de l'évaluation des données collectées en temps réel selon un ensemble de règles prédéterminées par des connaissances d'experts. En général, les règles sont exprimées sous la forme **IF** ⟨*antécédent*⟩ **THEN** ⟨*conséquence*⟩. De cette façon, la conséquence peut apporter des changements qui affectent le système qui est surveillé, si les conditions de l'antécédent sont remplies.

Les ontologies fournissent un moyen d'intégrer, de partager et de réutiliser les connaissances du domaine, mais d'autres méthodes de raisonnement doivent être intégrées aux ontologies pour parvenir à une maintenance prédictive. Le raisonnement basé sur des règles peut être utilisé pour réaliser le suivi et le diagnostic. Ces règles sont construites à partir de connaissances d'experts, ou sont extraites de l'analyse de grands ensembles de données, comme mentionné ci-dessus. Cependant, les deux approches ont des difficultés à traiter de nouvelles failles (inconnues des experts et non vues dans les données). Il est donc nécessaire d'acquérir des connaissances complètes pour construire un ensemble de règles fiables.

Nous résumons la couverture du domaine des ontologies existantes dans le tableau 1.1. Nous évaluons la couverture du domaine et les portées de ces modèles de connaissance en examinant si les concepts clés requis pour la surveillance de l'état du système dans l'industrie 4.0 existent et sont formellement décrits. Nous avons classé ces concepts clés en cinq sous-domaines : Fabrication, Contexte, Évolution dans le temps, IoT, et Surveillance et Diagnostic. Pour le sous-domaine Fabrication, les concepts clés sont le produit, le processus et la ressource. Pour le sous-domaine Contexte, les concepts clés sont l'identité, l'activité, le temps et le lieu. Pour le sous-domaine IoT, les concepts clés sont le capteur et l'observation. Pour le sous-domaine Monitoring et Diagnosis, les concepts clés sont situation et cause. Ces concepts sont les colonnes du tableau 1.1, et les modèles ontologiques sont les lignes. Si une coche est placée dans le tableau, alors le concept existe dans l'ontologie correspondante. Sinon, une croix est attribuée.

Table 1.1 – Comparaison entre les ontologies du domaine de la fabrication celles sur la modélisation du contexte.

	Manufacturing			**Context**				**Change over time**	**IoT**	**Monitoring and Diagnosis**	
Ontologies	**Product**	**Process**	**Resource**	**Identity**	**Activity**	**Time**	**Location**		**Sensor**	**Situation**	**Cause**
MASON	✓	✓	✓	✓	✓	✗	✗	✗	✗	✗	✗
ADACOR	✓	✓	✓	✓	✓	✗	✗	✗	✗	✗	✗
PSL	✗	✓	✓	✓	✓	✓	✗	✗	✗	✗	✗
MSDL	✗	✓	✓	✓	✓	✓	✓	✗	✗	✗	✗
SIMPM	✗	✓	✓	✓	✓	✗	✓	✗	✗	✗	✗
MaRCO	✗	✓	✓	✓	✓	✗	✓	✗	✗	✗	✗
ONTO-PDM	✓	✓	✓	✓	✓	✓	✓	✗	✗	✗	✗
P-PSO	✗	✓	✗	✓	✓	✓	✗	✗	✗	✗	✗
MCCO	✓	✓	✓	✓	✓	✓	✗	✗	✗	✗	✗
[CZX+16]	✓	✓	✓	✓	✓	✓	✓	✗	✗	✗	✗
OntoSTEP	✓	✗	✗	✓	✗	✗	✗	✗	✗	✗	✗
CoBrA-Ont	✗	✗	✗	✓	✓	✓	✓	✗	✗	✗	✗
SOUPA	✗	✗	✗	✓	✓	✓	✓	✗	✗	✗	✗
CONON	✗	✗	✗	✓	✓	✗	✓	✗	✗	✓	✓
SAWA	✗	✗	✗	✓	✓	✗	✗	✗	✗	✓	✓
Situation Ont.	✗	✗	✗	✓	✓	✗	✗	✗	✗	✓	✗

Après avoir examiné les modèles ontologiques existants, aucun d'entre eux ne fournit une représentation satisfaisante de la connaissance des cinq sous-domaines requis. Certains de ces modèles de connaissance se concentrent sur un domaine restreint, comme la planification des ressources de fabrication et les bons de travail, et ils ne formalisent pas les concepts liés au contexte, comme l'activité et le lieu. En outre, aucune des ontologies existantes ne permet de représenter des connaissances qui évoluent dans le temps.

Pour effectuer une surveillance intelligente de l'état du système, la base de connaissances doit contenir non seulement des connaissances interprétables par les machines pour caractériser les entités ou les processus de fabrication qui sont surveillés, mais aussi les connaissances sur les situations anormales qui sont associées aux défaillances. Ceci motive le développement d'un modèle ontologique plus expressif et plus complet qui fournit une représentation riche des connaissances du domaine dans les domaines de la fabrication en considérant la notion de contexte.

De plus, les modèles ontologiques des systèmes intelligents sont statiques. Comme décrit dans l'introduction, les machines exécutent des processus de fabrication dans différents contextes et ces contextes changent au fil du temps. Selon le contexte dans lequel un processus de fabrication est exécuté, les règles qui gèrent le processus peuvent changer. Afin de représenter ces contextes et le fait qu'une machine exécute un processus dans ces contextes, le modèle sémantique doit évoluer dans le temps pour représenter cette connaissance changeante.

1.3 Contributions de cette thèse

Les principaux éléments de notre proposition sont présentés dans la Figure 1.5 en utilisant la notation classique du diagramme de flux[6]. Il s'agit (1) de la composante de surveillance (*Monitoring*), (2) de la composante de diagnostique (*Diagnosis*) et (3) de la composante de décision (*Decision making*). Chacune des composantes fonctionne comme un système intelligent permettant de résoudre des problèmes complexes par le raisonnement, comme le fait un expert.

Ils s'appuient tous sur un modèle formel pour enrichir sémantiquement la représentation et le traitement des données. Les sections suivantes expliquent chacun de ces éléments.

1.3.1 Modèle sémantique pour la modélisation du contexte dans l'Industrie 4.0

Le **modèle sémantique** pour l'Industrie 4.0 est au fondement de notre approche. Il s'agit d'une représentation virtuelle de l'usine réelle, comme l'illustre la figure 1.6. Il représente les éléments de l'usine, tels que les machines, les processus et les capteurs, en mettant particulièrement l'accent sur la modélisation du contexte de fonctionnement de ces éléments. Des situations pertinentes représentant des comportements anormaux ainsi que

[6]https://en.wikipedia.org/wiki/Data-flow_diagram

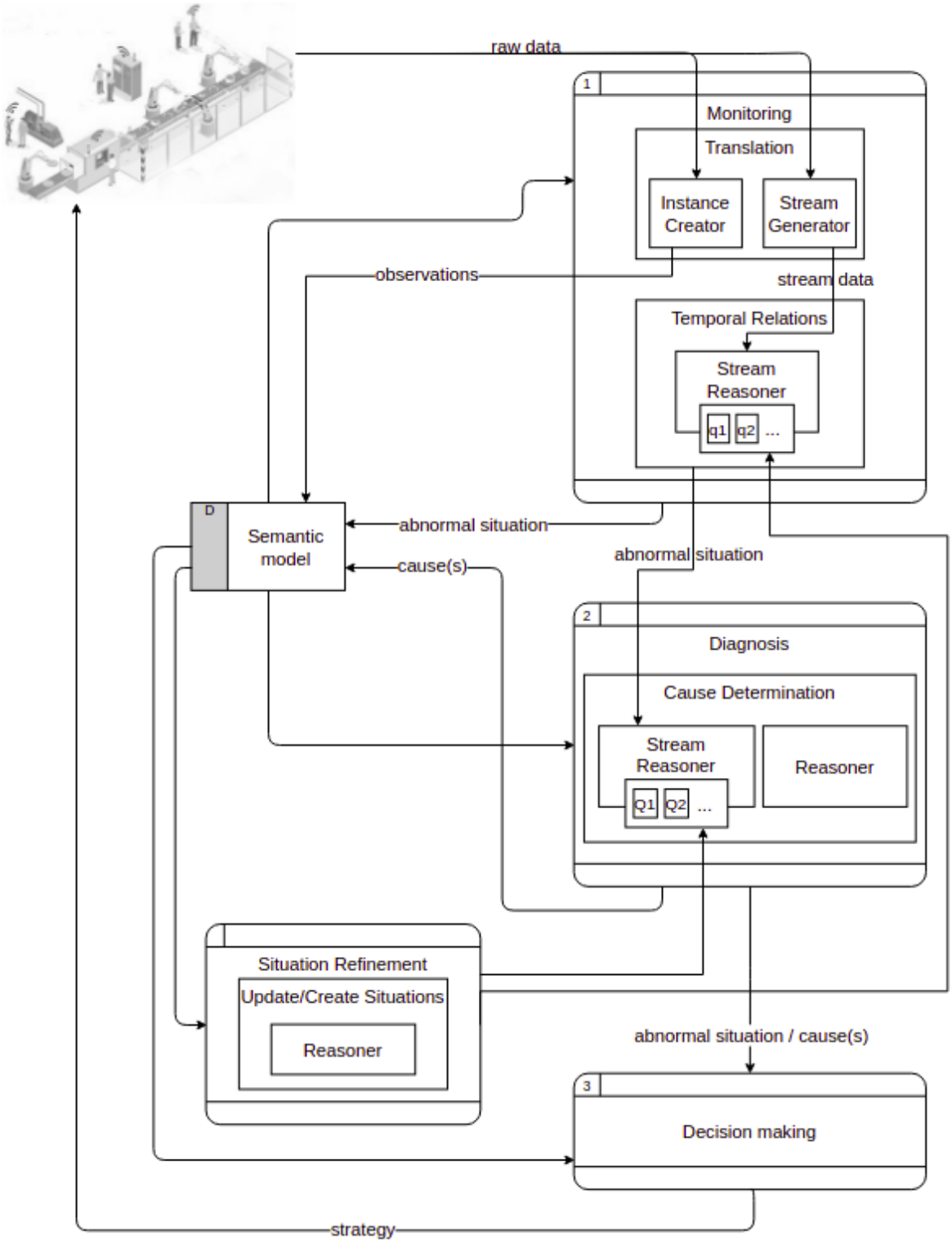

Figure 1.5 – Principaux composants et flux de travail du cadre proposé.

des connaissances d'experts et des règles de gestion des processus sont également représentées dans le modèle.

Dans une usine réelle, les ressources (telles que les machines et les lignes de production) exécutent leurs tâches au fil du temps et dans différentes situations. Le modèle représente dans quelle(s) situation(s) la ressource exécute ses tâches pour permettre une prise de décision plus éclairée, puisque les actions à prendre et les règles qui gèrent les

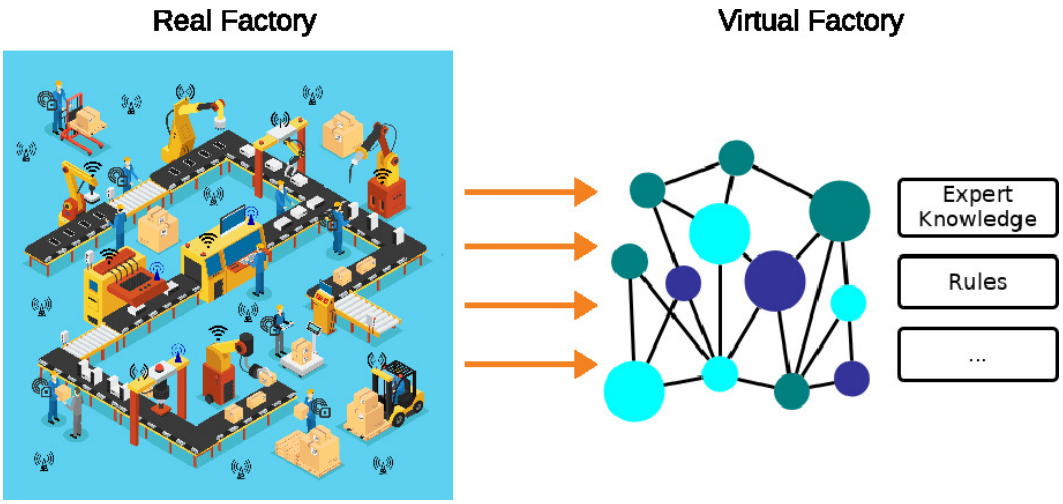

Figure 1.6 – Représentation virtuelle d'une usine réelle.

processus de fabrication peuvent varier en fonction de la situation identifiée. En d'autres termes, le modèle sémantique doit évoluer pour pouvoir représenter la(les) situation(s) dans laquelle (lesquelles) une certaine ressource se trouve pendant l'exécution de ses tâches. Cette évolution est illustrée graphiquement dans la Figure 1.7.

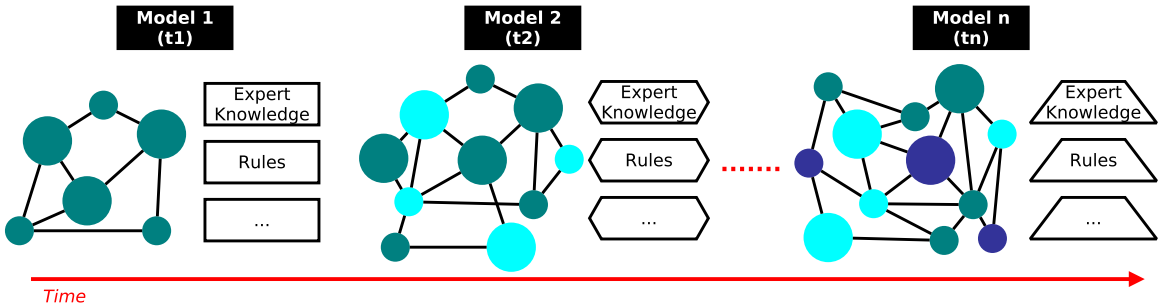

Figure 1.7 – Évolution du modèle sémantique dans le temps.

1.3.2 Raisonnement sur les flux de données pour la détection et le diagnostic des situations anormales

Certaines situations qui peuvent conduire à des défaillances de machines peuvent être détectées en interprétant les observations dans leur contexte. Par exemple, si une observation a une valeur anormale, c'est peut-être parce qu'un autre paramètre a également des valeurs anormales. Considérons le cas où la température d'une machine et la température d'un de ses composants sont surveillées. On sait qu'une augmentation de la température de la machine peut être due à une augmentation de la température de son composant, ou vice versa. Cela permet d'exploiter les connaissances des experts sur les relations entre les valeurs de certains paramètres des machines, des processus et de leur contexte pour interpréter les observations. Grâce à la détection précoce des situations, le programme de maintenance peut être adapté ou des mesures peuvent être prises pour éviter des arrêts imprévus de la production.

Les principaux composants du cadre que nous proposons sont présentées dans la figure 1.5. Les modules permettant de détecter les situations en temps réel sont la traduction (*Translation*) et les relations temporelles (*Temporal relations*). Ces deux modules font partie de la surveillance. Le composant de diagnostic ne comporte qu'un seul module appelé détermination des causes (*Cause Determination*). Les composants de surveillance et diagnostic effectuent des tâches différentes mais leurs tâches sont étroitement liées. En outre, la figure 1.5 montre également le composant de raffinement des situations (*Situation Refinement* et le composant de prise de décision (*Decision making*. Le premier contient le module *Update/Create situation* qui vise à affiner les situations déjà définies ou à en définir de nouvelles. Le composant Decision making ne fait pas partie de la tâche de détection des situations ou de détermination des causes possibles, mais il exploite ces informations pour soutenir les tâches de prise de décision.

Le composant Monitoring

Les modules impliqués dans la détection des situations sont *Translation* et *Temporal Relations*. Les deux sous-sections suivantes expliquent et décrivent l'interaction entre eux ainsi qu'avec le modèle ontologique.

Le module *Translation*

Ce module est responsable (i) de la conversion des données acquises par les capteurs en flux RDF, et (ii) de leur insertion en tant qu'instances dans l'ontologie. Ces deux tâches sont effectuées respectivement par les sous-modules *Stream Generator* et *Instance creator*.

Le sous-module *Stream Generator* effectue l'enrichissement sémantique des données acquises, en utilisant les concepts et les relations entre eux tels que définis dans le modèle ontologique. Cela permet au module de diffuser des flux de données enrichis sémantiquement qui sont ensuite consommés par le *Stream Reasoner*. Les flux de sortie sont des flux RDF. Un flux RDF est une séquence ordonnée de paires, où chaque paire est constituée d'un triplet RDF et de son horodatage t, (<Subject,Predicate,Object>,t).

Le sous-module *Instance creator* crée des instances à partir des données reçues et les insère dans l'ontologie, c'est-à-dire qu'il est chargé de remplir le modèle ontologique avec les observations et leurs métadonnées correspondantes, telles que le(s) capteur(s) qui a(ont) fait l'observation, la propriété observée et l'horodatage.

Le module *Temporal Relations*

Une fois que les données des sources de données distribuées et hétérogènes sont disponibles dans une représentation homogène, contextualisée et ordonnée dans le temps, les flux peuvent être explorés pour générer de nouvelles connaissances.

Un ensemble de requêtes, qui combine les connaissances de base extraites de l'ontologie et certaines parties pertinentes des flux, est enregistré et exécuté par le raisonneur sur les flux de données. Ces requêtes représentent des situations particulières à identifier et

elles incluent principalement des dépendances temporelles entre les observations (qui peuvent être des situations normales ou des anomalies).

Lorsque ce module détecte des situations, celles-ci peuvent être converties en flux RDF et être renvoyées en sortie, c'est-à-dire que ce module produit des flux de situations en sortie. Cette sortie alimente un autre raisonneur sur flux de données dans le module *Cause determination*. De cette façon, le module *Temporal Relations* lui-même peut être considéré comme un capteur avancé capable de produire des données de haut niveau. De plus, les situations détectées sont stockées dans le modèle ontologique indiquant également les ressources impliquées dans cette situation.

Le composant Diagnosis

Ce composant ne dispose que d'un seul module appelé *Cause Determination* pour déterminer les causes possibles qui ont provoqué une situation anormale.

Le module *Cause Determination*

L'objectif du module *Cause Determination* est d'identifier les causes possibles qui ont généré une situation détectée par le module *Temporal Relation*. Pour cela, deux éléments sont utilisés séparément : un raisonneur sur flux de données *Stream Reasoner*, et un raisonneur classique *Reasoner*.

Le raisonnement sur flux de données est plus adapté aux données hautement dynamiques que les approches de raisonnement classiques. Ainsi, dans le cas où les causes doivent être déterminées en temps réel, le module *Stream reasoner* est utilisé pour identifier les causes. Les associations entre les situations et leurs causes possibles sont stockées dans le modèle ontologique et sont exploitées par ce module pour les restituer. Cependant, il est possible que certaines situations n'aient pas de causes identifiées dans un scénario réel, auquel cas le système notifie que les causes sont inconnues.

Par conséquent, afin de déterminer les causes possibles d'une situation, des approches de raisonnement classiques peuvent être utilisées. Elles fournissent d'autres inférences qui peuvent aider à déterminer les causes d'une situation. Des requêtes plus complexes peuvent être effectuées sur le modèle ontologique afin d'extraire des informations utiles pour la détermination des causes. Par exemple, il peut être nécessaire de considérer l'hypothèse du monde ouvert, qui stipule que l'absence d'une déclaration ne peut pas être utilisée seule pour déduire que la déclaration est fausse. Dans ce cas, le *Reasoner* peut également être utilisé sur l'ontologie pour déduire les causes, si l'exigence de temps réel n'est pas nécessaire. Cette dernière option présente certains avantages par rapport à la précédente. Si la cause est identifiée ultérieurement, elle est ajoutée comme instance au modèle ontologique et liée à la situation pour une utilisation future.

Dans les deux cas, la composante Diagnosis fournit à la composante Decision making la situation identifiée ainsi que les causes possibles déduites par le(s) raisonneur(s). L'association entre la situation et les causes est également ajoutée au modèle ontologique.

Le composant Abnormal Situation Refinement

Comme mentionné dans la sous-section précédente, le module *Translation*, en plus de générer les flux de données, alimente l'ontologie avec des données sémantiquement enrichies. L'objectif principal du module *Update/Create situations* est d'exploiter les données historiques stockées en tant qu'instances dans le modèle ontologique, d'affiner les situations déjà définies et, éventuellement, d'en trouver de nouvelles. Ceci peut être accompli (i) en raisonnant sur l'ontologie pour en tirer des connaissances implicites, et (ii) en interagissant avec l'ontologie en émettant des requêtes ponctuelles. Ces requêtes peuvent être exécutées pour vérifier la fréquence à laquelle une situation se produit ou pour vérifier la tendance des valeurs de certaines propriétés afin d'affiner les contraintes qui leur sont imposées et ainsi affiner les situations associées à ces contraintes.

Ce module utilise un raisonnement classique et non un raisonnement sur flux de données car le raisonnement est appliqué aux données statiques stockées dans le modèle ontologique et non aux flux de données.

Le composant Decision Making

Compte tenu des situations et de leurs causes, il est possible d'appuyer les tâches de prise de décision pour déterminer les actions à lancer pour corriger le comportement des machines et éviter les pannes. Les actions déclenchables sont de natures diverses. Par exemple, certaines tâches de maintenance peuvent être lancées à distance et effectuées par les machines elles-mêmes ou le système intelligent peut émettre une alerte pour avertir l'opérateur le plus proche de la machine afin qu'il l'inspecte, si une action humaine est nécessaire.

Chaque application peut effectuer un raisonnement et un traitement plus avancés sur les données reçues afin de déterminer les stratégies d'adaptation nécessaires pour améliorer le comportement du système industriel. Ces stratégies peuvent inclure des modifications des paramètres de fonctionnement d'une machine ou le lancement d'une tâche de maintenance.

1.3.3 Hiérarchies de situation pour soutenir la prise de décision

Une fois qu'une situation anormale et ses causes ont été détectées, la composante Decision making est chargée de déterminer et d'appliquer les stratégies d'adaptation nécessaires pour améliorer le comportement du système de production. Ces stratégies peuvent inclure des modifications des paramètres de fonctionnement d'une machine ou le lancement d'une tâche de maintenance.

Le choix de la (des) action(s) à lancer est effectué par le composant Decision making. Ce composant utilise également le modèle sémantique pour déterminer les actions qu'il est possible d'exécuter pour améliorer les conditions de fonctionnement.

Comme la détection de situations anormales peut déclencher des actions pour adapter le comportement du processus, cette modification du comportement du processus peut conduire à la génération de nouvelles situations.

Ces situations anormales peuvent avoir différents niveaux de gravité, et peuvent être imbriquées de différentes manières. Pour comprendre et gérer les relations entre les situations, une approche efficace pour les organiser est nécessaire. C'est pourquoi il est proposé une méthode permettant de construire un treillis pour ordonner les situations possibles associées à un processus de fabrication, en fonction des contraintes sur lesquelles elles reposent. Ce treillis représente une feuille de route de toutes les situations qui peuvent être atteintes à partir d'une situation donnée. Cette feuille de route aide à la prise de décision, en permettant l'identification des actions qui peuvent être prises pour corriger l'anomalie. Par conséquent, la composante Decision making utilise à la fois le modèle sémantique et le treillis pour déterminer les actions à déclencher pour corriger la situation anormale.

1.3.4 Mise en œuvre du cadre proposé

Dans les sections précédentes, les composants du cadre proposé ont été introduits. Un modèle ontologique est développé pour représenter le domaine industriel et il est utilisé comme base d'un système intelligent. Une approche sémantique hybride est proposée pour automatiser la détection de situations anormales, basée sur l'utilisation combinée du raisonnement classique et du raisonnement sur flux de données. Une approche fondée sur un treillis est présentée pour représenter une feuille de route de toutes les situations qui peuvent conduire à des défaillances potentielles, afin de soutenir les tâches de prise de décision.

Afin d'appliquer notre approche, nous avons développé un prototype de logiciel. Le logiciel utilise des approches déductives, des ontologies de domaine et des raisonnements ontologiques, ainsi que des raisonnements sur flux de données pour analyser les données industrielles et détecter les situations anormales qui peuvent conduire à des défaillances.

Les technologies et les outils utilisés pour le développement du cadre proposé sont présentés ainsi que la mise en œuvre des fonctionnalités de base de celui-ci. Une preuve de concept est également présentée à travers une étude de cas illustrative du domaine industriel. L'objectif est de montrer l'interaction entre les différents composants et comment ils modifient le modèle sémantique, le faisant évoluer pour la représentation de ce qui se passe dans l'usine réelle. Les résultats obtenus par notre preuve de concept à l'aide de données simulées sont encourageants. Malheureusement, il n'a pas été possible de valider l'ensemble de l'approche sur des données industrielles réelles, même hors ligne. Dans l'idéal, il serait également intéressant de tester notre proposition dans une chaîne de production réelle. Cela permettrait de vérifier si les décisions prises sur la base de l'utilisation de notre proposition améliorent effectivement son efficacité et sa fiabilité.

1.4 Conclusions et Travaux Futurs

Dans cette section, nous présentons un résumé de ces travaux de thèse avec leurs contributions correspondantes. Ensuite, quelques perspectives de travaux futurs sont également présentées.

Conclusions

Comme évoqué au long de ce manuscrit, la gestion de données industrielles hétérogènes est une tâche difficile dans le cadre de la surveillance de l'état du système. De plus, comme la structure et le comportement des systèmes de production deviennent de plus en plus complexes, le volume de données augmente considérablement. C'est pourquoi les entreprises industrielles recherchent des solutions pour traiter efficacement ces données hétérogènes et effectuer des tâches de surveillance et de diagnostic de manière intelligente.

Dans ce cadre, il est nécessaire de disposer de modèles bien définis pour gérer les données hétérogènes et les exploiter, ainsi que de connaissances expertes. Pour développer un tel modèle, la connaissance du domaine de la fabrication doit être structurée de manière à être interprétable par les machines et utilisable par le système de surveillance. En outre, comme le domaine de l'industrie devient de plus en plus axé sur la connaissance, une représentation uniforme des ressources physiques et des capacités de raisonnement est nécessaire pour automatiser les processus décisionnels. Ces processus de décision comprennent la détection des situations anormales et la détermination des causes, la programmation de la maintenance et l'adaptation des processus. Pour atteindre ces objectifs, les technologies sémantiques ont montré des résultats prometteurs en formalisant les connaissances sur les tâches de surveillance de l'état dans plusieurs applications industrielles.

Les ontologies existantes pour représenter le domaine de la production ne permettent pas de représenter des concepts et des relations qui évoluent dans le temps, comme les différents contextes dans lesquels une ressource de fabrication peut accomplir ses tâches. Ces modèles ontologiques existants conviennent pour des informations qui ne changent pas ou très peu dans le temps. Cependant, le domaine de l'industrie traite de données très dynamiques et un système de production devrait être capable de s'adapter à des situations changeantes. Pour cela, le modèle ontologique doit pouvoir faire face à l'évolution des connaissances. Différents ensembles d'actions peuvent être sélectionnés pour corriger le comportement anormal en fonction du contexte physique spécifique de la machine en fonctionnement. Cela permet de garantir que la production continue sans qu'il soit nécessaire de l'arrêter si la situation anormale n'est pas grave.

Pour répondre à ces problématiques, un nouveau cadre sémantique est proposé pour aborder l'évolution des modèles sémantiques dans l'Industrie 4.0. Le cadre proposé permet d'automatiser et de faciliter la surveillance et le diagnostic de l'état du système, et de soutenir la prise de décision dans le domaine de l'industrie.

À cette fin, nous proposons tout d'abord un modèle ontologique qui représente les ressources et les processus qui font partie d'une usine, en mettant particulièrement l'accent sur la modélisation du contexte de ces ressources et processus. Les situations pertinentes qui combinent les observations des capteurs avec la connaissance du domaine sont également représentées dans le modèle. Le cadre proposé enrichit les données collectées par les capteurs avec des informations contextuelles en utilisant le modèle ontologique et utilise le raisonnement sur flux de données pour permettre la détection de situations

anormales en temps réel. En outre, il utilise également des approches de raisonnement classiques pour compenser la non-détection éventuelle des causes par la méthode du raisonnement sur flux de données. Grâce à la détection de ces situations et de leurs causes, des décisions appropriées peuvent être prises pour éviter l'interruption du processus surveillé. Enfin, pour soutenir la prise de décision, le cadre proposé fournit une hiérarchie de situations qui représente une feuille de route des situations, souhaitable ou non, auxquelles on peut parvenir à partir d'une situation donnée. Cela permet d'identifier les actions à prendre pour corriger le comportement anormal, en évitant ou en minimisant de cette façon l'interruption des processus de fabrication.

Les contributions de cette thèse sont résumées ci-dessous.

Dans le Chapitre 4, une vue d'ensemble du cadre proposé pour traiter l'évolution des modèles sémantiques dans l'Industrie 4.0 est introduit. Les composantes de ce cadre sont les suivantes : (1) la composante Monitoring ; (2) la composante Diagnosis ; et (3) la composante Decision making. Chacune de ces composantes fonctionne comme un expert. Elles utilisent toutes le modèle sémantique pour remplir leurs fonctions et le font évoluer en introduisant les changements correspondants. De cette façon, les tâches de chaque composant sont exécutées en tenant compte du modèle sémantique mis à jour qui est une représentation virtuelle de ce qui se passe dans l'usine réelle.

Dans le Chapitre 5, une ontologie pour la modélisation du contexte dans le domaine industriel est proposée. Le modèle ontologique répond aux exigences de l'Industrie 4.0, en mettant l'accent sur les relations temporelles et spatiales entre les ressources et les processus afin de représenter des situations particulières d'intérêt dans un scénario industriel. En outre, notre modèle ontologique permet de saisir les changements dynamiques et l'évolution des concepts dans le temps. L'ontologie proposée est générique et extensible et son architecture modulaire permet la description des capacités des ressources de fabrication à différents niveaux de granularité.

Dans le Chapitre 6, les composants de notre cadre proposé sont détaillés: les composants Monitoring, Diagnosis, Situation Refinement et Decision making. L'utilisation de méthodes de raisonnement sur les flux de données permet l'intégration de données provenant de différentes sources, avec différentes significations sous-jacentes, différentes résolutions temporelles ainsi que le traitement de ces données en temps réel. Les données collectées par les capteurs sont enrichies d'informations contextuelles pour permettre la détection de la situation en temps réel. En outre, notre proposition utilise également des approches de raisonnement classique pour compléter la détermination des causes qui n'ont pas pu être obtenues par la première méthode.

Dans le Chapitre 7, une approche qui utilise la théorie des treillis pour représenter une hiérarchie de situations pouvant conduire à des défaillances potentielles est présentée. La hiérarchie des situations est construite automatiquement en tenant compte des contraintes sur lesquelles elles reposent. La hiérarchie permet de représenter des situations qui se produisent partiellement, c'est-à-dire lorsque seules certaines de leurs contraintes sont satisfaites. L'approche proposée peut être appliquée à d'autres domaines d'application, où une surveillance en temps réel est nécessaire.

Enfin, le Chapitre 8 présente les technologies et les outils utilisés pour le développe-

ment du cadre proposé ainsi que la mise en œuvre des fonctionnalités de base de celui-ci. Une étude de cas illustrative du domaine industriel est présentée pour démontrer l'application du cadre proposé. Malheureusement, il n'a pas été possible de valider l'ensemble de l'approche sur des données industrielles réelles. Cependant, les résultats obtenus par notre preuve de concept en utilisant des données simulées sont encourageants.

Travaux Futurs

Les contributions présentées dans cette thèse permettent d'identifier plusieurs perspectives de recherche.

Les travaux futurs peuvent être orientés vers l'enrichissement du modèle sémantique proposé. De nouveaux types de relations peuvent être explorés en dehors des relations spatiales et temporelles abordées dans cette thèse, afin de permettre la représentation d'informations contextuelles plus riches. Par exemple, la représentation de la relation selon laquelle deux lignes de production exécutent le même processus pourrait être utilisée pour détourner la production d'une ligne de production à l'autre si la première se comporte anormalement. Une autre possibilité peut être la représentation des relations entre les lignes de production et les bons de travail, de sorte que, selon si la ligne de production est proche de la fin d'un bon de travail et que la date limite approche, la décision de poursuivre la production jusqu'à ce que la commande soit terminée ou d'arrêter la ligne de production puisse être prise.

Comme le domaine de la fabrication est très dynamique, la manière de traiter les flux de données hétérogènes en temps réel est une préoccupation cruciale. Afin de tester les questions d'extensibilité et de complexité pour détecter les situations en temps réel, il faut explorer une étude de cas plus large avec des situations plus complexes, impliquant davantage d'observations de propriétés qui sont liées dans le temps et dans l'espace. Différentes implémentations de moteurs de raisonnement de flux devraient être évaluées en termes d'efficacité et d'extensibilité, en tenant compte également de la taille du modèle sémantique.

Une autre direction intéressante à explorer est l'identification de modèles de causalité qui pourraient permettre la définition de requêtes génériques pour certains types d'anomalies. Ces requêtes génériques pourraient être réutilisées dans différents cas, et effectuer des opérations complexes parmi les valeurs détectées.

En ce qui concerne l'affinement et l'identification de nouvelles situations, il peut être intéressant d'appliquer des méthodes d'apprentissage automatique pour exploiter les données stockées dans le modèle sémantique. Les requêtes étant exécutées pendant certaines fenêtres temporelles à des intervalles de temps donnés, l'application des méthodes d'apprentissage automatique peut aider à déterminer la durée des fenêtres temporelles dans lesquelles une requête doit être exécutée. De plus, il serait également intéressant d'adapter dynamiquement la durée des fenêtres temporelles.

Il existe également des perspectives associées à la construction de la hiérarchie des situations à partir de données industrielles réelles. Tout d'abord, cette construction soulève également des questions d'extensibilité et de complexité pour construire le treillis. Par

conséquent, des tests doivent être effectués avec différentes variations de l'algorithme 1 pour évaluer leur efficacité relative. En outre, une autre perspective consiste à étudier comment ajouter de nouvelles situations ou des situations affinées sans avoir à reconstruire complètement la hiérarchie.

Enfin, une étude plus exhaustive et plus détaillée peut être réalisée sur les dépendances entre les contraintes pour l'identification de nouvelles situations ou le raffinement de celles qui existent déjà. De cette manière, il est possible d'obtenir différentes hiérarchies de situations en fonction de la relation $\mathscr{T}$ choisie, qui peuvent être utilisées seules ou combinées avec d'autres, permettant ainsi différentes stratégies de prise de décision.

Introduction

Context

Historically, industrial revolutions have changed the way society lives and works. The first industrial revolution (18th to 19th century) used water and steam power to mechanize production [DD79]. The second industrial revolution (1870 to 1914) used electric power to enable mass production of goods [Mok98]. The third industrial revolution (1980 to 2010) used electronics and information technology to automate production [Rif11]. Now a fourth industrial revolution, also known as Industry 4.0, is building on the third one. Industry 4.0 is characterized by a fusion of technologies that is blurring the lines between the physical and digital spheres. Among these technologies there are Artificial Intelligence (AI), Robotics, the Internet of Things (IoT) and Cyber Physical Systems (CPS). Figure 1.8 shows these industrial revolutions and the key technologies associated with each of them.

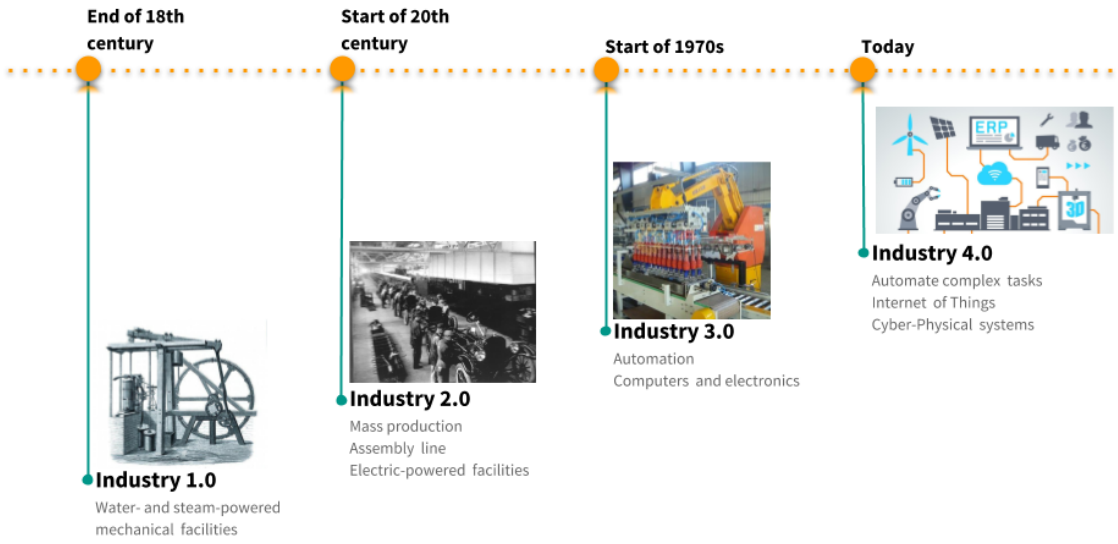

Figure 1.8 – History of industrial revolution.[7]

The main objective of Industry 4.0 is, then, to improve the production and associated services through the use of the innovative technologies just mentioned [SMB+17, WWLZ16]. To ensure high productivity, availability, and efficiency of manufacturing pro-

[7] adapted from: https://www.i-scoop.eu/industry-4-0/

cesses, the detection of abnormal conditions of production lines is a crucial issue for manufacturers [Has11]. In order to tackle this issue, factories rely on condition monitoring. This is the task of monitoring all the equipment involved in a manufacturing process for early detection of undesirable behaviors or situations that could lead to anomalies, which affect its performance, energy-use or reliability. Through the early detection of these situations, proactive decisions can be made to avoid production downtime. These decisions can imply, for example, changing the process parameters to adapt the processes behavior.

An important characteristic for industrial production in Industry 4.0 is that physical items such as sensors, actuators, machines and enterprise assets are connected to each other and to the Internet. In this environment, devices and sensors generate an increasing amount of data that can be used for effective equipment condition monitoring. In order to detect abnormal situations, the data collected by the sensors must be interpreted. This is a challenging task as it requires the integration and processing of heterogeneous data coming from different sources, with different temporal resolutions and different underlying meanings. Moreover, a key point to consider is that the monitoring of industrial processes should depend not only on their internal state and on user interactions but also on the context of their execution. Making industrial processes context-aware allows to provide added-value information to improve their performance. Context is any information that can be used to characterize the situation of an entity [DA00]. It is usually considered as a mixture of geospatial data, environmental sensor inputs, service descriptions, among others. In manufacturing, a dynamic context model should not only take into account the context of tools, machines, parts and products, but also the planning of the manufacturing processes, the specification of resources and the control system configuration. Context data is subject to constant change and can be highly heterogeneous.

Semantic Web technologies have proved their efficiency to deal with this data integration issue. The Semantic Web is an extension of the World Wide Web that combines knowledge engineering and AI methods to represent, integrate, and reason upon data and knowledge through ontologies and rules. In computer science, an ontology is considered as "an explicit specification of a conceptualization for a domain of interest" [Gru93]. Ontologies emerge as a pertinent method to represent knowledge of any kind (in particular, manufacturing knowledge) in a machine-interpretable way. The semantic models built in this way provide a virtual representation of the resources of a manufacturing plant as well as of the relevant situations associated to them. Furthermore, reasoning on ontologies through its rule-based extensions allows transforming raw observations collected by sensors into higher-level abstractions, such as situations of interest, that are meaningful for humans and provide a better understanding about the physical world to support decision making tasks. For example, observations from sensors can be used to optimize power consumption in a production line to avoid failures. This data transformation process is illustrated using the well known "knowledge hierarchy" [Row07] shown in Figure 1.9.

Current solutions for reasoning on ontologies were traditionally developed for static or slow changing data. The highly dynamic nature of data in the industrial domain introduces new issues. To tackle these, a number of recent works propose to unify reasoning and stream processing, giving rise to the research field of stream reasoning. Stream rea-

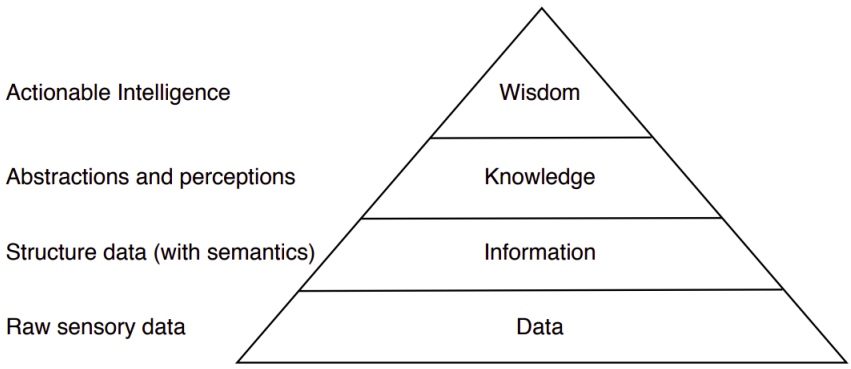

Figure 1.9 – Knowledge Hierarchy (taken from [Row07])

soning supports decision systems based on the continuous processing of data streams together with rich background knowledge [SCDV+19].

Under these circumstances, the semantic model itself must evolve in order to represent in which situation(s) the resource(s) is during the execution of its tasks to support decision-making.

Therefore, this thesis investigates the use of tractable Knowledge Representation & Reasoning (KRR) methods, in particular Semantic Web technologies, for building an evolving semantic model that represents the industrial domain, making emphasis on context modeling to provide the notion of situation.

Motivation

As mentioned in the previous section, semantic models are useful for the representation of knowledge about a particular domain, in this case Industry 4.0. However, these semantic models are rather static, meaning that they do not meet the dynamics of manufacturing processes. For the model to represent in which situations the processes are executed, it is necessary that the semantic model evolves in order to represent what happens in the real world.

The semantic model can evolve due to different reasons. The first possibility is related to a change in the structure of the model itself, *i.e.* addition/removal of concepts and relations. This type of change is studied in the field of ontology evolution [SMMS02, Sto04, NK04, PDTC07] and is not addressed in this thesis. The second possibility is related to the addition of instances to concepts already defined and to the addition of relations over existing instances. One example can be the addition of a physical resource to the factory that would be reflected as a new instance of the corresponding resource concept in the semantic model. Another example can be the addition of a detected situation and the fact that a certain resource is involved in that situation (meaning a new link between two instances through an existing relation).

Situations of interest in the industrial domain depend on sensor data and domain knowledge. Besides the static information about industrial entities, dynamic information, such as the way the industrial processes are executed, has to be described. Therefore, the

data collected from the sensors deployed in a factory needs to be explicitly represented to describe the process state. This is the first requirement for the semantic model to be build.

Another requirement is the representation of temporal relations among processes and of spatial relations between entities and locations in the factory. These relations are used to identify different situations of interest. This implies handling information that evolves in time, such as the changes of situation(s) a machine can go through or the changes in the values of a machine parameter according to the different decisions made. Consequently, the semantic model must provide enough expressiveness to capture high level knowledge, such as situations, from time annotated data coming from sensors.

More specifically, the requirements that the semantic model must meet are:

Data Integration. The integration of information that comes from different sources is a key requirement. Moreover, the integration of data coming from sensors (data streams) with background knowledge that describes the application domain is also required (*e.g.*, the streaming data collected from a machine is combined with background knowledge about different constraints that indicate abnormal behavior).

Time representation. Time plays a central role in this thesis. This demands for a suitable representation of time, where data can be annotated with their time of occurrence and validity. Beside processing streaming data, it is also required to manage historical data, *i.e.* time annotated information about previous states of the resource under analysis (*e.g.*, the history of temperature readings from a certain machine). This can be used, for example, to identify trends, to extract statistics, or to compare previous and current information to identify new types of anomalies or situations.

Efficiency. Regarding situation detection, processing efficiency is related to the ability to cope with large amount of data being collected to timely generate new results (e.g. abnormal situations need to be promptly detected for enabling counter actions).

High level decision support. Once a situation that may lead to failures is detected, an action needs to be undertaken to counter-act it. Actions change the state of the system and lead to other situations. In order to help in decision making and choose the most appropriate action, it is necessary to understand the relationships among situations.

These requirements motivate the proposal of a novel framework to deal with the evolution of semantic models in Industry 4.0. The proposed framework uses semantic technologies to represent the manufacturing domain with special emphasis on modeling the context of the resources involved in a factory. The proposed framework aims at detecting situations that can lead manufacturing processes in Industry 4.0 to failures and their possible causes. Through the detection of these situations and their causes, appropriate decisions can be made to avoid the interruption of manufacturing processes.

Contributions

A framework to address the evolution of semantic models in Industry 4.0 is proposed in this thesis. An overview of the framework is shown in Figure 1.10, the blue-dotted rectan-

gles represent the main contributions of this thesis that are described below.

The first contribution is related to the development of the semantic model; and the other ones are related to the management of the evolution of the semantic model.

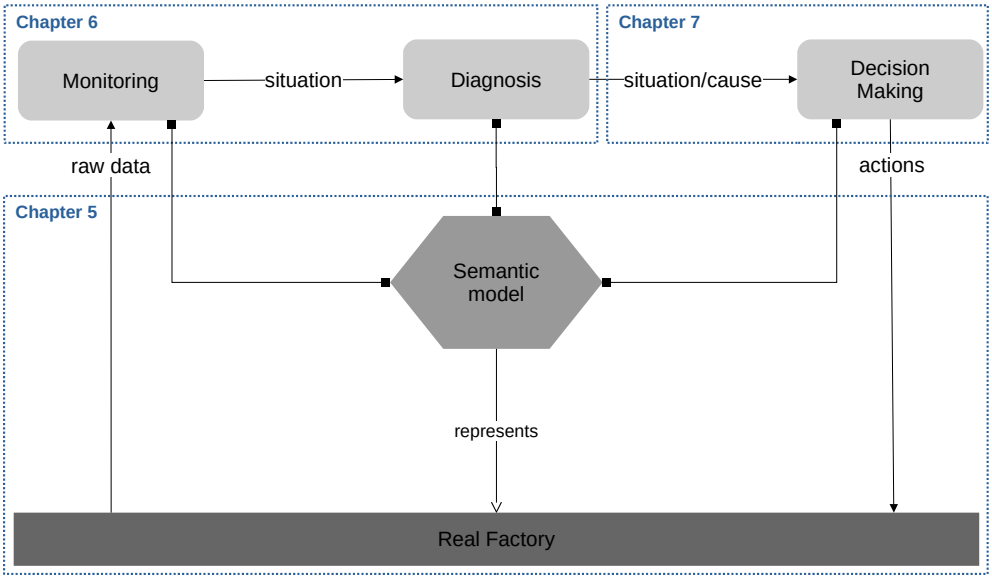

Figure 1.10 – Proposed framework overview.

- An ontological knowledge model for the manufacturing domain is proposed to provide a declarative, abstract representation of the resources, processes, sensors and the relationships among them. Furthermore, the model is strongly oriented towards modelling the context of the resources involved in a manufacturing process. Relevant situations that combine sensor observations with domain knowledge are also represented in it.

- An approach that uses stream reasoning to detect relevant situations that lead to potential failures is introduced. This approach enriches data collected from sensors with contextual information using the proposed semantic model. The use of stream reasoning facilitates the integration of data from different data sources, different temporal resolutions as well as the processing of these data in real time. This allows to identify situations from low-level context and sensor data streams combined with background knowledge. Furthermore, our proposal uses classical reasoning approaches for determining the possible causes that generate the situation when stream reasoning results are not sufficient.

- Detected situations can trigger actions to adapt the process behavior, and in turn, this change in behavior can lead to the generation of new contexts leading to new situations. To organize the situations based on the constraints they rely on, the building of a lattice is proposed. This lattice represents a road-map of all the situations that can be reached from a given one, desirable or undesirable. This helps

in decision support, by allowing the identification of the actions that can be taken to correct the process behavior.

Finally, an industrial application scenario for the proposed approach is studied. By means of it, we verify that (i) the proposed semantic model is generic and extensible to accommodate a wide spectrum of manufacturing processes, and that its modular architecture allows the description of manufacturing resources with different levels of modularity; (ii) the use of stream reasoning together with classical reasoning approaches allows the detection of abnormal situations as well as of their possible causes in a suitable way; and (iii) the exploitation of the lattice helps in decision making, by allowing the identification of the actions that can be taken to correct the abnormal behavior of the process.

Road-map of the Thesis

This manuscript is composed of two parts. The first part introduces the theoretical foundations and gives a comprehensive review of existing research works to frame our work. It is structured into two chapters. The second part presents the contributions of this thesis. It consists of five chapters. Figure 1.11 shows the structure of this thesis. The content of each chapter is briefly described below.

Chapter 2. This chapter gives the theoretical foundations and basic concepts of smart systems (also known as knowledge-based systems). First, the classic architecture of smart systems is described. Then, the KREM architecture, a novel and generic knowledge-based framework for problem-solving in engineering disciplines, is exposed to frame our proposal. The concept of ontology is introduced as well as the key concepts and the associated technologies of the Semantic Web. The concepts of classical reasoning and stream reasoning are also presented. Finally, the notion of context and the main approaches for context handling are described.

Chapter 3. This chapter presents the related works about condition monitoring in Industry 4.0. The key elements of Industry 4.0, including Cyber-Physical Systems (CPS), the Internet of Things (IoT), Cloud Computing and Big Data analysis techniques are briefly described. Some of the existing maintenance strategies in the industrial domain are also introduced. The existing approaches for condition monitoring are reviewed. We classify the existing works for condition monitoring into data-driven approaches and knowledge-based approaches. The advantages and disadvantages of each approach are also discussed. In the review of approaches for condition monitoring, we pay special attention to the ontological models and their rule-based extensions that are relevant to represent the manufacturing domain as well as the existing ontological models for context modeling. Approaches for handling the representation of knowledge that change over time are also described in this chapter.

Chapter 4. This chapter provides a global introduction to the proposed framework. A general architecture of Industry 4.0, coming from the literature, is introduced. This general architecture permits a clear positioning of each of our contributions. An overview of the components of the proposed framework is presented as well as the interactions among them.

Chapter 5. In this chapter an ontology-based approach for context modeling in the industrial domain is proposed. The ontological model for the representation of the entities involved in a manufacturing process, emphasizing the modeling of the context, is described. The conceptual modules that shape the proposed ontological model are presented. An evaluation of the proposed ontological model is performed according to different ontology evaluation criteria such as structure, function and usability.

Chapter 6. This chapter presents an approach that combines stream and classical reasoning methods to detect certain situations that can lead to potential failures and their causes. The proposed approach enriches data collected from sensors with contextual information to allow real-time situation detection.

Chapter 7. This chapter presents an approach to establish an order among relevant situations, depending on how their constraints are related to each other. In this way, it is possible to identify the actions that can be taken to correct the abnormal situation, considering that certain actions can lead to other situations than can be also abnormal. The proposed approach uses the lattice theory to provide an expressive formalization of the hierarchy of situations by considering how the situation's constraints are related to each other.

Chapter 8. This chapter describes the design and implementation of the proposed framework. The technologies and tools used for this development are introduced as well as the implementation of the core functionalities of it. An illustrative case study from the industrial domain is presented as a proof of concept.

Finally, some concluding remarks about the issues tackled in this thesis are presented as well as some perspectives for future work.

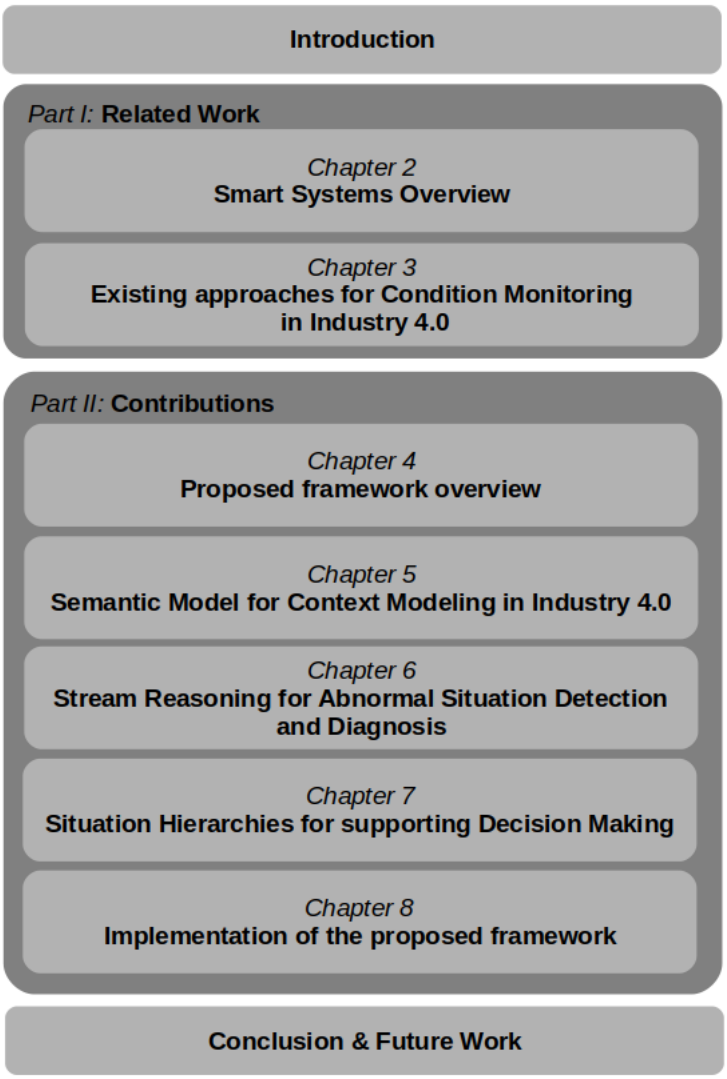

Figure 1.11 – Structure of the thesis.

Publications

The following research works were published during the development of this thesis:

Peer-reviewed journal papers

- Silva de Oliveira C., **Giustozzi F.**, Zanni-Merk C., Sanin C. and Szczerbicki E., (2020), *"Stream Reasoning to Improve Decision-Making in Cognitive Systems"*. Cybernetics and Systems, 51:2, 214-231, DOI: 10.1080/01969722.2019.1705553

- Cao Q., **Giustozzi F.**, Zanni-Merk C., de Bertrand de Beuvron F. and Reich C., (2019), *"Smart Condition Monitoring for Industry 4.0 Manufacturing Processes: An Ontology-based Approach"*. Cybernetics and Systems, 50:2, 82-96, DOI: 10.1080/0196972-2.2019.1565118.

Peer-reviewed international conference papers

- **Giustozzi F.**, Saunier J., Zanni-Merk C., *"Towards the use of Situation Hierarchies for supporting Decision Making: A Formal Lattice-Based Approach"*. In Proceedings of

the 14th International Rule Challenge (RuleML+RR 2020) as part of Declarative AI 2020, Jun 2020, Oslo, Norway. pp.73-86.

- **Giustozzi F.**, Saunier J. and Zanni-Merk C., *"Abnormal Situations Interpretation in Industry 4.0 using Stream Reasoning"*. In International Conference on Knowledge Based and Intelligent Information and Engineering Systems, KES-2019, 4-6 September 2019, Budapest, Hungary.

- **Giustozzi F.**, Saunier J. and Zanni-Merk C., *"Context Modeling for Industry 4.0: An Ontology-based Proposal"*. In International Conference on Knowledge Based and Intelligent Information and Engineering Systems, KES-2018, 3-5 September 2018, Belgrade, Serbia.

Part I

Related Work

Chapter 2

Smart Systems overview

Contents

In the last decade, a growing number of companies have realized that a tool that effectively enables the capture, representation, retrieval, and reuse of knowledge is the key to supporting various organizational decisions [OBGM11, Lei10]. However, large parts of useful knowledge are hidden and not readily available. Knowledge Technologies are computer-based techniques and tools that provide a rich and intelligent use of information. They combine ideas and applications from a number of fields: Psychology, Philosophy, Artificial Intelligence, Engineering, Business Studies, Computer Science and Web Technologies [Mil08].

A smart system is a computer program that uses Artificial Intelligence techniques to solve complex problems that would normally be performed by a person who has a specific expertise [Mil08]. These systems have a computational model of some domain of interest in which symbols serve as surrogates for real world domain artifacts, such as physical objects, events and relationships, etc [Sow99]. A knowledge base stores those symbols in form of statements and the system performs reasoning by manipulating these symbols. Applications can base their decisions on domain questions posed to the knowledge base. They are programmed to solve problems in a similar way to that of an expert, e.g. make inferences based on the current case and adopt the right strategy to solve the problem; they can deal with incomplete information by making requests for further information; they have a user interface that makes intelligent requests for information and can explain how it has arrived at its answers; and finally, they can be developed by reusing generic structures, rules and problem-solving methods.

In this chapter, the basic concepts of smart systems are introduced. Two architectures are described: first, the classic architecture of smart systems and then the KREM architecture. In subsequent sections, the four components of the KREM architecture are described in detail. Finally, in the last section, a summary of the chapter is presented as well as some relevant conclusions.

2.1 Smart Systems architecture

In this section, two smart system architectures are presented. First, the basic components of the classic smart systems architecture are described. Second, the KREM components that provide the framework for structuring our work are detailed.

2.1.1 The Classic architecture

Normally, the classic architecture of a smart system consists of the following components. Figure 2.1 shows this architecture and the interaction among its components.

- **Knowledge Base (KB)**: It contains domain-specific information, structures and rules. One of its components is a formal conceptual model of the domain of interest. Among the existing formal conceptual models, ontologies are formal representations of vocabularies that are specific to a certain area [Gru93]. The concept of ontology is detailed in section 2.2.1.

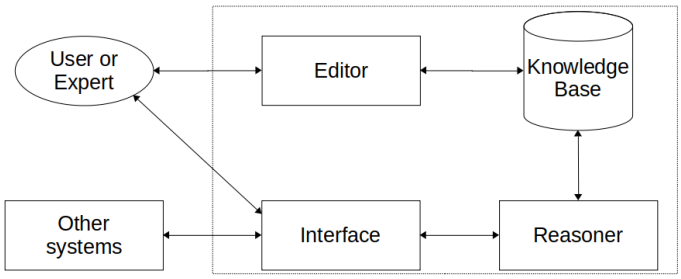

Figure 2.1 – Knowledge-based system architecture (adapted from [Mil08])

- **Reasoner (Inference or Reasoning Engine)**: The reasoner establishes the mechanisms that operate on the facts and rules to comply with some objective statement. For this, the inference engine implements a reasoning algorithm.

- In addition, some smart systems incorporate a series of interfaces for users or other systems to access knowledge and editors that facilitate the maintenance of the knowledge base.

The development of a smart system is a complex task that relies on the application of methods and tools specially designed to support the acquisition and management of knowledge. In particular, Knowledge representation is the fundamental task involved in the development of any information system. The choice of the objects and attributes of the real world that are captured by the program and the way they are represented are what determines what operations the system can perform and what queries it will be able to answer.

In summary, a smart system contains a conceptual yet executable model of an application domain. It is made machine-interpretable by means of knowledge representation techniques and can therefore be used by applications to base decisions on reasoning about domain knowledge.

Smart systems do not always provide satisfactory results when solving a problem. It suffers of several drawbacks:

- Incompleteness of the knowledge models for their implementation: Usually it is difficult to determine what knowledge needs to be captured to provide an appropriate solution to a real-world problem;

- Difficulty in knowledge elicitation: Most of the knowledge an expert operates is tacit, making that often, the set of elicited rules cannot accurately describe the problem solving process [Sla91]. This situation is known as the ”knowledge elicitation bottleneck” [HRWL83];

- Lack of efficiency: rule-based systems do not remember previously solved problems. Every time there is a new problem to be solved, the rule-based system has to execute rules from the beginning;

- Lack of robustness: if the current problem does not match any of the rules, the smart system will fail in providing a solution.

2.1.2 The KREM architecture

In [ZMS19] the authors proposed a novel architecture called KREM (Knowledge, Rules, Experience and Meta-Knowledge) to manage the complexity of developing smart systems. Its main goal is to solve the problems associated with knowledge acquisition from experts or from data and to incorporate the capitalization of experience with the aim of improving decision-making.

The four components of KREM are (Figure 2.2):

- The **Knowledge** component that contains the domain knowledge.
- The **Rules** component that allows different types of reasoning (monotone, spatial, temporal, fuzzy, or other) depending on the application.
- The **Experience** component that allows the capitalization and reuse of prior knowledge.
- The **Meta-knowledge** component that includes knowledge about the other three components, depending on the application.

Figure 2.2 – The KREM architecture with its four components (taken from [ZMS19])

The way domain knowledge is formalized shapes the way in which the rules are expressed. Experience comes to complete the available knowledge and rules. Finally, meta-knowledge directly interacts with the rules and the experience to indicate which rules need to be launched according to the context of the problem to solve. Meta-knowledge is knowledge about domain knowledge, about rules or about experience. This meta-knowledge can take the form of context, culture or protocols. Context is information that characterizes a situation in relation to interaction among human-beings, applications and their environment [DA99].

Unlike a traditional smart system, a KREM-based system is able to deal with incomplete expert knowledge models, by progressively completing them, learning with experience. The use of meta-knowledge, to formally represent the context of the decision to be made, can steer its execution more efficiently. The KREM model has already been successfully used in several applications: the semantic analysis of urban satellite images, the diagnosis in small and medium enterprises and the formalization of the Lean enterprise [ZMS19].

In the following, each component as well as different technologies for implementing the components of the KREM architecture are explained. We use this decomposition in four components to guide the review of related works in the following sections.

2.2 The Knowledge component

The knowledge component contains knowledge about a certain domain by means of a formal conceptual model. Among the existing conceptual models, ontologies have been widely used to formalize domain knowledge. This section introduces the concept of ontology. Then, semantic web technologies to develop ontologies are described.

2.2.1 Concept of Ontology

The notion of "ontology" originally denotes the study of existence in philosophy. In semantic technologies, ontologies are conceptual models of what "exists" in some domain, brought into machine-interpretable form by means of knowledge representation techniques.

Ontologies have been explored from different points of view, and there exist several definitions of what an ontology is. In computer science, the most cited definition of an ontology is the following.

Definition 1 *An ontology is a formal, explicit specification of a shared conceptualization of a domain of interest [Gru93].*

This definition captures several characteristics of an ontology as a specification of domain knowledge:

- **Formal** refers to the fact that the specification of domain knowledge in an ontology is machine readable and is interpreted in a well-defined way, which excludes natural language;
- **Explicit** means that the concepts and the constraints on their use are explicitly defined;
- **Shared** reflects the notion that an ontology captures consensual knowledge, that is, not private to some individual but accepted by a group; and,
- A **conceptualization** refers to an abstract model of some phenomenon in the world built by identification of the relevant concepts associated to it. Moreover, an ontology describes a conceptualization in general terms. Instead of making statements about a specific situation involving particular individuals, an ontology tries to cover as many situations as possible, that can potentially occur.

There are several reasons to develop and use ontologies according to [Noy01, WDTP04]. Some of the most common ones are: (1) to share a common understanding of the structure of information among people or software agents, (2) to analyse domain knowledge,

(3) to separate domain knowledge from operational knowledge, (4) to enable reuse of domain knowledge, (5) to infer high-level knowledge, and (6) to make domain assumptions explicit.

In the following subsections, the ontology-related background knowledge is introduced, including ontology components, common ontology development methods and the different types of ontologies.

Components of an ontology

The main components of an ontology are the following.

- **Concepts** (also called classes, categories or types) are abstract representations that reflect some aspects of a domain (e.g. physical objects, ideas, tasks, people, etc.). They map to unary predicates in First Order Logic (FOL). They represent the ontological categories that are relevant in the domain of interest. Some concepts that contain individuals are shown in Figure 2.3. There are three concepts in the figure: Person, Country and Animal. The concept Animal has a sub-concept called Mammal, which represents the animals that are mammals.

- **Relations** semantically connect concepts, as well as instances, specifying their relations. They map to binary predicates in First Order Logic. Two examples are the generalization-specialization relation *"is a"* and the composition relation *"is part of"*. Relations are also called properties or roles. Figure 2.3 shows relations (arrows) that link individuals. The names of these relations are livesIn and hasPet.

- **Instances** (or individuals) represent the named and identifiable concrete objects in the domain of interest, i.e. the particular individuals belonging to each concept. They map to constants in First Order Logic. Figure 2.3 shows a set of instances that represent the concepts of Person, Country and Animal.

- **Axioms** are statements that allow to structure the domain. There are two fundamental types of axioms: integrity and derivation axioms. Integrity axioms define the relationships among the different concepts in the domain, along with their eventual restrictions. These restrictions may be static (e.g. "A waiter is a man who is also a waiter") or cardinal (e.g. "A person is a parent if and only if has at least one child"). Derivation axioms model the domain rules that permit to deduce new facts from stored knowledge. They are expressions like **IF** *{conditions}* **THEN** *{consequences}*. Consequences are deduced as long as conditions are met.

Ontology development

Building an ontology is a complex task mainly because there is no single way to model the knowledge about a domain. Consequently, it usually results in an iterative process. It is advisable to follow some basic design criteria (called ontological principles). These principles are described by [Gru93, GPFLCGP04]:

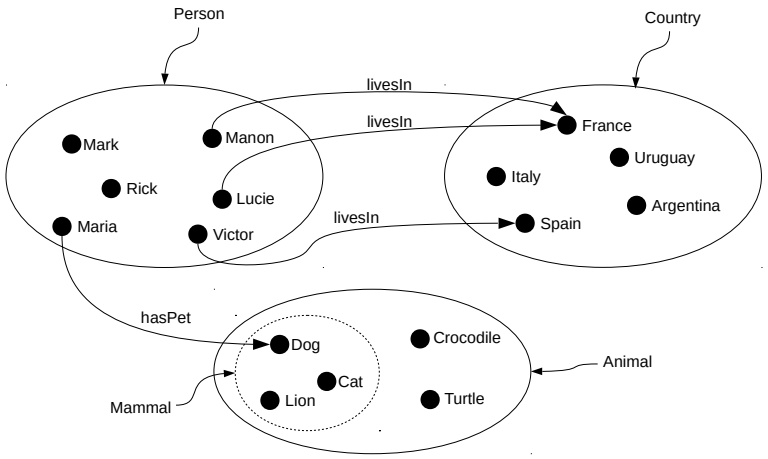

Figure 2.3 – An example of three concepts (Person, Country and Animal) and relations (livesIn and hasPet) among instances.

- **Clarity**, the model must represent the desired meaning of the terms in a clear way.
- **Coherence**, the model must allow only inferences that are consistent (not contradictory).
- **Interoperability**, the model must have well-defined semantics so that multiple entities in different environments can understand and inter-operate with each other correctly.
- **Re-usability**, the model needs to support the reuse of its modules to reduce the information needed to be transferred and processed.
- **Extensibility**, the model should be extensible for developing new functions and support extension of domain specific knowledge based on existing vocabulary.
- **Modularity**, the model needs to capture different structural knowledge to decrease the complexity of the model and simplify the development phase.

Several methodologies can be found in the literature to develop an ontology [Noy01, UG96]. In [UG96], the authors highlight a series of elementary steps to build an ontology. These steps are described below and are shown in Figure 2.4:

1. Identification of the purpose and the scope: It is important to be clear about why the ontology is being built and what its intended uses are. In particular, "competency questions" should be answered at this step. "Competency questions" are user-oriented questions that allow to scope the ontology. They are questions that must be answered through exploring and querying the ontology.
2. Construction of the Ontology.

(a) Ontology capture:

i. identification of the key concepts and relationships in the domain of interest;

ii. production of precise unambiguous text definitions for such concepts and relationships;

iii. identification of terms to refer to such concepts and relationships;

iv. all the designers need to agree on all of the above.

(b) Coding: explicit representation in a formal language of the conceptualization captured in the previous step.

(c) Integration of existing ontologies: how and whether to use (all or part of) ontologies that already exist.

3. Evaluation: technical assessment of the ontologies with respect to the requirements specification, the competency questions, and/or the real world is done.

4. Documentation: to enable knowledge sharing.

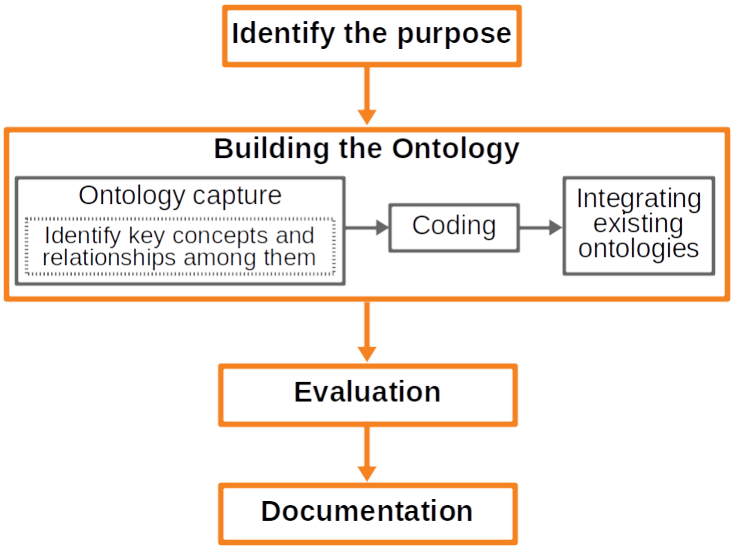

Figure 2.4 – Methodology for developing ontologies [UG96].

Type of Ontologies

A categorization of ontologies can be made according to their level of conceptualization. Different types of ontologies exhibit different potentials for reuse. The most prominent insights in this respect have been published in [Gua97]. There are different ways to characterize an ontology:

- Depending on the level of abstraction and scope of conceptualization we can distinguish among:

- **Upper Ontologies** (also called Foundational Ontologies) define general purpose concepts such as time, states, qualities, regions, etc., that are independent of a particular problem or domain. Among the most recognized top-level ontologies are DOLCE [GGM+02], SUMO [NP01] and BFO [SKB05].
- **Domain and Task Ontologies** describe respectively the general knowledge about a certain domain (such as Medicine or Industry) or a generic activity (such as diagnosis or control). Their entities are usually obtained by specialization of the concepts of an upper ontology.
- **Application Ontologies** provide the concepts that are required for a particular application. They are built by specializing the concepts of domain and task ontologies.

Figure 2.5 represents an inclusion hierarchy: the lower ontologies inherit and specialize concepts and relations from the upper ones. The lower ontologies are more specific and have thus a narrower application scope, whereas the upper ones have a broader potential for reuse.

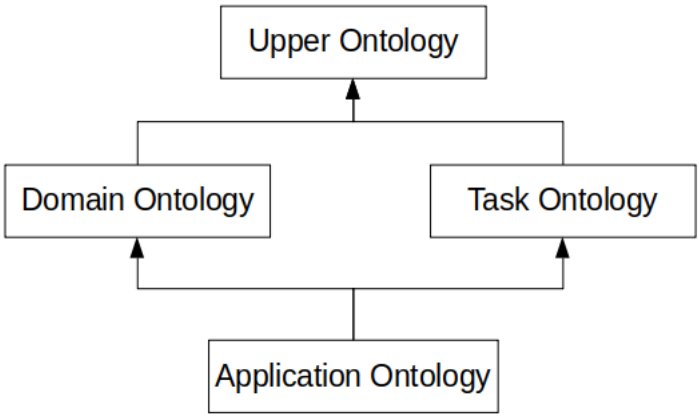

Figure 2.5 – Types of Ontologies.

- Depending on the semantic richness of the conceptualization, an ontology can be distinguished between:
  - **Light-weight Ontologies** are taxonomies of related concepts that do not include axiomatic definitions that formally define the semantics of their elements. Due to their simple design, they are sometimes not considered as ontologies.
  - **Heavy-weight Ontologies** model the semantics of the domain, including a rich axiomatization to clarify the meaning of their concepts.

There are several modeling approaches to support the development of formal ontologies through different formalizations. Some of the best known are Description Logic (DL) [BCM+03], First Order Logic (FOL) [Hod01] and the Frames approach [Min74]. Each of

these paradigms is supported by several languages, which differ in terms of expressiveness and computational tractability. In particular, Description Logic is the paradigm chosen by the Semantic Web community, and also the most used in the development of ontologies, by means of the Web Ontology Language (OWL) [BHS05].

2.2.2 Semantic Web Technologies

The Semantic Web is an initiative of the World Wide Web Consortium[1] (W3C) for a new generation of the Word Wide Web. The idea was originally presented by Tim Berners-Lee as a solution to improve the exploitation of knowledge stored on the web. Its goal was to give meaning to the content of traditional websites, allowing that knowledge content to be interpreted by software systems.

Semantic Web technologies offer a set of solutions to capture and manage the semantics of data. In the Semantic Web, ontologies are used as knowledge representation models and the language for writing such models is the Web Ontology Language (OWL).

Figure 2.6 shows the Semantic Web "layer cake" proposed by Tim Berners-Lee [BLF00]. This figure shows the hierarchy of the Semantic Web languages mentioned above, within the Semantic Web stack. OWL is based on Resource Description Framework (RDF) and Resource Description Framework Schema (RDFS), which are layered on the top of the Extensible Markup Language (XML). OWL provides additional vocabulary along with formal semantics to facilitates greater machine interpretability of Web content than that supported by XML, RDF, and RDF Schema [MVH+04]. The main languages are described below.

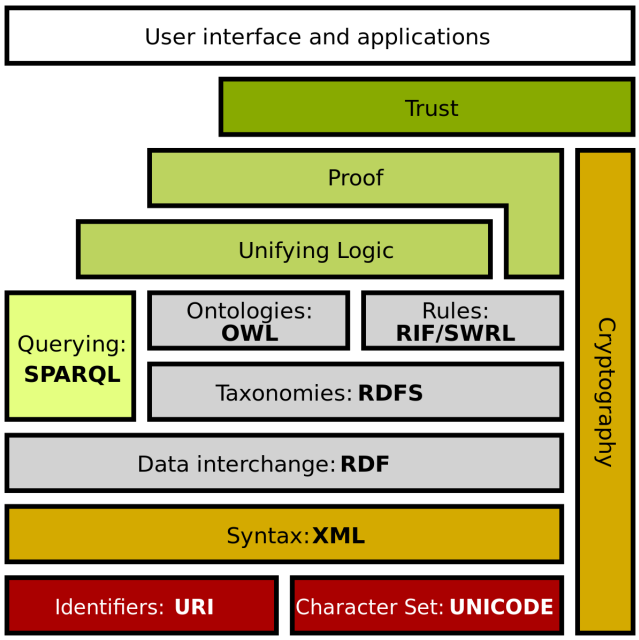

Figure 2.6 – Semantic Web Stack [BLF00].

[1] https://www.w3.org/

XML. Extensible Markup Language is a markup language that defines a set of rules for encoding documents in a format that is both human and machine-readable [BPSM+00]. Users creates their own data-model and then they create the corresponding XML document with their defined entity tags and values. In order to use the information embedded in the document users must follow their own data-model. In addition, XML does not provide any logic relationship among different structures inside the users' data-model, which makes it difficult to express complex knowledge and relationships with XML alone. Therefore, an upper layer is needed to express this knowledge. RDF addresses this issue.

RDF. The Resource Description Framework (RDF) is a generic data model for interchanging data in the Web recommended by the W3C. In RDF, data is represented as triplets consisting of subjects, predicates, and objects. Formally, an RDF triplet is defined as follows:

Definition 2 *Let* $\mathtt{I}, \mathtt{B}, \mathtt{L}$ *be disjoint infinite sets of URIs*[2]*, blank nodes, and literals, respectively. A tuple* $(\mathtt{s},\mathtt{p},\mathtt{o}) \in (\mathtt{I} \cup \mathtt{B}) \times \mathtt{I} \times (\mathtt{I} \cup \mathtt{B} \cup \mathtt{L})$ *is denominated an RDF triple, where* $\mathtt{s}$ *is called the subject,* $\mathtt{p}$ *the predicate, and* $\mathtt{o}$ *the object.*

An example RDF graph is shown in Figure 2.7. It expresses the fact that "John is the father of Maria". Despite the fact that RDF provides an open language to express knowledge, it does not make assumptions or define the semantics about any particular application domain (e.g. "Fathers are male").

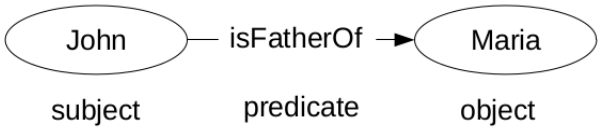

Figure 2.7 – RDF graph example.

RDF Schema. RDFS extends RDF by allowing to define a particular vocabulary (i.e. the terms) to be used in an RDF graph. RDFS is used to indicate that specific kinds or classes of resources are described, and specific properties are used to describe those resources. RDFS is a simple ontology definition language (compared with other languages such as OWL). It only allows to define taxonomies and do some basic inference about them.
RDFS defines several classes which allow to identify the various concepts that need to be described in a particular domain. A class in RDFS corresponds to the generic concept of a type or category. In RDFS, a class `C` is defined by a triple of the form `C rdf:type rdfs:Class`. For example, we can define the class `Animals` (`Animals rdf:type rdfs:Class`) to represent animals.
Essentially, RDFS provides modelling primitives for describing how properties and classes are intended to be used together in RDF data. Some of the most important ones are `rdfs:Class`, `rdfs:subClassOf` (relating a class to one of its superclasses; all instances

[2]Uniform Resource Identifiers (URIs) are used to uniquely identify each resource in the web.

of a class are instances of its superclass), `rdfs:subPropertyOf` (relating a property to one of its superproperties), `rdfs:domain` (indicating that a particular property applies to instances of a designated class; i.e. defining the domain of the property), `rdfs:range` (indicating that the values of a particular property are instances of a designated class; i.e. defining the range of the property). Important annotation constructs are also added in RDFS such as `rdfs:label` (associating human-friendly label, a name, with a resource) and `rdfs:comment` (associating longer text with a resource). A complete list of all modeling primitives can be found at [AH04].

OWL. The Web Ontology Language [MVH+04, MR07] is an expressive ontology language based on Description Logics with sound and complete reasoning algorithms [BCM+03]. OWL is built on top of RDF Schema. To summarize, XML is the basis, RDF is a dialect of XML for describing and sharing web resources; RDF Schema is a set of RDF constructions; and OWL is a form of RDF Schema for building ontologies. Compared to RDF, OWL offers a richer set of operators to define the elements of the ontology (e.g. classes, properties, etc.). It also possesses a formal semantics that allows reasoning.
The semantics of OWL addresses distributed knowledge representation scenarios where complete knowledge about the domain cannot be assumed. OWL has two variants:

- **OWL DL** only has constructions that are supported by Description Logics. OWL 2 DL is based on the SROIQ logic [HKS06]. It retains computational completeness and decidability. OWL DL includes all OWL language constructs, but they can be used only under certain restrictions (e.g., while a class may be a subclass of many classes, a class cannot be an instance of another class).

- **OWL FULL** supports a larger number of constructions than OWL DL, but these are not supported by Description Logics and it is undecidable. The purpose of OWL Full is to extend the vocabulary of the RDF Schema, but with no computational guarantees. For example, in OWL FULL a class can be treated simultaneously as a collection of individuals and as an individual.

OWL is based on the Open World Assumption (OWA). This assumption enables incomplete data to be used for reasoning. In a Closed World Assumption (CWA), such as in entity-relationship databases, anything that is not explicitly expressed is considered as false. Under the OWA, anything that is not declared as false may be true.
As described in section 2.2.1, an ontology is basically composed of individuals, classes and properties. Individuals or instances represent the particular elements of the domain, classes can be interpreted as sets of instances and properties are binary relationships over those individuals.

Below are some of the main OWL characteristics definitions of properties and classes. Each property has a domain and a range. There are three main types of properties:

- **Object properties** describe relationships between individuals.

- **Data properties** are relationships between individuals and data values. The range of these properties is a Datatype, such as Integer, String, Float, among others.

- **Annotation properties** are used to add metadata to classes, to individuals and to object/data properties.

OWL has several constructions to specify the characteristics of the properties. Some of the most relevant are:

- **Inverse**: a property p_1 is inverse of a property p_2 iff $\forall x, y : p_1(x, y) \Leftrightarrow p_2(y, x)$.
- **Functional**: a property p is functional iff $\forall x, y, z : p(x, y) \wedge p(x, z) \Rightarrow y = z$.
- **Symmetric**: a property p is symmetric iff $\forall x, y : p(x, y) \Rightarrow p(y, x)$.
- **Anti-symmetric**: a property p is anti-symmetric iff $\forall x, y : p(x, y) \wedge p(y, x) \Rightarrow x = y$.
- **Reflexive**: a property p is reflexive iff $\forall x : p(x, x)$.
- **Un-reflexive**: a property p is un-reflexive iff $\forall x, y : p(x, y) \Rightarrow x \neq y$.
- **Transitive**: a property p is transitive iff $\forall x, y, z : p(x, y) \wedge p(y, z) \Rightarrow p(x, z)$.
- **Property chain**: if a property p_1 is defined as a composition of the properties p_2 and p_3, then $\forall x, y, z : p_2(x, y) \wedge p_3(y, z) \Rightarrow p_1(x, z)$

In OWL, classes are defined by a set of restrictions. These restrictions set out the conditions that an individual must meet in order to be a member of the class. Three types of restrictions are available in OWL:

- **Restrictions on Quantification** OWL supports both existential ($\exists$) and universal ($\forall$) quantifications. An existential restriction (`someValueFrom` axiom) applied to a property describes the class of individuals who relates to, at least, a member of the class defined as the range of that property. An universal restriction (`allValuesFrom` axiom) applied to a property describes the class of individuals who, if they are in the specified property then the individuals are only related with members of the class defined as the range in the specified property.
- **Cardinality restrictions** allow to specify the minimum, maximum or exact number of times each class member participates in a given relationship.
- **Value restrictions** (hasValue restriction) The `hasValue` axiom allows to define classes based on the existence of specific values in their properties.

SPARQL Query Language. SPARQL[3] (a recursive acronym for SPARQL Protocol and RDF Query Language) is a query language able to retrieve and manipulate data stored in RDF. SPARQL is based on the RDF Turtle serialization syntax[4] and on graph pattern matching. A graph pattern is an RDF triplet containing variables, as evoked earlier. SPARQL is inspired by the Structured Query Language (SQL), thus many of its features are similar to

[3] https://www.w3.org/TR/sparql11-query/
[4] https://www.w3.org/TR/turtle/

it. A SPARQL query consists of triple patterns, conjunctions, disjunctions, and optional patterns. Triple patterns are similar to RDF triples where the subject, predicate, and object may be variables. In a query, the variables act like placeholders which are bound with RDF terms to build the solutions. In general, a SPARQL query consists of four main parts as described below

- **Prefix declarations** allow the definition of IRI (Internationalized Resource Identifier) prefixes that can be used for shortcuts later in the query.
- **Result clause** allows for specifying what type of SPARQL query is being executed, and what results should be returned.
- **Query pattern** allows for specifying the query patterns that are matched against the data and used to generate the variable bindings of the defined variables in the query.
- **Query modifiers** allow for ordering and grouping the results.

Depending on the query type, the output of a query can be an RDF graph, a table, or a Boolean value. Listing 2.1 shows an example of a select query using the FOAF vocabulary[5] that returns all persons who know someone with the first name `Maria`.

```
#PREFIX DECLARATIONS
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
#RESULT CLAUSE
SELECT ?person
#QUERY PATTERN
WHERE {
   ?person foaf:knows ?p.
   ?p      foaf:firstName "Maria".
}
```

Listing 2.1 – SPARQL query example (comment lines are prefixed with '#')

2.3 The Rules component

The Rules component of a KREM smart system allows different types of reasoning over the individuals of the ontologies contained in the *Knowledge* component. We consider reasoning as the ability to generate non-trivial conclusions from premises or assumptions [ARFS12]. The known asserted facts are called explicit knowledge, while the inferred ones as implicit knowledge.

In the following section we describe, the Semantic Web Rule Language (SWRL) that extends OWL axioms with IF-THEN like rules. These logical rules are executed by a reasoner, which is a piece of software able to infer logical consequences from a set of asserted

[5] http://xmlns:com/foaf/spec/

facts or axioms [Abb12]. Some of the most known rules engines in the Semantic Web community are *Jess*[6], *Jena*[7] and *Drools*[8].

Current solutions for reasoning on ontologies were developed for static or slow changing data which is not the case for the dynamic nature of the data in the domain of Industry 4.0 where the manufacturing machines are equipped with sensors that collect data continuously. Sensor data are an example of stream data which are rapidly changing data. These data need to be quickly consumed and reasoned over. For example, if the values from a particular property drops from its allowed threshold then this information needs to be consumed quickly and an appropriate decision should be made.

To bridge this gap, a number of recent works propose to unify reasoning and stream processing, giving rise to the research field of Stream Reasoning (SR). The notion of stream reasoning is explained in section 2.3.2.

2.3.1 SWRL - The Semantic Web Rule Language

The Semantic Web Rule Language is a standard language based on OWL DL and on the Rule Markup Language (RuleML) [HPSB$^+$04]. SWRL allows to define rule expressions involving OWL concepts, thus enabling more powerful deductive reasoning than OWL alone.

SWRL extends OWL axioms with Horn-like rules [Hor51]. A Horn clause is a disjunction of literals with at most one positive; i.e. $\neg p \vee \neg q \vee \ldots \vee u$. This clause is equivalent to $p \wedge q \wedge \ldots \Rightarrow u$. The rules are composed in the form of implication between an antecedent (body) and a consequent (head).

$$antecedent \rightarrow consequent$$

The intended meaning can be read as: whenever the conditions specified in the antecedent hold, then the conditions specified in the consequent must also hold.

Both the rule's body and head consist of zero or more atoms. Atoms on these rules are OWL concepts: `C(x)` and `P(x, y)`, where `C` is an OWL class, `P` is an OWL property and `x`, `y` are variables, OWL individuals or OWL data values. Moreover, predicates `sameAs(x, y)`, `differentFrom(x, y)` and `operator(r, x, ...)` are added to the language to include the semantics of interpreting same and different objects and SWRL operators libraries, respectively.

Although SWRL extends OWL, it has some limitations. For example, *"SWRL rules cannot be used to modify existing information in an ontology"* [RBBSD11]. Indeed, SWRL rules are able to create new knowledge, i.e. new properties between existing instances, but they cannot create new instances "on the fly", nor change the value of an instance property.

SWRL, which has become the standard rule language in Semantic Web, is not used in stream data applications. Its nature makes SWRL unsuitable for doing continuous inference over stream data. For example, using aggregate functions on a particular window of

[6] https://jess.sandia.gov/
[7] https://jena.apache.org/
[8] https://www.drools.org/

time over data streams cannot be expressed in SWRL, nor the frequency with which a rule should be executed.

2.3.2 RDF Stream reasoning

In many real-world applications data take the form of continuous streams instead of the form of finite data sets stored in a traditional repository. This is the case for monitoring of network traffic, for telecommunications management, for manufacturing, for sensor networks, and for many other domains. In such applications, instead of classical "one-shot" queries, continuously running queries, which return new results as new data arrive on the streams are needed. A data stream is a sequence of items received continuously and in real-time, ordered either implicitly, by arrival time, or explicitly, by means of timestamps.

A number of Data Stream Management Systems (DSMSs) have been developed in the last decade to tackle the challenges raised by dynamic and high-velocity data streams.

To take advantage of the collected data streams, integration with domain knowledge containing resource specification and planning information is important. Reasoning on this information can infer new knowledge of the resources' current condition. Well integrated and aggregated information helps to monitor the resources and allows informed decisions.

However, the issues of handling heterogeneity, integration, and interpretation of data streams at semantic level are not addressed by DSMSs. The Semantic Web community, through its standards and technologies, provide answers to these issues, while employing ontologies and RDF data models. The integration of DSMSs and Semantic Web technologies has led to the semantically-enabled stream processing, usually called RDF Stream Reasoning [BBC+10]. The novelty of this approach is in extending incremental reasoning techniques with the notions of time window and continuous processing.

Time window. Traditional reasoning is based on the idea that all the information available should be taken into account when solving a problem. In stream reasoning, this principle is set aside and reasoning is restricted to a certain window of concern which consists of a subset of statement recently observed. This is necessary for several reasons. First of all, ignoring older statements allows to save computing resources in terms of memory and processing time to react to important events in real time. Secondly, in many real-time applications there is a silent assumption that older information becomes irrelevant at some point.

Continuous Processing Traditional reasoning approaches are based on the idea that the reasoning process has a well defined beginning (when a request is made to the reasoner) and end (when the result is delivered by the reasoner). In stream reasoning, requests are registered at the reasoner and continuously evaluated against a knowledge base that is constantly changing.

Figure 2.8 shows a conceptual view of a stream reasoning system as envisioned in [DVCB+08]. The system takes multiple data streams as well as multiple static or infre-

quently changing knowledge models as input. Users can register reasoning tasks (or queries) at a stream reasoner. Tasks consist of a predicate that is to be instantiated. The definition of this predicate as well as the additional information necessary to perform derivations can be considered as static knowledge models that are given to the system. The stream reasoner finds instantiations of the different predicates by applying a reasoning method and outputs either static or stream data. A static output is a variable binding that can be further processed by any reasoner. As an alternative, the stream reasoner can produce a stream as a result itself. In this case, instantiations of the goal predicates are added to the output stream as they are derived. Such a streaming answer is appropriate for further processing by other stream reasoners.

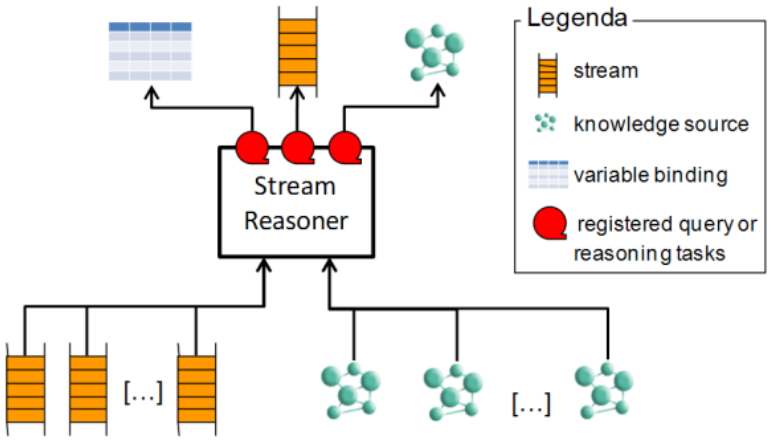

Figure 2.8 – Stream Reasoning (from [SCDV+19])

RDF Stream Processing engines

SPARQL (see section 2.2.2) is designed for one-shot queries. A number of RDF Stream Processing (RSP) languages are proposed to extend SPARQL with operators that take into account the streaming nature of RDF streams. These languages, similar to Continuous Query Language (CQL) [ABW04], integrate temporal windows and the `FROM STREAM` operator to define query graphs over a set of RDF streams, and to process them continuously. In this section, we describe existing approaches aiming to perform reasoning over data streams and present a comparison among them.

All these approaches use RDF to represent RDF streams, that are ordered sequences of pairs, where each pair is constituted by a RDF triplet and its timestamp `t`: `(<Subject,Predicate,Object>,t)`.

Streaming SPARQL [BGJ08] is one of the first language extensions to SPARQL for processing RDF data streams. The contribution of the model in the RSP field can be regarded as primarily theoretical. The first work on defining an algebra for a streaming version of SPARQL [GGKL07] was also theoretical, although it did provide an initial comparison with a purely static approach. Both approaches have been used as inspiration for later approaches.

C-SPARQL (Continuous SPARQL) [BBC+10] proposes a query language to process RDF

Table 2.1 – Comparison of RSP engines.

RSP engines	Input Model	Time Model	Reasoning	Background Knowledge
C-SPARQL	Triplet streams	Time points	RDFS subset	Yes
CQELS	Triplet streams	Time points	No	Yes
Straming SPARQL	Triplet streams	Time points	No	Yes
Sparkwave	Triplet streams	Time points	RDFS subset	Yes

streams. It supports time-stamped RDF triplets as input and uses a periodic execution strategy to continuously execute queries over these RDF streams. It has the capability of integrating both RDF streams and static background knowledge represented as RDF triples. Given that streams are intrinsically infinite, data are usually read through time windows using the CQL window concept [ABW04]. Queries are executed on all the triplets that are received during a given time interval. In C-SPARQL, continuous queries are divided into static and dynamic parts and streaming data is transformed into non-streaming data within a specified window in order to apply standard algebraic operations, such as aggregate functions like `COUNT`, `COUNT DISTINCT`, `MAX`, `MIN` and `AVG`. The static parts are loaded into relations, and the continuous queries are executed by processing the stream data against these relations.

CQELS (Continuous Query Evaluation over Linked Stream) [PDTPH11] is another language that combines static and streaming data in RDF format. Similar to C-SPARQL, it provides windowing and relational operators together with ad-hoc operators for generating new streams from the obtained results. CQELS queries deal with triplets in an element-based window (a given number of triplets). The main difference with C-SPARQL is that CQELS offers a processing model in which query evaluation is not periodic, but triggered by the arrival of new triples. These different execution methods lead to the possibility of having different query results produced for the same query and input data.

Finally, another possible approach is Sparkwave [KCF12] that is an RSP implementation that supports continuous reasoning over RDF data streams and time-based windows. Preliminary evaluations of the execution engine have shown that it can provide higher throughput than both C-SPARQL and CQELS under certain conditions. However, the experiments were based on an adaptation of the Berlin SPARQL Benchmark [BS09], which is designed to evaluate the performance of RDF stores, and contained no queries that required more than a single data stream. Sparkwave has some known limitations with respect to the size of the background knowledge that can be supported efficiently, and it provides only limited reasoning functionalities [MUVHB14].

Table 2.1 summarizes various features of existing RSP engines described above. All these RSP engines use simple streams, each incoming item consisting of a triplets associated with its timestamp. From the reasoning aspect, C-SPARQL and Sparkwave support a subset of RDFS rules to infer information from streams. CQELS does not provide reasoning features. All the RSP engines support integration of domain knowledge in the querying process, but reasoning capabilities on streaming data are limited.

2.4 The Experience component

The KREM architecture incorporates the *Experience* component, which allows the capitalization and reuse of prior knowledge. When the knowledge models, constructed in the *Knowledge* component, suffer from incompleteness (described in section 2.1.1), the *Experience* component performs as a complementary model to deal with this issue. The *Experience* component improves the knowledge model by progressively completing it with the experience acquired from the interventions of human experts. Furthermore, the reasoning process in the *Rules* component enables the evolution of the initial set of rules with experience, as proposed by [TSC+12]. More precisely, every time the native reasoner is executed, the system stores both the output of the reasoner and the final decision, if any, made by the expert. Decisions that do not follow the proposition of the reasoner produce an evolution of the set of experience rules. This process is termed as experience capitalization [ZM15].

This section presents Cased-based Reasoning (CBR) and Set of Experience Knowledge Structure (SOEKS), two complementary approaches for the capitalization of experience.

2.4.1 Case-based reasoning

CBR is regarded as a way to solve problems based on the retrieval and adaptation of cases, which are episodic descriptions of problems and their associated solutions [All94]. The CBR approach is used to capture experience by reproducing problem solving process of experts, at the psychological level, experts go back to previous problems they have solved and try to adapt the previous solution to solve the new problem.

The CBR approach can deal with the following situations:

- The elicitation of knowledge becomes a task of gathering typical cases and identifying features of the attributes, so that creating an explicit knowledge model of the domain is no longer necessary [WM94];
- A CBR system is able to access or manage large volumes of information [Rie88], thus giving available candidate solutions to new problems from analogies of past cases;
- A CBR systems can learn by acquiring new knowledge as cases [HH92], thus making maintenance easier [WM94] without raising logical incoherence.

From the beginning of the development of CBR, applications aiming at exploiting knowledge from experience in different domains have appeared; some examples are applications in law [AR88], civil engineering [WA94], marketing [CL05] and decision making [Kol91].

Case-based reasoning is suitable for experience capitalization in two aspects: experience collection and reuse. On the one hand, the main component of a CBR-based system is the case base that contains previous experiences of problem solving. Therefore, a suitable way to represent cases is vital. On the other hand, solving a new problem consists in finding old problems similar to the new one and constructing a solution either by reusing

old solutions or by adapting them. It is made possible by going through a cyclic process with four activities [AP94] (2.9):

1. retrieving the most similar case (*i.e.* most useful) to the new problem;
2. analyzing the possibility of reusing directly the solution of this case;
3. adapting the proposed solution if necessary; and
4. retaining the new case by updating the case base.

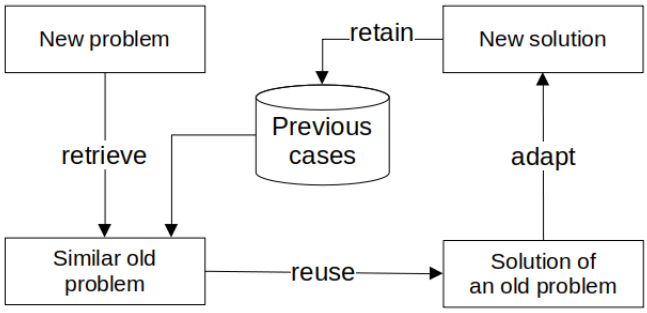

Figure 2.9 – The CBR architecture (adapted from [AP94])

Two crucial aspects are to be considered when applying a CBR approach, (1) in which way a case is represented and (2) in which way similarity between cases is calculated for retrieval purposes.

To deal with both aspects, SOEKS is presented below as a case representation framework and the similarity calculation measures for DDNA [STH+12] are used for case retrieval.

2.4.2 SOEKS - Set of Experience Knowledge Structure

Set of Experience Knowledge Structure (SOEKS) is a knowledge structure for representing experience. It can store uncertain and incomplete information for further knowledge processing [SS09, SST07]. SOEKS can be shaped in a language such as XML or OWL [SMASC09].

The experience collected by SOEKS can assists organizations in making precise decisions, predictions, and recommendations [SS09]. SOEKS is a dynamic structure that is dependent on the information and data that it has received.

A SOEKS contains four components [SS09] (Figure 2.10):

- Variables usually involve representing knowledge using an attribute-value language. This is the traditional frames approach [Min74] in knowledge representation.
- Functions describe associations among variables. Therefore, the set of experience uses functions and establishes links among the variables constructing multi objective goals.

- Constraints are another form expressing relationships among the variables. A constraint is a restriction of the feasible solutions in a decision problem and limits the performance of a system with respect to its goals.

- Finally, rules are suitable for representing inferences or for associating actions with conditions under which the actions should be performed. They are conditional relationships of the universe of variables.

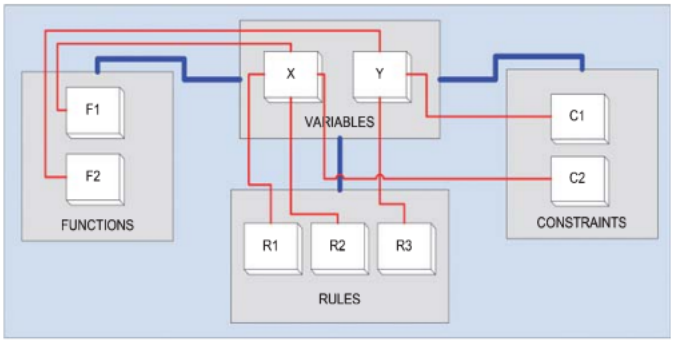

Figure 2.10 – A set of experience knowledge structure.

Each formal decision event can be stored in a combined structure of those four components of the SOEKS. Figure 2.11 shows part of the object properties in the SOEKS concept hierarchy with the four components involved in decision making events: variables, functions, constraints and rules.

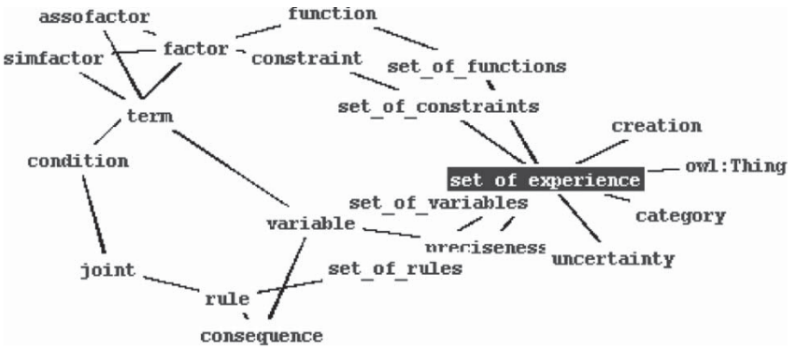

Figure 2.11 – The SOEKS ontology object properties.

A set of experience makes up a decisional *chromosome*, which stores decisional strategies for that category. Collection of these decisional chromosomes establishes an entire inference tool to offer a blueprint of knowledge inside a system, machine, or organization [SMASC09]. It is called DDNA (Decisional DNA) of the organization. In this way, DDNA is a knowledge repository that is designed to capture decisional fingerprints inside organizations through the use of SOEKS [SMASC09], making it also shareable and transportable [SST07]. Therefore, SOEKS is used as a case representation framework and the similarity calculation measures for DDNA [STH+12] are used for case retrieval.

2.5 The Meta-Knowledge component

Meta-knowledge is knowledge about domain knowledge, about rules and about experience. In particular, this component provides a formal representation of the context, to identify the one in which actions should be taken. In the industrial domain, meta-knowledge can control the firing of different sets of actions or rules according to different contexts, *e.g.* the rules for detecting the deviation of a machine temperature above its threshold may change according to the humidity of the machine's environment. In this way, the use of meta-knowledge can steer the execution of smart systems.

Context has been studied in areas like knowledge-based and ubiquitous systems, either for handling complex knowledge dynamically [GMD12] or for providing smart human interfaces [DAS01]. Diverse representations of context exist in different research areas.

According to [SAW94b], context-awareness is the capability of a system to gather information about its context or environment at any given time and to adapt its behavior accordingly by providing suitable services. The most widely referenced definition of context is the following.

Definition 3 Context *is any information that can be used to characterize the situation of an entity. An entity is a user, a place, or a physical or computational object that is considered relevant to the interaction between an entity and an application, including the entity and the application themselves [DA00].*

Such entities and applications are capable of specifically reacting to their current location, time and other environment attributes and adapt their behavior without explicit user intervention, thus aiming at increasing usability and effectiveness. In this section, we study the existing models for context handling and analyze their advantages and drawbacks with respect to the requirements of Industry 4.0.

There has been some research regarding context information management methods. A complete survey of them is presented in [PZCG13]. According to these authors, the context information life-cycle is composed of four phases:

- **Acquisition:** context needs to be acquired from various sources. The sources can be physical or virtual sensors.
- **Modelling:** the collected data needs to be modelled and represented meaningfully.
- **Reasoning:** modelled data needs to be processed to derive high-level context information from low-level raw sensor data.
- **Dissemination:** both high-level and low-level context have to be distributed to the consumers who are interested in it.

In our work, we focus on context modeling and reasoning. Approaches for context handling can be divided into: Data-driven approaches and Knowledge-driven approaches.

2.5.1 Data-driven approaches for context handling

The data-driven approaches learn the context models from datasets collected from diverse sensors via data mining and machine learning techniques [BBRC09, CNW12]. Among classical machine learning approaches, we can cite Decision trees, Bayesian networks, Neural networks, support-vector machines (SVM), among others. Decision tree is a technique that builds a tree from a dataset that can be used to classify data. This technique has been used in [HWCH08] for context handling in a learning environment. Bayesian networks is a technique based on probabilistic reasoning concepts [MN07]. Entities and relationships among them are represented by directed acyclic graphs and probabilities. Bayesian networks are commonly used in combining uncertain information from a large number of sources and deducing higher-level contexts [KK08], [POC11]. Artificial neural networks is a technique inspired from the biological neuron system. They are typically used to model complex relationships between inputs and outputs or to find patterns in data. It has been applied in [KK10] for pervasive healthcare monitoring. Support vector machines are widely used for pattern recognition in context-aware systems. It has been used to detect activity recognition of patients in the healthcare domain [DMT+07] and to learn situations in a smart home environment [BCR09].

Data-driven approaches have advantages such as a better handling of uncertain knowledge and the possibility to capture knowledge about temporal dependencies. However, they suffer from the cold start problem, since large datasets are needed to train the model for each activity. They also lack features regarding interoperability and re-usability.

2.5.2 Knowledge-driven approaches for context handling

In contrast to the Data-driven approaches, in Knowledge-driven approaches the difference between context modeling and context reasoning phases is clearer. Therefore, first we discuss the different techniques for context modeling and then we address the different context reasoning techniques.

In the context acquisition phase, data in multiple formats is obtained. To use it, it is necessary to store it in a machine readable and processable form, hereby all data should be converted into a unified format such that the context can be understood and shared. This can be achieved by the use of knowledge representation approaches, classified by the scheme of the used data structures to represent and exchange contextual information. Some of them may be classified in more than one category. The most relevant approaches are:

- **Key-value models** [SAW94a] use key-value pairs to represent information in a simple way, but that cannot embed complex structuring information.

- **Markup Scheme models** [W3C, HBS02, IRRH03] use XML or RDF/S (described in section 2.2.2) and describe information with markup tags. Examples such as Composite Capabilities/Preferences Profiles (CC/PP) allow a user to describe their devices with some predefined functionalities [W3C]. However, it is not flexible and

requires to use only the defined unambiguous attributes, and is limited in terms of expressivity of complex relationships.

- **Graphical models** [HIR03, HIR02, HM10, BKE03], such as the Unified Modeling Language (UML), allow the user to visually represent the classes and relationships in a particular domain. The key limitation of these Graphical models is that they only allow static specification of specialization and generalization of classes and relationships.

- **Object Oriented models** [SBG99] allow the user to design the abstract model of the context as well as the instances in it. These object oriented models are flexible but they do not provide complex logic inference . Logic defines the conditions and can derive related conclusions.

- **Logic Based models** [McC93, MB97] use First Order Logic for context modeling with its associated reasoning capabilities. It provides much more expressive richness compared to the other models discussed previously. Reasoning is possible up to a certain level. Lack of standardisation reduces the re-usability and applicability.

- **Ontology Based models** [SLPF03b, SLPF03a, ÖA97, DB03] can flexibly describe abstract information, logic inferences and represented instances. The context is organised into ontologies using semantic technologies. A number of different standards and reasoning capabilities are available to be used depending on the requirement. A wide range of ontology development tools and reasoning engines are also available.

These context modeling techniques require the task of context reasoning to deduce contexts of high abstraction level from simpler low abstraction level contexts. This task may also deal with some basic functionalities such as validating the context values, filling in missing values, checking context inconsistencies and applying some processing or calculations to obtain new values. Popular reasoning methods include, in addition to those mentioned in the data-driven approaches section.

- **Case-Based Reasoning.** In this method, context knowledge is deduced from previous similar cases [MKP04]. However, it is difficult to accurately calculate the similarity of different cases.

- **Rule-Based Reasoning.** This reasoning method infers high level information on the basis of predefined rules [GVCF07]. A limitation of this method is the definition of the rules in advance, as these rules can be complex to define.

- **Ontology-Based Reasoning.** As described in section 2.2.2, it is based on description logic. This technique applies logical reasoning over low-level, explicit context to derive high-level, implicit context. Ontological reasoning is mainly supported by two common representations of semantic web languages: RDF(S) and OWL. SWRL is one of the available solutions to add rules in OWL (see section 2.3.1). This technique is applied in [WDTP04].

Table 2.2 – Data-driven and Knowledge-driven approaches for context handling.

Approaches	Advantages	Disadvantages
Knowledge-driven	• Semantic reasoning • Explicit representation of context • Strong validation • Application independent • Allows sharing • Strong support by standards	• Representation can be complex • Resource intensive (processing, storage, time) • Data model in a compatible format (OWL, RDF)
Data-driven	• Fairly accurate • Different models are available	• Training data required • Resource intensive (processing, storage, time) • Models can be complex • Difficult to capture existing knowledge

To design a specific application, the selection of the appropriate modelling and reasoning technique should be made carefully considering the requirements of the application.

The ontology-based approach is particularly suitable for context modeling due to its powerful representation capabilities and reasoning methods for handling heterogeneous sensor data [SLP04]. However, this approach needs expert knowledge to build the initial model. Contextual knowledge is interpreted and evaluated by means of ontology reasoning. This allows to infer new and more complex context characteristics.

Table 2.2 presents a summary of the advantages and disadvantages of the data-driven approaches and knowledge driven approaches for context handling discussed in this section. Due to the requirements of condition monitoring in Industry 4.0, we choose to use the ontology-based approach for context modeling.

2.6 Conclusion

In this Chapter, we provide an overview of the theoretical foundations of smart systems. It starts with an introduction of the classic and the KREM architecture for the development of this kind of system. We frame our approach through the KREM architecture, that has four components. In the *Knowledge* component section, the concept of ontology is introduced as well as the key concepts of Semantic Web and the associated technologies used in this thesis. In the *Rules* component description, we present the notion of reasoning in static and dynamic scenarios. SWRL is a powerful language for offline reasoning, which has become the de facto standard rule language in the Semantic Web. Stream reasoning, that combines continuous processing of data streams with reasoning with rich background knowledge, can be used to deal with dynamic domains such as Industry 4.0. The *Experience* component section presents Cased-based Reasoning and Set of Experience Knowledge Structures, which are two complementary approaches for the capitalization of experience. This thesis does not deal with this component of KREM. In the *Meta-knowledge* component section, the definition of context as well as the main approaches for context handling are discussed. Both context representation and context reasoning in the Industry 4.0 domain can be handled through semantic technologies. All the concepts

presented in this chapter provide a crucial background to our work.

In the next chapter, we review the existing approaches for condition monitoring in Industry 4.0. We pay special attention to those that use Semantic Web Technologies to facilitate condition monitoring tasks in industry.

Chapter 3

Existing approaches for Condition Monitoring in Industry 4.0

Contents

Monitoring is a crucial activity in industry. Any unexpected machine failure can degrade or even interrupts the manufacturing processes of a company. Therefore, it is fundamental to develop a well-implemented and efficient monitoring strategy to prevent unplanned stoppages of production, improve reliability and reduce operating costs.

This chapter describes existing approaches for condition monitoring in Industry 4.0. We focus particularly on the ontological models and their rule-based extensions that are relevant to condition monitoring. The remainder of the chapter is organized as follows: in section 3.1, we introduce the key elements of Industry 4.0. In section 3.2, the main existing maintenance strategies are described. This section also reviews related approaches for fault detection and diagnosis in condition monitoring systems. These approaches can be categorized into data-driven approaches and knowledge-based approaches. In section 3.3 some of the most relevant ontologies to model the manufacturing domain are reviewed. We analyse them highlighting their advantages as well as their drawbacks according to the requirements presented in the introduction of this thesis. In section 3.4, we describe a number of widely accepted and reusable ontologies to model context knowledge across different applications. In section 3.5 we present the approaches for handling the representation of knowledge that change over time. Finally, section 3.6 summarizes the chapter.

3.1 Key elements of Industry 4.0

Industry 4.0 follows and extends the notion of automation technology that is introduced in the third industrial revolution by applying a set of technologies for automatic data exchange and processing in manufacturing factories [WSJ17].

Cyber-Physical Systems (CPS), the Internet of Things (IoT), Cloud Computing and Big Data analysis techniques are key elements of Industry 4.0. They allow the automatic interconnection and data exchange among manufacturing entities. Collecting and analyzing this data can improve the productivity and reliability of production systems [LBK15].

Industry 4.0 aims at transforming traditional manufacturing factories into ”smart factories” by equipping manufacturing with sensors, actuators and autonomous systems [DGP19]. In this way, manufacturing machines can achieve high levels of self-optimization and automation.

The key elements of Industry 4.0 are detailed below.

Internet of Things (IoT). IoT refers to the network of physical objects that are embedded with sensors and software for the purpose of connecting and exchanging data with other devices and systems [XYWV12]. The combined use of IoT and industrial applications gave rise to the Industrial Internet of Things (IIoT) [LFK+14]. In IIoT, industrial objects are equipped with electronics and smart devices. They provide identification and communication capabilities for industrial objects. This facilitates access to industrial data thus enabling monitoring of the manufacturing entities in a factory [WSJ17, CBS+17].

Cyber-Physical Systems (CPS). CPS is the basic technology platform for IoT and IIoT, therefore the main enabler to connect physical machines. CPS integrates elements of the physical world (such as sensors and machines) with software, providing abstractions and analysis techniques, thus transferring the physical world into the virtual one. CPS have been widely adopted in a variety of production processes.
Lee et al. proposed the 5C architecture for utilizing CPS in manufacturing process [LBK15]. It consists of 5 levels, namely the Connection, Conversion, Cyber, Cognition, and Configuration levels. This architecture covers the steps from acquiring industrial data until generating meaningful information and decision-making for the end system. Below we briefly describe each of the levels.

- the *Connection* level acquires data from machines in an accurate and reliable way;
- the *Conversion* level derives useful information from data collected by the previous level;
- the *Cyber* level applies specific analytics to extract additional information and provides a better understanding of the state of physical assets in physical space;
- the *Cognition* level provides a proper presentation of the acquired knowledge to expert users to support decision making tasks. It transfers the acquired knowledge to the users.
- the *Configuration* level provides feedback from cyber space to physical space. Corrective and preventive actions are triggered in the physical space, based in the feedback.

To summarize, a CPS provides (1) the connectivity that allows real-time data collection and communication between the physical space and the cyber space; and (2) intelligent data processing and analytics that are capable of supporting decision making tasks in the cyber space.

Cloud Computing. Cloud Computing allows the storage of large amounts of data. This is important to store the data generated during a whole production process, considering that the machines and sensors produce more data than an operator. Likewise, cloud computing reduces investment in technological resources, allowing the storage space and processing capacity, which provides flexibility and adaptability [TS16].

Big Data analysis technologies Big Data technologies allow to analyze enormous volume of information that is generated in an Industry 4.0 production ecosystem. These technologies enables to assess the state and operation of the machines involved in manufacturing processes [YK15]. The analysis of data for condition monitoring reduces inefficiencies and costs, anticipating equipment failures and allowing better responses to emergent situations caused by different factors such as high humidity, high temperature

or exposure to gases. In summary, big data technologies allow to extract valuable information based on the intelligent use of data.

The joint use of IoT, CPS, Cloud Computing and Big Data technologies can help production systems to self-access their health conditions to perform smart decisions to avoid potential failures [ZTL15].

3.2 Maintenance strategies in the industrial context

Maintenance strategies in the industrial domain generally fall into one of the three following categories, each with its issues and benefits:

- Reactive Maintenance (RM) [Mob02, Swa01] is a run-to-failure maintenance management method. The maintenance action for repairing equipment is performed only when the equipment has broken down. A company using run-to-failure management does not spend any money on maintenance until a machine or system fails to operate. However, the cost of repairing or replacing a component would potentially be more than the production value received by running it to failure. Moreover, as components begin to vibrate, overheat and break, additional equipment damage can occur, potentially resulting in further costly repairs.

- Preventive Maintenance (PM) [Mob02, WTL+17, Ger13] schedules regular maintenance activities on specific equipment to reduce the likelihood of failures. The maintenance is executed even when the machine is still working and under normal operation so that the unexpected breakdowns with the associated downtime and costs are avoided. Almost all Preventive maintenance management programs are time-driven [Mob02, AK12]. In other word, maintenance activities are based on elapsed time. The general process of Preventive maintenance can be presented in two steps: The first step is to statistically investigate the failure characteristics of the equipment based on the set of time series data collected. The second step is to decide the optimal maintenance policies that maximize the system reliability and availability at the lowest maintenance costs. Preventive maintenance reduces the repair costs and unplanned downtime, but might results in unnecessary repairs if it is conducted to soon or failures if it conducted too late.

- Predictive Maintenance (PdM), a.k.a. Condition-based Maintenance [WDD94], aims to predict when the equipment is likely to fail and decide which maintenance activity should be performed such that a good trade-off between maintenance frequency and cost can be achieved. Predictive maintenance uses the actual operating condition of systems and components to optimize the production. The predictive analysis is based on data collected from sensors connected to machines and tools, such as vibration data, ultrasonic data, operation availability, etc. The model processes the information through predictive algorithms, discovers trends and identifies when the equipment should be repaired. Rather than running a piece of equipment or

a component to failure, or replacing it when it still has useful life, predictive maintenance helps companies to optimize their strategies by conducting maintenance activities only when completely necessary. However, compared with Reactive maintenance and Preventive maintenance, the cost of the condition monitoring devices (e.g., sensors) needed for Predictive maintenance is often high. Another issue is that the Predictive maintenance system is becoming more and more complex due to data collection, data analysis and decision making.

Figure 3.1 summarizes the maintenance plans of each of the three possible types of maintenance. Reactive Maintenance has the lowest prevention cost due to using run-to-failure management. Preventive Maintenance has the lowest repair cost due to well scheduled downtime while Predictive Maintenance can achieve the best trade-off between repair cost and prevention cost. Ideally, Predictive maintenance allows the maintenance frequency to be as low as possible to prevent unplanned Reactive maintenance, without incurring costs associated with doing too much Preventive maintenance.

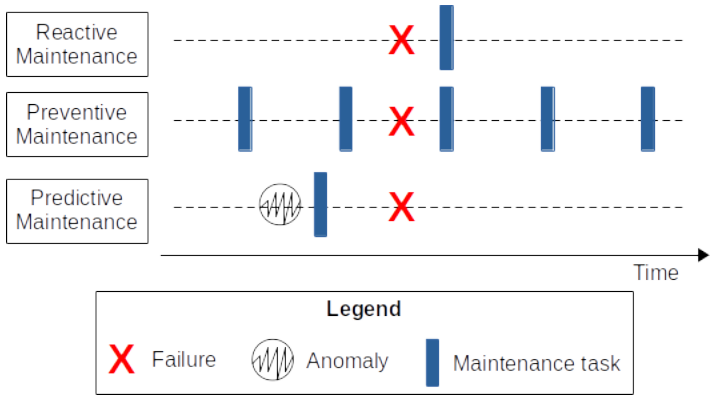

Figure 3.1 – Maintenance strategies in industry(RM - PM - PdM)

The use of the technologies associated to Industry 4.0 makes predictive maintenance more efficient and flexible [CBL+18, Wan16]. This is called Predictive Maintenance 4.0. It employs advanced and online analysis of collected data for the early detection of the occurrence of possible machine failures, and supports technicians during the maintenance interventions by providing decision support. In other words, Predictive maintenance 4.0 applies real-time condition monitoring to machines, and alerts are given based on pre-established rules or critical levels. Over the last decades, a considerable amount of research efforts has been undertaken to address the development of different models for industrial predictive maintenance, using the key elements of Industry 4.0.

In the following, we describe the two main approaches to implement Predictive maintenance: data-driven approaches and knowledge-based approaches.

3.2.1 Data-driven approaches to condition monitoring

With the development of big data related techniques and the ever-increasing availability of data, data-driven predictive maintenance is becoming more and more attractive. To extract useful knowledge and help to make appropriate decisions from huge amount of

data, machine learning and deep learning techniques have been regarded as powerful solutions. The models obtained by applying those techniques perform like a black box that learn the behavior of physical assets directly from their operation data [JGZ17]. They have the ability for feature learning, fault classification and fault prediction. Within a data-driven approach, knowledge about machines are extracted internally from machine operation data, instead of externally from domain experts. In the following, we present works that are representative of the domain.

In [SAB03], the authors present an ANN-based fault diagnosis for rolling element bearings using time-domain features. Five vibration signals from sensors are used to extract five time-domain features as the input of the designed ANN. The experimental results show that the accuracy can reach 62.5%-100%. Benkercha et al. [BM18] propose an approach based on the building of a Decision Tree to detect and diagnose the faults in grid connected photovoltaic system (GCPVS). The features used include temperature, irradiation and power ratio, and the class labels contains free fault, string fault, short circuit fault or line-line fault. Experimental results indicate that the diagnosis accuracy can reach 99.80%. In [SMZ14], Soualhi et al. apply the Hilbert-Huang transform (HHT)[GP07] to extract health indicator from vibration signals and utilized SVM to achieve fault classification of bearings. A method based on the evidential k-NN (EKNN) rule in the framework of evidence theory to achieve a condition monitoring and early warning in power plants is developed and presented in [CWHZ18].

Several surveys on different data-driven approaches for fault detection and diagnosis exist. Zhao et al. [ZYC+19] provide an overview of Deep Learning based machine health monitoring systems. All the systems presented are aimed at fault identification and classification. In addition, a series of survey papers focus on the fault diagnosis for a specific type of components, e.g., bearing, rotating machinery, wind turbines [BLDdO18]. In [ZZWH19], Zhang et al. systematically summarize the existing literature employing machine learning (ML) and data mining techniques for bearing fault diagnosis. Liu et al. [LYZC18] provide a comprehensive review of AI algorithms in rotating machinery fault diagnosis from the perspectives of theories and industrial applications.

The aforementioned works have given interesting contributions related to the field of fault detection and diagnosis, but most of them are equipment specific. They monitor the behavior of different properties of a machine. They are suitable and efficient when consuming data streams to detect abnormal patterns in the values of a machine's property (e.g. temperature, vibration, etc.). However, they have two drawbacks: (i) the need, in advance, for huge amount of annotated data for model training; and (ii) the lack of explicit model to explain decisions. This make it difficult to interpret the data and it also complicates the interoperability and re-usability of the models.

3.2.2 Knowledge-based approaches to condition monitoring

Many traditional condition monitoring systems and fault diagnosis systems make use of a priori expert knowledge and deductive reasoning processes [PDZ10].

Expert knowledge is exploited to build an ontology that formally describes concepts

and relationships that exist in the industrial domain. Different ontologies have been built for monitoring tasks such as predictive maintenance and health assessment systems [NB18]. As evoked in section 2.2.1, ontology-based modeling allows: (1) knowledge sharing between computational entities by having a common set of concepts, (2) logic inference by exploiting various existing logic reasoning mechanisms to deduce high-level concepts from raw data, and (3) knowledge reuse by reusing well-defined ontologies of different domains. Ontologies can be employed to represent different machinery systems and devices, and can be combined with various existing rule-based reasoning algorithms to achieve monitoring and fault diagnosis. This is done based upon the evaluation of online monitored data according to a set of rules which is pre-determined by expert knowledge. Usually, rules are expressed in the form: **IF** ⟨*antecedent*⟩ **THEN** ⟨*consequence*⟩. In this way, the consequence can make changes that affect the system that is being monitored, if the conditions in the antecedent are met.

In the following, we make a review on several knowledge-based approaches for condition monitoring.

Recently, Gul et al. [GC18] propose a condition monitoring based approach for risk assessment in a rail transportation system by using a fuzzy rule-based expert system. In [KMS+19], Kharlamov et al. design a rule-based language for equipment diagnosis in ontology mediated data integration scenarios such as industrial IoT. Its advantages include the easy formulation of diagnosis programs with hundreds of pieces of complex equipment. Berredjem et al. [BB18] propose a fuzzy expert system to localize bearing faults diagnosis as well as distributed faults. The fuzzy rules are automatically induced from numerical data using an improved range overlaps method.

Other approaches use data mining and machine learning methods to extract diagnosis knowledge or mine rules from databases in a smart system. These approaches include the works of Martinez et. al. that uses an artificial neural network based expert system for detecting the status of several components in agroindustrial machines using a single vibration signal [MGGGGRG15]; the works of Liu et. al. that use support vector machines and rule-based decision trees for fault diagnosis of water quality monitoring devices [LXL+18] and the works of Antomarioni et. al. which use association rule mining in maintenance [Gra20] to minimize the probability of breakages in an oil refinery [APP+19]. Although these approaches could be considered as data-driven, data mining technologies are here used to elicitate knowledge that will be exploited later.

Another rule-based model named Adaptive Neuro-Fuzzy Interference Systems (ANFIS) is applied for monitoring wind turbine SCADA (Supervisory Control and Data Acquisition) signals in [SS14, SSA13]. In order to obtain turbine condition statements, the authors implement rules given by an expert who is familiar with the behavior of the turbine, typical faults and their root causes. There are two types of rules: generic rules used to highlight anomalies, and specific rules providing specific condition or potential root cause. Schmidt et al. [SWG17] develop a semantic framework, using an ontology-based approach for data aggregation, to support cloud-enabled maintenance of manufacturing assets. Xu et al. [XLC+18] propose an ontology-based fault diagnosis model to integrate, share and reuse fault diagnosis knowledge for loaders. A loader is a self-propelled heavy

machinery having for main function to push and lift (load) ground pieces.

Ontologies provide a way to integrate, share and reuse of domain knowledge, but other reasoning methods have to be integrated with ontologies to achieve predictive maintenance. Rule-based reasoning can be employed in order to achieve monitoring and diagnosis. These rules are built from expert knowledge, or are extracted from the analysis of large data sets, as mentioned above. However, knowledge-based approaches have difficulties in dealing with new (unknown by experts) faults and acquires complete knowledge to build a reliable set of rules.

3.3 Ontological models for the manufacturing domain

We gave a review of the existing knowledge-based predictive maintenance systems in the previous section. In this section, we give an overview on the development and usage of ontologies in the manufacturing domain. These ontologies were defined with distinct purposes and, therefore, describe different types of information related to that area.

The development of a knowledge-based predictive maintenance system requires domain knowledge about manufacturing processes to be represented in a formal way, thus making this knowledge usable by the system. To achieve this goal, semantic technologies, especially ontologies with their rule-based extensions, have shown promising capabilities for formalizing knowledge about predictive tasks in various domains [PC09, SWWP10].

A typical manufacturing system can be characterized according to three main notions: Product, Process and Resource [MD03].

In [PDT12], authors have developed a product ontology, named ONTO-PDM. It provides a semantic layer to business, design and manufacturing product-related information. ONTO-PDM harmonizes the product-related knowledge and standards, and this harmonization has shown positive results in solving the interoperability problems among different enterprises and applications. Other works in this direction are: OntoSTEP (ONTOlogy of Standard for the Exchange of Product model data) [BKR+12], which allows the description of product information mainly related to geometry; and MCCO (Manufacturing Core Concepts Ontology) [UYC+11] that focuses on interoperability across the production and design domains of product life-cycle. It provides some core classes in categories such as `ManufacturingProcess`, `ManufacturingFacility`, `ManufacturingResource` and `Feature`.

Like ONTO-PDM, the Process Specification Language (PSL) ontology [Grü09] is another notable development work in the area of product information modelling. The PSL ontology covers several domains such as manufacturing, engineering and business processes. In this section we only focus on the manufacturing domain. Even though it mainly focuses on fundamental concepts for representing manufacturing processes, the ontology also provides a basis for the formal description of elements and entities that constitute a Process. The foundational elements of the core of the PSL ontology are four primitive classes (`Activity`, `Activity-occurrence`, `Timepoint`, `Object`), three primitive relations (`participates-in`, `before`, `occurrence-of`) and two primitive functions

(`beginof`, `endof`). From the manufacturing product point of view, the notion Product could be deemed as a sub-concept under the core concept `Object` in the ontology. The PSL ontology provides a robust semantic foundation for modelling manufacturing product information. Furthermore, as indicated by the name, the PSL ontology is a powerful approach for the representation of manufacturing processes.

Other ontologies have been developed to enhance the performance of product information modeling in the manufacturing domain. As the PSL ontology, these ontologies include the notions of resource and process. Among them, the MASON [LSDS06] and the ADACOR [BL07] ontological models are considered as pertinent. The MASON ontology presents a detailed conceptualization mainly focused on the products because it was used for cost estimation of the production of mechanical parts. MASON is an upper ontology for representing what the authors consider the core concepts of the manufacturing domain: products, processes and resources. As a result, the main classes of MASON are `Entity`, for specifying the product; `Operation`, for describing all processes linked to manufacturing; and `Resource`, for representing concepts regarding machines, tools, human resources and geographic resources. The ADACOR Ontology, developed as a part of the ADACOR architecture, provides a refined conceptualization to model operations, production plans and work orders regarding customer orders. The ontology is not openly available. Hence to be applied outside of the ADACOR project, the ontology has to be rebuilt with the information provided in the documentation. The SIMPM (Semantically Integrated Manufacturing Planning Model) ontology [ŠS19] is an upper ontology that is also focused on the representation of production plans. It models the fundamental constraints of manufacturing process planning: manufacturing activities and resources, time and aggregation.

Besides the ontologies presented above, there exist other ontologies that are more focused on modeling manufacturing processes and resources. The P-PSO (Politecnico di Milano Production Systems) ontology [GF12] considers three aspects in the manufacturing domain: the physical aspect (the material definition of the system), the technological aspect (the operational view of the system) and the control aspect (the management activities), for information exchange, design, control, simulation and other applications. Thus, its main classes are `Component`, `Operation` and `Controller`, which model the aforementioned three aspects, as well as part, `Operator` and `Subsystem`. The MSDL (Manufacturing Service Description Language) ontology [AD06] allows to describe manufacturing services. More precisely, a `ManufacturingService` is seen as a service that is provided by a `Supplier` and that has some `ManufacturingCapability`, which is enabled by some `ManufacturingResource` and delivered by some `ManufacturingProcess`. Another ontology with similar purposes to MSDL is MaRCO (Manufacturing Resource Capability Ontology) [JSHL19], that defines capabilities of manufacturing resources. Its main class is `Capability`, which is specialized to cover both simple capabilities (e.g. Fixturing, SpinningTool) and combined capabilities (those that require a combination of two or more simple capabilities, e.g. PickAndPlace, which requires Finger-Grasping or Vacuum Grasping, Moving and Releasing). Another interesting work is the one presented in [CZX+16]. It proposes a base ontology to represent a production line. The base ontology

integrates four ontologies: (1) the device ontology, with concepts such as `Machine`; (2) the process ontology, with a taxonomy of the different `Operations` performed by the technical equipment; (3) the parameter ontology, with concepts such as `QualityofService`; (4) the product ontology, with the product information.

The presented ontologies define terms and relations for modeling the manufacturing domain, such as machine, process and the possible relations between them. The ontologies previously described do not allow representing the sensors attached to the company's resources and the properties they measure which are also fundamental for anomaly detection on production lines and reflect the correctness of mechanical system conditions. In addition, they do not allow the representation of the context of the machines and processes. This is important to determine whether the observations collected by sensors represent abnormal behavior or not.

As mentioned in section 3.1, IoT is a key element of Industry 4.0. Nowadays, it is possible to use sensor networks to detect and identify a wide variety of observations, from simple phenomena to complex events and situations. However, the lack of integration between these sensor networks often isolates important or relevant data. To deal with this issue, the Semantic Sensor Web (SSW) approach provides tools that allow the integration of data from multiple data sources [SHS08]. It introduces semantic annotations for describing: (1) the data produced by the sensors, introducing spatial, temporal, or situation/context semantics; and (2) the sensors and the sensor networks that provide such data. Furthermore, there are also works on defining suitable ontologies for data and sensors to enable both the integration of data from multiple sensor networks and external sources, and reasoning on such data. As an example, the W3C Semantic Sensor Network Incubator Group [HJC+19] developed an ontology to describe sensors and sensor networks, the Semantic Sensor Network Ontology (SSN). Another work in this direction is SAREF4INMA [dRFID+19], that pursues interoperability with industry standards. The SSN ontology is further detailed in section 5.1.3, which describes its use in this thesis.

To deal with context modelling in the industrial domain, the following section discusses the main ontologies in the literature for context modelling.

3.4 Ontological models for context modeling

A context modeling approach must satisfy two types of requirements: (1) those concerning the ontology design principles, mentioned in section 2.2.1, and (2) those concerning the application domain, in our case Industry 4.0.

The requirements from the application domain focuses on specific tasks from the industrial domain. The model must support the description of industrial entities (users, devices, sensors, etc) and the integration of entities which are not known at design-time. Besides, the static information about an entity, it is also necessary to have some dynamic information associated to the processes the entity is involved in. Data collected from various sensors deployed in a factory need to be explicitly represented in order to describe that process. Also in the application domain, the representation of (a) temporal relations

CoBrA Ontology Classes		CoBrA Ontology Properties	
"Place" Related	Agents' Location Context	"Place" Related	Agent's Location Context
Place AtomicPlace CompoundPlace Campus Building AtomicPlaceInBuilding AtomicPlaceNotInBuilding Room Hallway Stairway OtherPlaceInBuilding Restroom Gender LadiesRoom MensRoom ParkingLot	ThingInBuilding SoftwareAgentInBuilding PersonInBuilding ThingNotInBuilding SoftwareAgentNotInBuilding PersonNotInBuilding	latitude longitude hasPrettyName isSpatiallySubsumedBy spatiallySubsumes accessRestricted-ToGender lotNumber	locatedIn locatedInAtomicPlace locatedInRoom locatedInRestroom locatedInParkingLot locatedinCompoundPlace locatedInBuilding locatedInCampus
"Agent" Related	Agent's Activity Context	"Agent" Related	Agent's Activity Context
Agent Person SoftwareAgent Role SpeakerRole AudienceRole IntentionalAction ActionFoundInPresntation	PresentationSchedule Event EventHappeningNow PresentationHappeningNow RoomHasPresentationHappeningNow ParticipantOfPresentation-HappeningNow SpeakerOfPresentationHappeningNow AudienceOfPresentationHappeningNow PersonFillsRoleInPresentation PersonFillsSpeakerRole PersonFillsAudienceRole	hasContactInformation hasFullName hasEmail hasHomePage hasAgentAddress fillsRole isFilledBy intendsToPerform desiresSomeone-ToAchieve	participatesIn startTime endTime Location hasEvent hasEventHappeningNow invitedSpeaker expectedAudience presentationTitle presentationAbstract presentation eventDescription eventSchedule

Figure 3.2 – The CoBrA Ontology [CFJ03]

among processes and of (b) spatial relations between entities and locations in the factory is required. These relations are used to identify different situations of interest that represent specific states of an industrial process or an industrial entity. This involves handling information that evolves in time, such as the changes of situation a machine can go through or the changes in the values of a property according to the different decisions made.

There exist a number of widely accepted and reusable ontologies to model context knowledge across different applications, that are presented below.

CoBrA-Ont is an ontology, proposed in [CFJ03], to describe the attributes and relationships related to people, location and activities. It is specially focused on smart meeting places. Its main concepts and relations are shown in Figure 3.2. Although CoBrA-Ont offers concepts to represent the context according to Dey's definition of context [DA00]. It does not offer the spatial and temporal reasoning that is necessary for Industry 4.0. As an extension of the previous work, the authors of CoBrA-Ont have presented SOUPA (Standard Ontology for Ubiquitous and Pervasive Applications) [CPFJ04] shown in Figure 3.3. It is a general ontology for pervasive applications, with two parts: a core ontology and user extensions. The core ontology contains the following concepts: person, agent, belief-desire-intention (BDI), action, policy, time, space and event. Users can create their own ontology as an extension of SOUPA for specific pervasive applications. SOUPA reuses terms in different domain ontologies, such as FOAF[1], DAML-Time[2], OpenCyc [MWCD06] and CoBrA-Ont [CFJ03]. SOUPA is a very general and complex ontology, lacking features needed for our aims, such as spatio-temporal reasoning.

[1]http://xmlns.com/foaf/spec/

[2]https://www.cs.rochester.edu/ ferguson/daml/

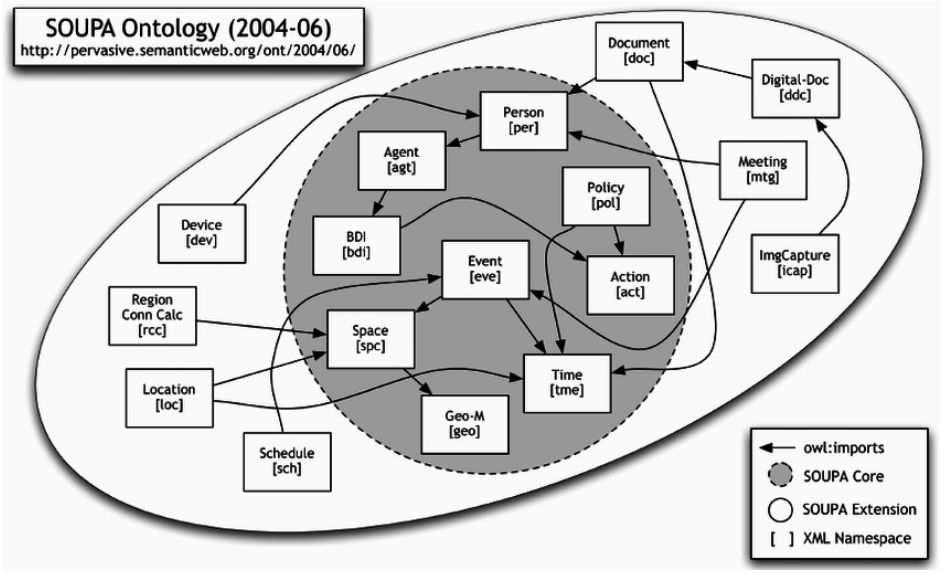

Figure 3.3 – The SOUPA ontology [CPFJ04].

The CONtext ONtology (CONON) [WDTP04] is designed to model context in pervasive computing environments. The ontology is shown in Figure 3.4. CONON defines generic concepts regarding context and provides extensibility for adding domain specific concepts. It can be separated into two parts: an upper and several domain specific ontologies. The upper ontology is a high-level ontology which captures general features of basic contextual entities, such as CompEntity, Location, Person and Activity. The domain specific ontologies define concepts and their features in different sub-domains. In CONON, context is classified into direct and indirect context. The former includes facts that can be either sensed by the physical objects or defined by a human-being, while the latter is obtained by interpreting direct context through aggregation and reasoning processes. CONON also contains rules to reason about the context. CONON does not provide a way to represent temporal data and the evolution of concepts in time. As SOUPA, this ontology is very general, which makes it difficult to apply it directly to a particular domain, such as industry.

SAWA (Situation Awareness Assistant) [MKB+05], from the military domain, uses an upper ontology for situation awareness introduced in [MKB+06]. This ontology can be adapted to handle domain-specific situations by extending the core vocabulary. Situations are represented as objects, since the situation concept is derived from the object concept. SAWA has the same limitations as CONON. Qualitative approaches to the representation of time and space are not provided. Furthermore, the `PhysicalEntity` concept, which is derived from `Object`, is incomplete, as a definition for non-physical objects is missing.

The situation ontology in [YL06], from the field of pervasive computing, incorporates situations as well as contexts. Situations are classified into atomic and composite situations; they are not composed of objects. They are directly (atomic) or indirectly (composite) represented by contexts. This ontology does not provide any explicit relations, roles, or events; only the situation concept is present. Furthermore, space, time and situation

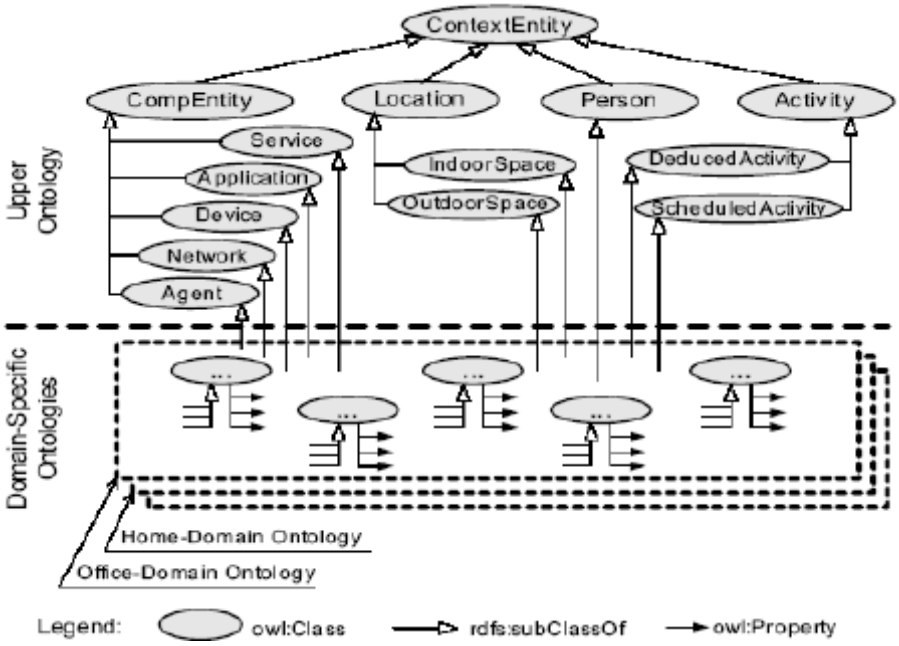

Figure 3.4 – CONON Ontology [WDTP04]

types, are not represented in the ontology. Also, the possibility of representing situations as objects is very complex.

Additionally, there are several context-aware frameworks that use ontologies as underlying context models in the industrial domain.

In [UPML15], an ontology-based context model for real-time decision making is proposed to optimise the Key Performance Indicators[3] of flexible manufacturing systems. In [ZON15], the authors proposed an authoring system to create context-aware augmented reality (AR) applications for the maintenance of a system that uses an ontology for context modelling. In [LL13], a context-aware industrial monitoring system is presented that provides information to users at the right time in the right modality. In [AMX+16], the authors present a context-aware information distribution system to support users in an industrial environment. The system aims at using data collected from sensors located at a shop-floor in order to increase the visibility of shop-floor processes by providing the right information, to the right people, at the right time.

In all the works mentioned in the previous paragraph, there is no notion of situation that can be treated as an object. Instead, they represent events which can lead to a waterfall of events, *i.e.* these events can be caused as a consequence of one another. This does not facilitate a global interpretation of each event considering the others. In addition, they do not provide an explicit description of the sensors and observations.

All the discussed ontologies provide common knowledge concepts and statements, which facilitate knowledge sharing and interoperability. However, they do not meet all the requirements mentioned at the beginning of this section. Table 3.1 shows the requirements covered and those not covered by the different ontologies described above. Except for SOUPA, they provide no or little support for spatio-temporal modeling. But, exploring

[3] A Key Performance Indicator is a measurable value that demonstrates how effectively a company is achieving key business objectives.

the supported top-level concepts presented by SOUPA, they are not general enough to describe industrial entities in a clear and precise way. Furthermore, situations of interest are not explicitly anchored by SOUPA.

There is a need to put together concepts of these ontologies, concepts about sensors and the observations made by them, concepts about situations of interest, concepts about time and space, as well as the relationships between these concepts to aspire to build an ontology that allows the representation of the context in the domain of Industry 4.0.

Some of the concepts and relations of the ontologies previously presented are reused in this thesis and are explained in Chapter 5.

Table 3.1 – Comparison of context ontologies.

Ontologies	**Situation as objects**	**Situation types**	**Space and Time**	**Change over time**
CoBra-Ont	-	-	✓	-
SOUPA	-	-	✓	-
ConOn	-	-	-	-
SAWA Ontology	✓	-	-	-
Situation Ontology	✓	-	-	-

The presented ontologies do not provide a way to capture the dynamic changes and the evolution of concepts in time, such as the different situations a machine can go through or the changes in the values of a property according to the different decisions made. This is a key point to deal with the dynamics of manufacturing processes as mentioned in the introduction to this thesis. In the following section, different approaches to deal with this issue are discussed.

3.5 Modeling knowledge that evolves in time

Representing knowledge that evolves in time through ontologies is a challenging problem. Temporal concepts can be represented by an ontology called OWL-Time [CL17]. It provides rich descriptions of temporal intervals, instants, durations, and calendar terms. However, it cannot specify how these concepts can be used to represent properties of objects changing in time and it does not propose inference rules to automatically infer new temporal data.

Ontology languages such as OWL and RDF are all based on binary relations that complicate representation of temporal properties, since a property holding for a specific time instant or interval is a relation involving three objects (an object, a subject and a time instant or interval). Binary relations can simply represent the relation between two instances without any temporal information. Dynamic features representation calls for mechanisms that allow uniform representation of the notions of time (and of properties varying in time) within a single ontology. Existing methods for achieving this include, among others, Temporal Description Logics (TDLs) [AF01], Concrete Domains [Lut03], Temporal RDF [GHV05], Named Graphs [TB09], Versioning [KF01], N-ary relations [NA06]

and the 4D-fluents (or perdurantist) approach [WF06]. All result in more complex ontologies compared with their static counterparts where relations do not change in time.

Temporal Description Logics (TDLs) [AF01] extend standard description logics (DLs) with additional constructs such as *"always in the past"*, *"eventually in the future"*. TDLs offer additional expressive capabilities over non temporal DLs and retain decidability (with an appropriate selection of allowed constructs) but they require extending the OWL syntax and semantics with the additional temporal constructs. Unfortunately, it is not possible to represent that a predicate holds in a particular time interval of time but only over past and future times.

Versioning [KF01] suggests that the ontology has different versions. When a change takes place, a new version is created. Versioning has two main disadvantages: (i) changes even on single attributes require that a new version of the ontology is created, leading to information redundancy and (ii) searching for events occurred at time instances or during time intervals requires exhaustive searches in multiple versions of the ontology.

Using an improved form of reification, the N-ary relations approach [NA06] suggests representing a n-ary relation as two properties each related with a new object (rather than as the object of a property).

The 4D-fluents approach [WF06] shows how temporal information and the evolution of temporal concepts can be represented in OWL. Concepts in time are represented as 4-dimensional objects with the 4th dimension being the time (timeslices). Time instances and time intervals are represented as instances of a TimeInterval class, which in turn is related with concepts varying in time. Changes occur on the properties of the temporal part of the ontology keeping the entities of the static part unchanged. The 4D-fluents approach suffers from proliferation of objects since it introduces two additional objects for each temporal relation (instead of one in the case of N-ary relations). The N-ary relations approach requires only one additional object for every temporal relation, maintains property semantics but, compared to the 4D-fluents approach, it suffers from data redundancy in the case of inverse and symmetric properties (e.g., the inverse of a relation is added explicitly twice instead of once as in 4D-fluents). In the case of transitive properties additional triples are introduced as well.

The Named Graphs approach [TB09] represents the temporal context of a property by the inclusion of a triple representing the property in a named graph (*i.e.* a subgraph into the RDF graph of the ontology specified by a distinct name). The main RDF graph contains definitions of interval beginning and ending bounds for each named graph, thus a property is stored in a named graph with beginning and ending bounds corresponding to the time interval that the property holds. Named graphs are not part of the OWL specification (*i.e.* there are not OWL constructs translated into named graphs) and they are not supported by OWL reasoners [BP11].

Concrete Domains [Lut03] introduce datatypes and operators based on an underlying domain. This approach requires extending OWL or RDFS. TOWL [FMK10] is an approach combining the concrete domains and 4D-fluents approaches. However, it does not support qualitative temporal relations. Furthermore, it is not compatible with existing OWL editing, reasoning and querying tools.

Table 3.2 – Comparison of some approaches for representing temporal data in ontology.

Approaches	Reasoning	Quantitative data	Qualitative data	Advantages (+) Disadvantages (-)
TDLs	Yes	Yes	Yes	(−) Require extending OWL or RDF(S). (+) Retain decidability. (+) Do not suffer data redundancy.
Concrete Domains	Yes	Yes	Yes	(−) Require extending OWL or RDF(S).
Versioning	No	Yes	Yes	(−) Suffers from data redundancy. (−) Searching in all created versions.
N-ary relations	Yes	Yes	Yes	(−) Suffers from data redundancy.
4D-fluents	Yes	Yes	Yes	(−) Suffers from data redundancy. (+) Philosophical support.
Named Graphs	No	Yes	No	(−) Do not offer reasoning support.
Temporal RDF	No	Yes	No	(−) Require extending OWL or RDF(S). (−) Less expressiveness than OWL.

Temporal RDF [GHV05] proposes extending RDF by the annotation of properties with data about the time interval they hold on. It uses only RDF triples and requires extending the RDF syntax. Therefore, it does not have all the expressiveness of OWL. For instance, it is not possible to employ qualitative relations. A number of temporal representations are based on this approach.

Table 3.2 compares the approaches previously discussed for representing knowledge that evolves over time in ontologies. These approaches are compared in terms of the following semantic web standards: the compatibility with querying and reasoning support, and the supported temporal data (quantitative/qualitative). Most of the approaches allow only representing time intervals and associated qualitative relations. In other words, they are not intended to handle time points or qualitative relations between a time interval and a time point or even two time points. The temporal description logics, concrete domain, temporal RDF and named graphs approaches do not support OWL constructs. A basic design decision in our work is to choose an approach which relies on existing OWL constructs. Thus, we exclude the temporal description logics, the concrete domain, the named graphs and temporal RDF approaches. Based on this analysis, we have decided to use the N-Ary relation approach, that is further explained in section 5.1.3.

3.6 Conclusion

In this chapter, we introduce background knowledge about Industry 4.0, by detailing its four key elements: IoT, CPS, Cloud Computing, and Big Data analysis technologies. Then, we describe the existing maintenance strategies in the industrial domain emphasizing the predictive maintenance strategies for Industry 4.0. Data-driven approaches, based on annotated data, and knowledge-based approaches, based on expert knowledge, for condition monitoring are also discussed.

We have reviewed the existing ontologies and their rule-based extensions for knowledge-based condition monitoring systems. First, we reviewed the ontological models for the manufacturing domain. We analyse them highlighting their advantages and drawbacks according to the requirements for these works. Second, a survey about widely accepted and reusable ontologies to model context knowledge across different applications is presented. We analyzed them considering their application in the Industry 4.0 domain.

We summarize the domain coverage of the existing ontologies in Table 3.3. We evaluate the domain coverage and scopes of these knowledge models by examining whether the key concepts required for condition monitoring in Industry 4.0 exist and are formally described. We categorized these key concepts into five sub-domains: Manufacturing, Context, Change over time, IoT, and Monitoring and Diagnosis. For the Manufacturing sub-domain, the key concepts are Product, Process and Resource. For the Context sub-domain, the key concepts are Identity, Activity, Time, and Location. For the IoT sub-domain, Sensor and Observation are the key concepts. For the Monitoring and Diagnosis sub-domain, the key concepts are Situation and Cause. These concepts are the columns of Table 3.3, and the ontological models are the rows. If a check mark is placed in the table then the concept exists in the corresponding ontology. Otherwise, a cross mark is assigned.

After reviewing the ontological models mentioned previously, we recognize that none of them provides a satisfactory knowledge representation of the five required sub-domains. Some of these knowledge models focus on a narrow field, such as manufacturing resource planning and work orders, and they do not formalize context-related concepts, *e.g.*, Activity and Location. Also, none of the existing ontologies provides a way of representing knowledge that evolve in time. To perform smart condition monitoring, the knowledge base should contain not only the machine-interpretable knowledge for characterizing the manufacturing entities or processes which are being monitored but also the knowledge about abnormal situations that are associated to failures. This motivates us to develop a more expressive and complete ontological model that provides a rich representation of the domain knowledge in the fields of manufacturing considering the notion of context.

In addition, the ontological models of the smart systems reviewed in this chapter are static. As described in the introduction, machines perform manufacturing processes in different contexts and these contexts change over time. According to the context in which a manufacturing process is executed, the rules that manage the process can change. In order to represent these contexts and the fact that a machine is performing a process in these contexts, the semantic model needs to evolve in time to represent this changing knowledge.

Table 3.3 – Comparison of manufacturing ontologies and context ontologies.

	Manufacturing			Context				Change over time	IoT	Monitoring and Diagnosis	
Ontologies	**Product**	**Process**	**Resource**	**Identity**	**Activity**	**Time**	**Location**		**Sensor**	**Situation**	**Cause**
MASON	✓	✓	✓	✓	✓	✗	✗	✗	✗	✗	✗
ADACOR	✓	✓	✓	✓	✓	✗	✗	✗	✗	✗	✗
PSL	✗	✓	✓	✓	✓	✓	✗	✗	✗	✗	✗
MSDL	✗	✓	✓	✓	✓	✓	✓	✗	✗	✗	✗
SIMPM	✗	✓	✓	✓	✓	✗	✓	✗	✗	✗	✗
MaRCO	✗	✓	✓	✓	✓	✗	✓	✗	✗	✗	✗
ONTO-PDM	✓	✓	✓	✓	✓	✓	✓	✗	✗	✗	✗
P-PSO	✗	✓	✗	✓	✓	✓	✗	✗	✗	✗	✗
MCCO	✓	✓	✓	✓	✓	✓	✗	✗	✗	✗	✗
[CZX+16]	✓	✓	✓	✓	✓	✓	✓	✗	✗	✗	✗
OntoSTEP	✓	✗	✗	✓	✗	✗	✗	✗	✗	✗	✗
CoBrA-Ont	✗	✗	✗	✓	✓	✓	✓	✗	✗	✗	✗
SOUPA	✗	✗	✗	✓	✓	✓	✓	✗	✗	✗	✗
CONON	✗	✗	✗	✓	✓	✗	✓	✗	✗	✓	✓
SAWA	✗	✗	✗	✓	✓	✗	✗	✗	✗	✓	✓
Situation Ont.	✗	✗	✗	✓	✓	✗	✗	✗	✗	✓	✗

Part II

Contributions

Chapter 4

Proposed framework overview

Contents

This chapter provides a global overview of the proposed framework that encompasses the contributions of this thesis. The goal of the proposed framework is to address the evolution of semantic models in Industry 4.0, as explained in the introduction of this manuscript where the goals of this thesis were exposed.

The remainder of the chapter is structured as follows: Section 4.1 introduces the general architecture of Industry 4.0, according to the literature. This general architecture allows positioning each of our contributions, that are developed in the following chapters. In section 4.2, the components of the proposed framework are introduced as well as the interaction among them. Each of the components is detailed in the following chapters. Finally in section 4.3 some conclusions are presented.

4.1 General architecture for Industry 4.0

In [TBP15] the authors focus on the conceptual model, architecture and key elements needed for the support and enhancement of Industry 4.0 with smart systems. They emphasize that contextual knowledge, user experience and semantics involved in manufacturing and production processes should be taken into consideration if a company aims at improving the equipment and production systems safety and availability, to reduce the number of unnecessary maintenance tasks and to optimize production costs.

In order to interpret raw sensor readings combined with contextual knowledge for the detection of certain situations that may lead to failures, our proposal relies on an adaptation of this framework. This adaptation is depicted in Figure 4.1 and is described below.

- The **Sensors and Actuators** layer contains the sensors and actuators that are deployed in the machines and in their environment. A sensor is a device that detects and responds to some type of input from the physical environment. Some examples are pressure sensors, accelerometers for measuring vibration, acoustic sensors for detecting leaks and temperature detectors. The output is a signal that can be converted to human-readable format or transmitted electronically over a network for reading or further processing. An actuator is a component of a machine that is responsible for moving and controlling a mechanism or system, for example by opening a valve. It can be activated by electric voltage or hydraulic pressure, or even human power. When it is activated the actuator converts the signal's energy into mechanical movement.

- The **Communication** layer is in charge of pre-processing the data coming from the Sensors and Actuators layer through processes such as filling missing values, smoothing noisy data, etc. Furthermore, data can be normalized and/or aggregated as required. This layer is also able to divide and distribute information and must ensure security and anonymity where necessary.

- The **Semantic Enhanced CPS Representation** layer contains the reasoning processes and modules that allow the semantic enrichment of raw data coming from sensors. The Reasoner module uses First Order Logic (FOL) under the Open World

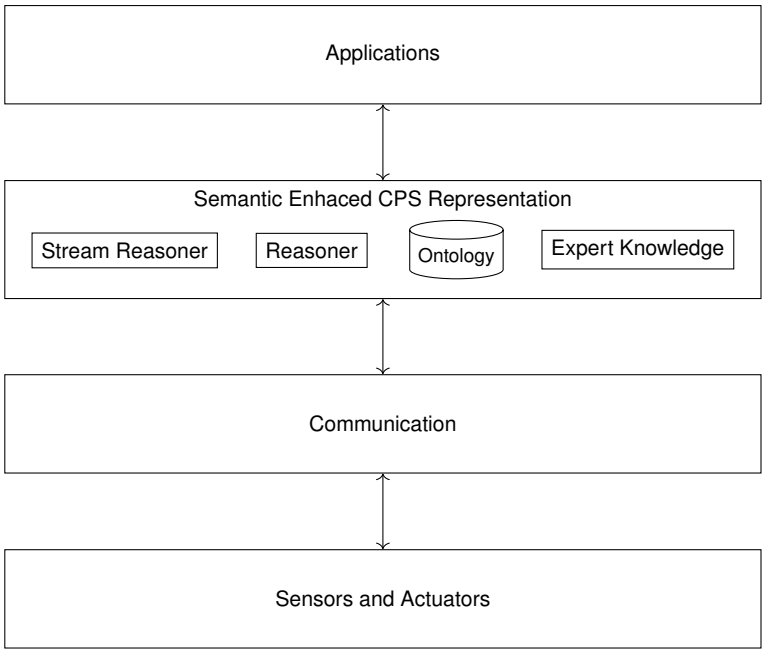

Figure 4.1 – The general architecture for Industry 4.0.

Assumption (OWA), as mentioned in section 2.2. In this layer, we propose to complement the Reasoner with a Stream Reasoner to process data in real-time. This module executes queries continuously over data streams coming from sensors. The Reasoner and the Stream Reasoner are not only fed with data streams but also with rules provided by domain experts. The Expert Knowledge module is in charge of storing these rules. It may also encompass data from other external sources.

- The **Application** layer comprises different applications that exploit the semantic enriched information, such as Business Information Systems (BIS) or Enterprise Resource Planning (ERP) software, that can also provide data to the previous layer.

The contributions presented in this thesis are framed in the Semantic Enhanced CPS Representation layer. This layer makes use of all its components to go from raw sensor readings combined with expert knowledge to the detection of relevant situations to support decision making tasks in the Application layer.

4.2 Overview of the proposed framework

The main components of our proposal are shown in Figure 4.2 using the classic Data Flow Diagram notation[1]. They are (1) the Monitoring component, (2) the Diagnosis component and (3) the Decision making component. Each of the components operates as a smart system to solve complex problems by reasoning, like an expert does. They all rely on a formal model to semantically enrich data representation and processing.

[1] https://en.wikipedia.org/wiki/Data-flow_diagram

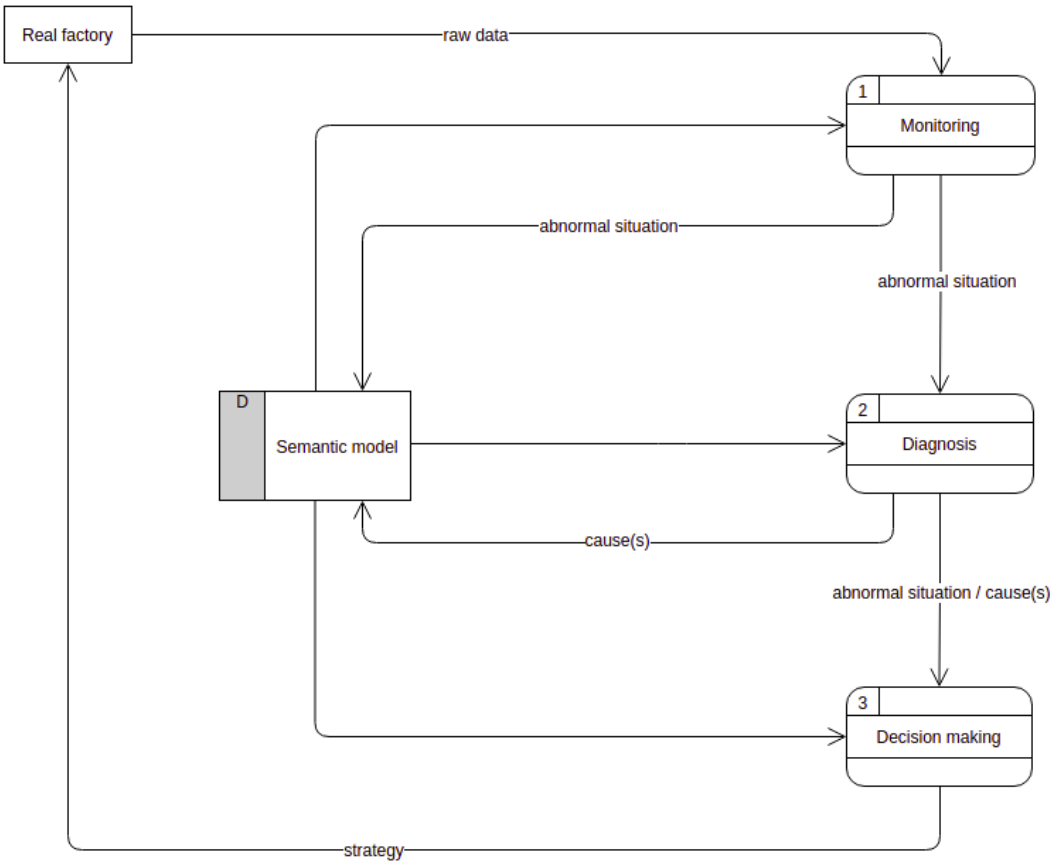

Figure 4.2 – The main components of the proposed framework.

4.2.1 The Semantic Model for Industry 4.0

The **semantic model** for Industry 4.0 is a foundation of our approach. It is a virtual representation of the real factory, as shown in Figure 4.3. It represents the elements of the real factory, such as machines, processes and sensors, with special emphasis on modeling the context of operation of these elements. Relevant situations representing abnormal behaviors as well as expert knowledge and rules handling process management are also represented in the model.

In a real factory, resources (such as machines and production lines) execute their tasks over time and under different situations. It is therefore necessary that the model represents in which situation(s) the resource is performing its tasks to allow a more informed decision making, since the actions to take and the rules that manage the manufacturing processes may vary according to the identified situation. In other words, the semantic model must evolve to be able to represent the situation(s) a certain resource is in during the execution of its tasks. This evolution is graphically shown in Figure 4.4. The semantic model is described in detail in Chapter 5.

4.2.2 The Monitoring component

The **Monitoring** component is responsible for collecting data from sensors in order to detect abnormal situations. It both uses the semantic model and modifies it. This compo-

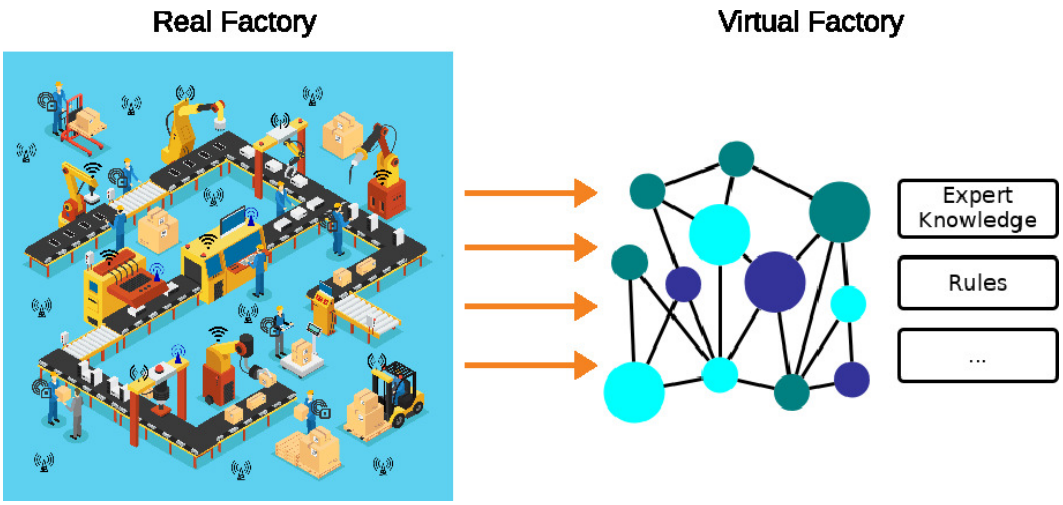

Figure 4.3 – Virtual representation of a real factory.

nent mainly uses the model for enriching data collected from sensors with contextual information. This allows to derive situations from context and sensor information of lower-level abstraction. Once an abnormal situation is detected, the situation together with the resources and the processes involved in it are added to the model; so that the model also represents which resources and processes are involved in the situation. This allows the model to continuously represent what is happening in the real process. The Monitoring component is explained in detail in Chapter 6.

4.2.3 The Diagnosis component

The **Diagnosis** component is responsible for determining the possible causes of the abnormal behavior. The crucial difference between monitoring and diagnosis relies on the nature of the output [SAA+00]. Monitoring observes a discrepancy between the expected and detected behavior without exploration of the cause or fault underlying it. However, in many domains such as Industry 4.0, monitoring and diagnosis are tightly coupled tasks: when monitoring observes an anomaly in the behavior, a diagnosis task is started, using the monitored information as input.

As the monitoring component, the Diagnosis component both uses the semantic model and modifies it. This component mainly uses the model to determine the cause(s) of a detected abnormal situations. The association between the causes and the abnormal sit-

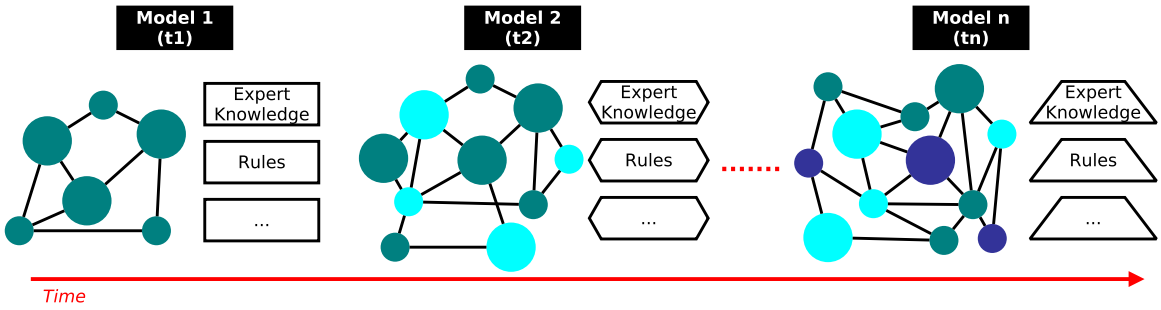

Figure 4.4 – Semantic model evolution over time.

uations are obtained from expert knowledge. Once the possible cause(s) associated to the detected abnormal situation are identified, these cause(s) are linked to the detected situation in the semantic model. This allows the model to represent which are the causes of an abnormal situation detected from the real manufacturing process. The Diagnosis component is explained in detail in Chapter 6 together with the Monitoring component since the tasks performed by these two components are tightly coupled, as previously evoked.

4.2.4 The Decision Making component

Once an abnormal situation an its causes have been detected, the **Decision making** component is responsible for determining and applying the adaptation strategies that are needed to improve the behavior of the production system. These strategies may include changes to the operation parameters of a machine or the launching of a maintenance task by the Application layer of Figure 4.1.

The choice of which action(s) to launch is made by the Decision making component. This component also uses the semantic model to determine which actions are possible to execute to improve the operating conditions.

Since detecting abnormal situations can trigger actions to adapt the process behavior, this change in the behavior of the process can lead to the generation of new situations.

These abnormal situations can have different levels of severity, and can be nested in different ways. To understand and manage the relations among situations, an efficient approach to organize them is needed. Therefore, Chapter 7 proposes a method to build a lattice to order the possible situations associated to a process, depending on the constraints they rely on. This lattice represents a road-map of all the situations that can be reached from a given one. This helps in decision support, by allowing the identification of the actions that can be taken to correct the abnormality. Therefore, the Decision making component uses both the semantic model and the lattice to determine which actions to trigger to correct the abnormal situation.

4.3 Conclusion

In this chapter we presented the general architecture of Industry 4.0, presented in the literature. This general architecture allows us to position our contributions.

The proposed framework to address the evolution of semantic models in Industry 4.0 is introduced. Its components are: (1) the Monitoring component, which is in charge of detecting abnormal behaviors of the production system; (2) the Diagnosis component, which identifies the possible cause(s) that generate an abnormal behavior; and (3) the Decision making component, which determines the adaptation strategies that are needed to correct abnormal behaviors of the production system. They all rely on a formal model to semantically enrich data representation and processing. Each of the components operates as an expert does. They can be seen as smart systems to solve complex problems by reasoning. Each one uses the semantic model to perform its functions and also makes the semantic model evolve by introducing the corresponding changes in it. In this way, the

tasks of each component are performed considering the updated semantic model which is a virtual representation of what happens in the real factory.

The following chapters explain each of the components and the semantic model in more detail.

Chapter 5

Semantic Model for Context Modeling in Industry 4.0

Contents

As mentioned in Chapter 2, the development of a smart system requires that the domain knowledge is represented in a formal way. To achieve this objective, ontologies have been widely used to formalize knowledge about the industrial domain [PC09, XLC+18]. However, as indicated in the sate of the art, most of these existing ontological models focus on a very specific field and sometimes lack a formal representation of context. This is why it is needed to develop a new ontological model to describe the ”Smart Factory” industrial domain with special emphasis on modelling the context of the components that constitute a real factory.

Therefore, this chapter proposes an ontology-based approach for context modeling in the industrial domain, suitable for Industry 4.0. The motivation for this ontological model lies on the fact that an isolated sensed value has to be interpreted in the context in which it is measured to become information that is relevant for decision making. Context is any information that can be used to characterize the situation of an entity. It is usually considered as a combination of geo-spatial data, environmental sensor inputs and service descriptions, among others. Context data is subject to constant change and can be highly heterogeneous. In manufacturing, a dynamic context model not only has to take into account the context of tools, machines, parts and products, but also information regarding the planning of manufacturing processes.

The remainder of the chapter is structured as follows: In section 5.1 the conceptual modules that compose the proposed ontological model are presented. In section 5.2 a complete description of the semantic alignment among ontologies is presented, based on the foundational concepts of DUL[1] (DOLCE Ultra Lite) for obtaining a rigorous conceptualization. DUL is an upper-level ontology providing general concepts and relations at a high abstraction level. The proposed approach is evaluated according to several ontology evaluation criteria such as structure, function and usability in Section 5.3. Finally, some concluding remarks are given in Section 5.4.

5.1 The proposed ontological model

This section describes the modules that constitute the proposed model, called Context Ontology for Industry 4.0 (COInd4). The goal of this semantic model is to represent the concepts and relations in the industrial domain to enable context representation and reasoning.

The notion of context is based on Dey’s definition of context [DA00], presented in section 2.5. Location, identity, time, and activity are the primary context types for characterizing the situation of a particular entity. These context types not only help to answer the classic "W-questions" (Where, who, when and what), but also allow to deduce other contextual information. For example, with the location of an entity, it is possible to apply spatial reasoning and determine which other objects or people are located near that entity.

For building the ontological model, the criteria for ontology designed (also called "on-

[1] http://ontologydesignpatterns.org/wiki/Ontology:DOLCE%2BDnS_Ultralite

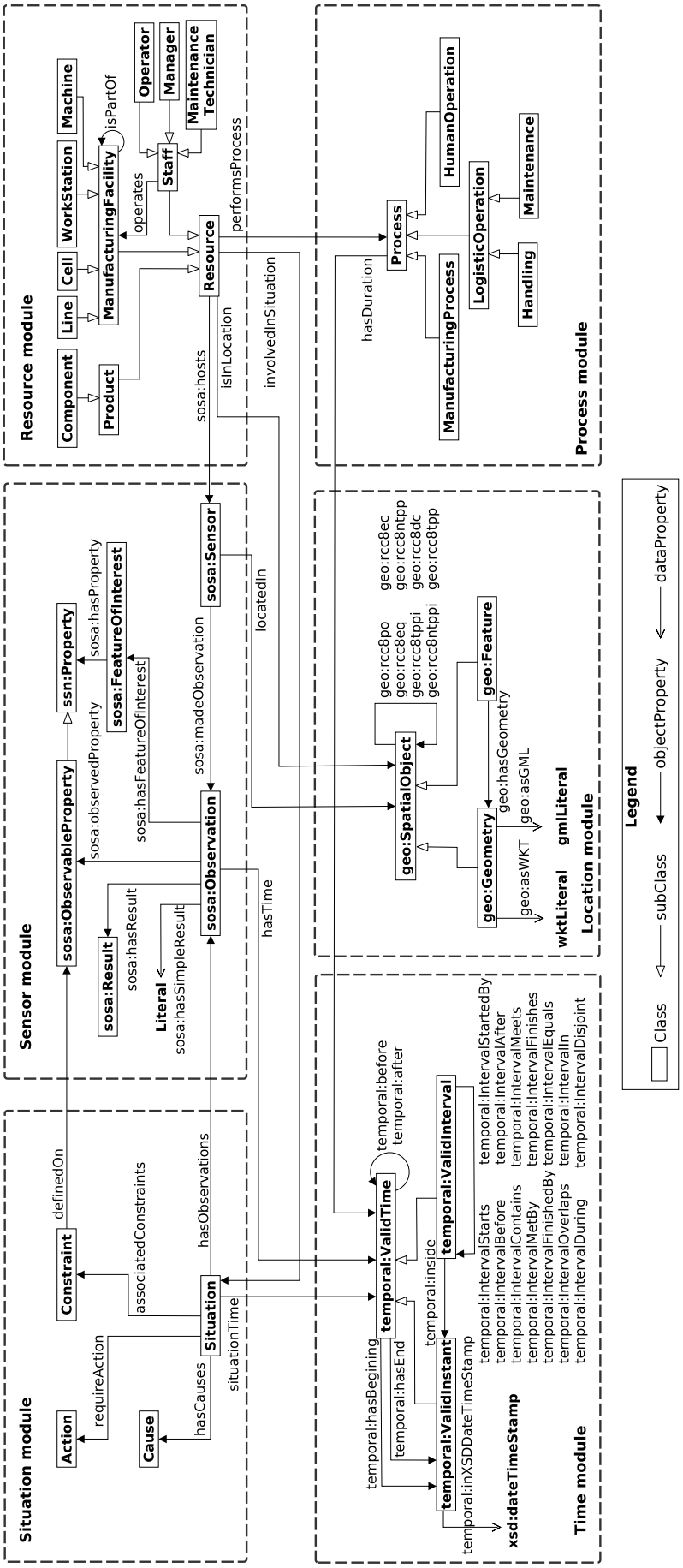

Figure 5.1 – The Context Ontology for Industry 4.0 (COInd4).

tological principles", described in section 2.2.1) were followed: clarity, coherence, interoperability, re-usability, extensibility and modularity. The ontology development process is based on the methodology proposed by [UG96], also presented in section 2.2.1. We first define the purpose and scope of the ontological model; then we capture domain knowledge, conceptualize and formalize the ontologies and align them with existing ontologies, vocabularies and standards. In line with these best practices, we followed an iterative and incremental development process, *i.e.* the ontological model was continuously improved by a better understanding of the domain. The ontology is written in OWL and is developed with two specific APIs for Java, OWLAPI [HB11] and the SWRLAPI [ODSM+08].

Several ontologies, described in Chapter 3, were considered and reused. These ontologies can be classified, according to [GPFLCGP04], as core ontologies, domain ontologies and application ontologies based on the subject of their conceptualization (see section 2.2.1).

An overview of the classes and properties from the proposed ontological model is shown in Figure 5.1. This ontological model is composed of six modules: (1) the *Resource* module aims to provide a comprehensive representation of manufacturing products and resources which are physical objects used for executing a set of operations during the manufacturing processes; (2) the *Process* module gives a formal representation of processes related to manufacturing, such as cutting, drilling, milling, among others; (3) the *Sensor* module considers knowledge about sensors and the observations generated by them; (4) the *Location* module provides concepts and relations that allow representing the abstraction of physical spatial places; (5) the *Time* module enables a consistent representation of temporal information in the industrial system; and (6) the *Situation* module aims at representing relevant states of affairs associated to a particular scenario of interest and can consist of resources, observations and processes. These modules are described in detail in the following subsections. The main concepts of each module are defined using Description Logics (DL) [BCM+03] and some examples are provided in a simplified syntax based on the Semantic Web Rule Language (SWRL[2]).

5.1.1 The Resource module

The *Resource* module describes human entities and assets with attributes characterizing spatio-temporal features. In this section, global concepts are introduced to represent the resources of a factory. The `ManufacturingFacility` concept represents the machines and physical assets of a company. The `Staff` concept refers to all the people involved in industrial activities of a company who `operates` a `ManufacturingFacility`, such as operators and technicians. The `Product` concept aims to provide a representation of manufacturing products and product components.

Traditionally, machines used in manufacturing were monolithic pieces of equipment. The need for flexibility and reconfiguration as well as time and cost pressures has led to developing machines from modular subsystems which provide different operations. The *Resource* module contains concepts for different types of devices and hardware compo-

[2]https://www.w3.org/Submission/SWRL/

nents, and also contains a collection of equipment categories in the domain of production automation and manufacturing technology. A functional decomposition of a production line is performed:

- `Machine` is a device that performs a task by itself or by human intervention;
- `Workstation` is a small integrated physical groupings of machines;
- `Cell` is a set of combined workstations for a particular complex task; and
- `Line` is a group of cells.

This decomposition enables to represent different Industry 4.0 processing sequences and flexible or re-configurable production lines. Another relevant point is that by using this taxonomy, it is possible to describe the context of a `ManufacturingFacility` at different nested levels (*e.g.* characterising the context of a `Line` and depicting the context of a `Workstation` that belongs to that `Line`). The DL axioms for defining the `Resource` class and the `ManufacturingFacility` class are:

$$\begin{aligned} \texttt{Resource} &\equiv \texttt{ManufacturingFacility} \sqcup \texttt{Staff} \sqcup \texttt{Product} \\ \texttt{ManufacturingFacility} &\sqsubseteq \exists \texttt{operates}^{-1}.\texttt{Staff} \\ &\quad \sqcap (\texttt{Line} \sqcup \texttt{Cell} \sqcup \texttt{WorkSation} \sqcup \texttt{Machine}) \end{aligned}$$

As an example of how to represent resources, the following expression stands for a `Line :l` which contains a `Cell :c` with one `WorkStation :ws` composed by two machines: an `AssemblingMachine :am` operated by an `Operator :o` and a `ProcessingMachine :pm`.

$$\begin{aligned} &\texttt{AssemblingMachine(:am)} \wedge \texttt{ProcessingMachine(:pm)} \wedge \texttt{WorkSation(:ws)} \wedge \\ &\texttt{contains(:ws,:am)} \wedge \texttt{contains(:ws,:pm)} \wedge \texttt{Cell(:c)} \wedge \texttt{isPartOf(:ws,:c)} \wedge \\ &\texttt{Line(:l)} \wedge \texttt{isPartOf(:c,:l)} \wedge \texttt{Operator(:o)} \wedge \texttt{operates(:o,:am)} \end{aligned}$$

5.1.2 The Process module

A `Process` represents a task or a set of tasks performed by one or more resources and is specified with its contextual information, *e.g.* occurring time, place and related resources. The domain concepts given in the *Process* module represent a taxonomy of manufacturing processes. For example, processes include control operations and assembly operations. The `Process` class is sub-categorized into `LogisticOperation`, `HumanOperation` and `ManufacturingProcess`. This last kind of process includes cutting, drilling, milling, among others. The DL axiom used for defining the `Process` class is:

$$\begin{aligned}\texttt{Process} \sqsubseteq &(\texttt{LogisticOperation} \sqcup \texttt{HumanOperation} \sqcup \texttt{ManufacturingProcess}) \\ &\sqcap \exists \texttt{performsProcess}^{-1}.\texttt{Resource} \\ &\sqcap \exists \texttt{hasDuration.time:TimeInterval}\end{aligned}$$

The hasSubProcess property allows the representation of process composition. The spatio-temporal relationships among processes are expressed using spatial and temporal logic.

The following expression indicates that a WorkStation :ws performs a ManufacturingProcess :mp that lasts for an interval of time :intTP.

$$\begin{aligned}&\texttt{WorkSation(:ws)} \wedge \texttt{ManufacturingProcess(:mp)} \wedge \\ &\texttt{performsProcess(:ws,:mp)} \wedge \texttt{hasDuration(:mp,:intTP)}\end{aligned}$$

5.1.3 The Sensor module

All the sensors in a factory generate a stream of heterogeneous data. The lack of a formalized representation of these data would lead to weak interoperability among different systems and low re-usability. Therefore, the main goal of the *Sensor* module is to enrich the heterogeneous sensor data with a formalized semantics. This is done by adding metadata about contextual information (such as locations or timestamps).

The conceptualization of sensor measurements related to the sensing activity itself and to observations come from the Semantic Sensor Network[3] (SSN) ontology presented in [HJC+19]. The aim of SSN is to provide a formal representation of sensor characteristics, observable properties, observations and measurement processes that can be interpreted automatically. This allows to manage, interpret, query and control sensor networks using high-level specifications. The SSN ontology has been applied to a wide range of applications and use cases, such as satellite imagery, industrial infrastructures and the Internet of Things. The most relevant SSN concepts and relations that were re-used for the development of our ontological model are described below.

A sosa:Sensor is a device that detects and responds to some type of input from the physical environment. To do this, sensors implement a specific method that results in the calculation of the value of the observed property (sosa:ObservableProperty class).

The sosa:Observation class provides the structure to represent a single observation. Therefore, an instance of sosa:Observation is related to a single measurement (sosa:Result class) performed by a given sensor (sosa:Sensor class) on a single property of some entity (sosa:ObservableProperty and sosa:FeatureOfInterest classes, respectively).

The sosa:FeatureOfInterest class represents real-world phenomena and objects. When measuring the height of a building, the height is the observed property, and the building is the sosa:FeatureOfInterest.

[3] available at: http://www.w3.org/ns/ssn/

The ssn:Property class is defined as an attribute of an entity than can be measured (*e.g.* temperature). The sosa:ObservableProperty is a subclass of ssn:Property.

The sosa:Result class is the result of the observation: a value together with the unit (*e.g.* "26°C", "1000 mph", "1000 rpm"). To store the result value of an observation, the sosa:hasSimpleResult property can be used.

To specify the observation time, SSN has the sosa:resultTime datatype property that relates a sosa:Observation to a time instant. It indicates the time at which the observation was made.

One limitation of the SSN ontology is that it does not provide a way to represent the location of a sensor. In order to overcome this limitation, a new property is defined. The locatedIn property is used to assert that a sensor is deployed in a location.

The example below makes use of the previous properties to describe that a Machine :pm has a Sensor :vs which measures its vibration :pm_vibration.

$$\begin{aligned}&\texttt{Machine(:pm)} \wedge \texttt{WorkSation(:ws)} \wedge \texttt{isPartOf(:ws,:pm)} \wedge \texttt{Cell(:c)} \wedge \\ &\texttt{isPartOf(:ws,:c)} \wedge \texttt{Line(:l)} \wedge \texttt{isPartOf(:c,:l)} \wedge \texttt{Sensor(:vs)} \wedge \\ &\texttt{ssn:Property(:pm\_vibration)} \wedge \texttt{hasProperty(:pm,:pm\_vibration)} \wedge \\ &\texttt{observes(:vs,pm\_vibration)} \wedge \texttt{hasSensor(:pm,:vs)}\end{aligned}$$

5.1.4 The Location module

Many of the concepts that are represented in this ontological model are entities that can be located in space such as machines or sensors, and processes that can be located via their participating entities. The *Location* module provides concepts and relations that allow representing physical spatial places and the relations among them (e.g. a machine is located in a particular sector, and two sectors share a wall). To do so, the GeoSPARQL ontology [PH12] is used. It provides a flexible framework that is complete enough for geo-spatial objects representation as well as spatial relationships among geo-spatial objects. The geo:SpatialObject class contains two sub-classes called geo:Feature and geo:Geometry. The geo:Feature class represents 3D-objects or 2D-areas and can be assigned geometries that describe them through the geo:hasGeometry property.

The GeoSPARQL ontology also implements spatial operators to compute the topology between two geometries. They allow to model different characteristics such as orientation and distance. GeoSPARQL uses the Region Connection Calculus, introduced in [RCC92]. More specifically, it uses RCC-8 (the version with eight relations) that is well-known for representing mereo-topological relationships between spatial regions. RCC-8 consists of 8 basic relations (Figure 5.2) that are possible between two regions: disconnected (DC), externally connected (EC), equal (EQ), partially overlapping (PO), tangential proper part (TPP), tangential proper part inverse (TPPi), non-tangential proper part (NTPP), non-tangential proper part inverse (NTPPi). From these primitives relations, more complex combinations can be built. For example, a proper part (PP) is the union of TPP and NTPP.

Given two relational facts $\mathscr{R}(a,b)$ and $\mathscr{S}(b,c)$, where $\mathscr{R}$ and $\mathscr{S}$ are two RCC-8 relationships, the possible relation between a and c can be computed. The composition based

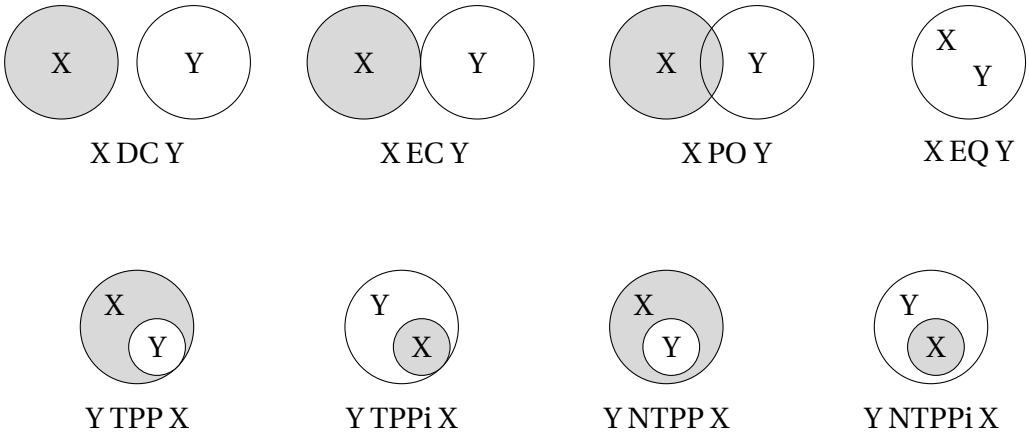

Figure 5.2 – RCC-8 basic relations

technique provides an efficient inference mechanism for a wide class of theories in the field of Artificial Intelligence, such as Qualitative Spatial Reasoning (QSR).

For any two RCC-8 relations $\mathscr{R}$, $\mathscr{S}$ in that set, their composition CT($\mathscr{R},\mathscr{S}$) is defined to be the unique smallest subset $\{\mathscr{T}_1,\ldots,\mathscr{T}_n\}$ of the set of eight relations such that $(\forall x \forall y \forall z)[\mathscr{R}(x,y) \wedge \mathscr{S}(y,z) \Rightarrow (\mathscr{T}_1(x,z) \vee \ldots \vee \mathscr{T}_n(x,z))]$ is a theorem in RCC.

Thus, CT($\mathscr{R},\mathscr{S}$) is the disjunction of all possible base relations which could hold between a and c provided that $\mathscr{R}(a,b)$ and $\mathscr{S}(b,c)$ hold. Table 5.1 gives the composition table of the RCC-8 relations. This table first appeared in [CCR93] and is known as the RCC-8 composition table (RCC-8 CT for short).

Table 5.1 – Composition table for RCC-8 relations (* stands for all eight RCC-8 relations)

o	**DC**	**EC**	**PO**	**TPP**	**NTPP**	**TPPi**	**NTPPi**	**EQ**
DC	*	DC,EC,PO, TPP,NTPP	DC,EC,PO, TPP,NTPP	DC,EC,PO, TPP,NTPP	DC,EC,PO, TPP,NTPP	DC	DC	DC
EC	DC,EC,PO, TPPi,NTPPi	DC,EC,PO, TPP,TPPi,EQ	DC,EC,PO, TPP,NTPP	EC,PO, TPP,NTPP	PO,TPP, NTPP	DC,EC	DC	EC
PO	DC,EC,PO, TPPi,NTPPi	DC,EC,PO, TPPi,NTPPi	*	PO,TPP, NTPP	PO,TPP, NTPP	DC,EC,PO, TPPi,NTPPi	DC,EC,PO, TPPi,NTPPi	PO
TPP	DC	DC,EC	DC,EC,PO, TPP,NTPP	TPP,NTPP	NTPP	DC,EC,PO, TPP,TPPi,EQ	DC,EC,PO, TPPi,NTPPi	TPP
NTPP	DC	DC	DC,EC,PO, TPP,NTPP	NTPP	NTPP	DC,EC,PO, TPP,NTPP	*	NTPP
TPPi	DC,EC,PO, TPPi,NTPPi	EC,PO, TPPi,NTPPi	PO,TPPi, NTPPi	PO,TPP, TPPi,EQ	PO,TPP, NTPP	TPPi,NTPPi	NTPPi	TPPi
NTPPi	DC,EC,PO, TPPi,NTPPi	PO,TPPi, NTPPi	PO,TPPi, NTPPi	PO,TPPi, NTPPi	PO,TPP,NTPP, TPPi,NTPPi,EQ	NTPPi	NTPPi	NTPPi
EQ	DC	EC	PO	TPP	NTPP	TPPi	NTPPi	EQ

The following expression uses the EC relation to express that the `geo:SpatialObject :l1` is externally connected with the `geo:SpatialObject :l2` (e.g. both locations share a wall).

```
geo:SpatialObject(:l1) ∧ geo:SpatialObject(:l2) ∧
spatial:rcc8ec(:l1,:l2)
```

5.1.5 The Time module

A suitable temporal model should allow a consistent representation of temporal information in the industrial system. Temporal modelling is key in the manufacturing domain because both sensor measurements and process representations need to be positioned in a temporal dimension so that they can be correctly interpreted by the industrial system. This module provides a way to represent temporal entities (*e.g.* time instants and time periods) as well as to interpret temporal relations (*e.g.* before, after, during, etc.). The *Time* module comprises all information related to the current time and allows time-stamping all the context information that may change over time.

The SWRL Temporal Ontology[4] (SWRLTO) proposed in [OD11] is reused as a temporal model. It provides a simple and efficient solution to operate with temporal information in queries and rules. This ontology has a main class named `temporal:ValidTime`, which has two sub-classes, `temporal:ValidInstant` and `temporal:ValidInterval`. The `temporal:ValidInstant` class denotes a point on a timeline (an instant) and the `temporal:ValidInterval` class models the time between two instants. These are declared by two data properties, `swrlto:hasStartTime` and `swrlto:hasFinishTime`. Besides, the `swrlto:Granularity` class describes the unit of measurement of the time reference (e.g., months, days, hours).

Another interesting class is `temporal:Fact`. It models an entity that can extend over time and that is associated to the `temporal:hasValidTime` property, indicating the time period during which the associated information is true. Values of this property belong to the `temporal:ValidTime` class.

Regarding temporal reasoning and querying, SWRLTO implements a series of predicates embedded in SWRL that allows operating with temporal relations. Most of them are based on Allen's time algebra [All94]. Although this algebra was not originally designed to relate an interval to an instant, nor was it designed to relate two instants to each other, SWRLTO includes specific operators to allow this. In addition, SWRLTO offers some SWRL operators to perform granularity conversions and duration calculations in different time units. The possible values are: *Years*, *Months*, *Days*, *Hours*, *Minutes*, *Seconds*, and *Milliseconds*. If no granularity is specified, the finest granularity supported by the library (i.e., *Milliseconds*) is used. The most common temporal operators are described below:

- `durationLessThan(duration,startDate,endDate,granularity)`: This predicate is satisfied if the first argument is less than the difference between two XML Schema date or date Time, specified by the `startDate` and `endDate` arguments, at the granularity specified by the final argument.

- `durationEqualTo(duration,startDate,endDate,granularity)`: This predicate is satisfied if the first argument is equal to the difference between two XML Schema date or date Time, specified by the `startDate` and `endDate` arguments, at the granularity specified by the final argument.

[4] http://swrl.stanford.edu/ontologies/built-ins/3.3/temporal.owl

- `durationGreaterThan(duration,startDate,endDate,granularity)`: This predicate is satisfied if the first argument is greater than the difference between two XML Schema date or date Time, specified by the `startDate` and `endDate` arguments, at the granularity specified by the final argument.
- `notdurationLessThan`, `notdurationEqualTo`, `notdurationGreaterThan`: These implement the inverse of the aforementioned duration operators.
- Allen's Temporal operators (Figure 5.3)

For example, let us suppose we have three machines `:pm1`, `:pm2` and `:pm3` which execute respectively three processes `:p1`, `:p2` and `:p3`. The duration intervals of each process are `:iP1`, `:iP2` and `:iP3`, respectively. We know that the execution of `:p2` starts when the execution of `:p1` is finished and that the whole execution of `:p3` is performed during `:p2`, as shown in (5.1), but we do not know the ending point of `:p2`, or the starting point of `:p3`. Even if the information about start and ending points is incomplete, it is possible to infer that the execution of `:p3` must be after the execution of `:p1` (*i.e.* `:iP3` must be `after` `:iP1`, as expressed in (5.2)), because the whole execution of `:p3` is performed during `:p2` (`:iP2 contains :iP3`).

$$\begin{aligned} &\texttt{ProcessingMachine(:pm1)} \wedge \texttt{ProcessingMachine(:pm2)} \wedge \\ &\texttt{ProcessingMachine(:pm3)} \wedge \texttt{ManufacturingProcess(:p1)} \wedge \\ &\texttt{ManufacturingProcess(:p2)} \wedge \texttt{ManufacturingProcess(:p3)} \wedge \\ &\texttt{performsProcess(:pm1,:p1)} \wedge \texttt{performsProcess(:pm2,:p2)} \wedge \\ &\texttt{performsProcess(:pm3,:p3)} \wedge \texttt{hasTime(:p1,:iP1)} \wedge \\ &\texttt{hasTime(:p2,:iP2)} \wedge \texttt{hasTime(:p3,:iP3)} \wedge \\ &\texttt{swrlto:meets(:iP1,:iP2)} \wedge \texttt{swrlto:contains(:iP2,:iP3)} \end{aligned} \tag{5.1}$$

$$\texttt{swrlto:after(:iP3,:iP1)} \tag{5.2}$$

Another example is a `WorkStation :ws` which is composed by a `Machine :pm` which performs a process `:mpp` whose duration is `:iMPP`, and a `Machine :am` which performs a process `:mpa` whose duration is `:iMPA`.

$$\begin{aligned} &\texttt{Machine(:am)} \wedge \texttt{Machine(:pm)} \wedge \texttt{WorkStation(:ws)} \wedge \texttt{isPartOf(:ws,:am)} \wedge \\ &\texttt{isPartOf(:ws,:pm)} \wedge \texttt{ManufacturingProcess(:mpp)} \wedge \\ &\texttt{performsProcess(:pm,:mpp)} \wedge \texttt{hasTime(:mpp,:iMPP)} \wedge \\ &\texttt{ManufacturingProcess(:mpa)} \wedge \texttt{performsProcess(:am,:mpa)} \wedge \\ &\texttt{hasTime(:mpa,:iMPA)} \end{aligned}$$

There are no specific spatio-temporal built-ins, however the combination of spatial and temporal operators allows a spatio-temporal analysis. The following expressions combines the use of spatial and temporal operators to show their straight integration. There are again three machines `:pm1`, `:pm2` `:pm3` which execute three processes `:p1`, `:p2` and `:p3`, respectively. `Process :p1 meets :p2` and `:p2 contains :p3`. The `:pm1` and `:pm2`

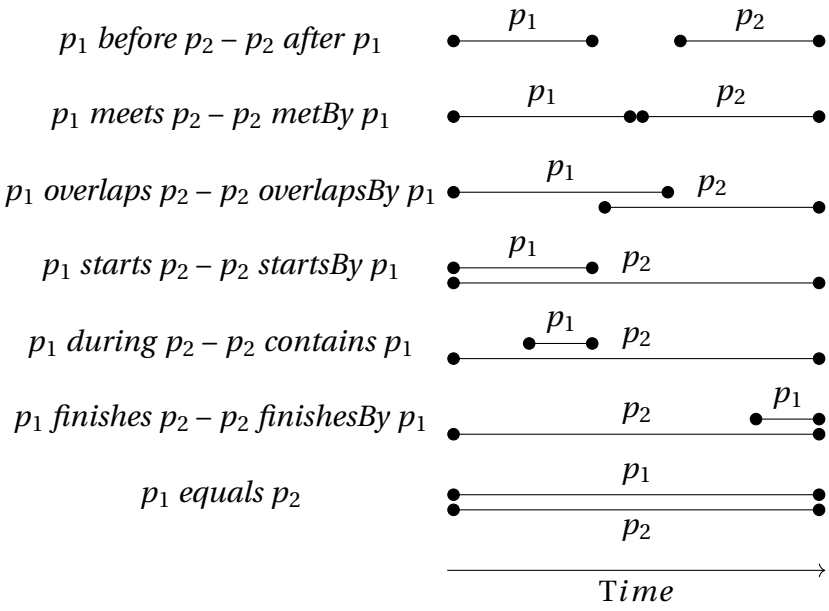

Figure 5.3 – Allen's Operators

machines are located in sector :l1 which is externally connected with sector :l2 where the :pm3 machine is located.

```
        Machine(:pm1) ∧ Machine(:pm2) ∧ Machine(:pm3) ∧
    ManufacturingProcess(:p1) ∧ ManufacturingProcess(:p2) ∧
    ManufacturingProcess(:p3) ∧ performsProcess(:pm1,:p1) ∧
    performsProcess(:pm2,:p2) ∧ performsProcess(:pm3,:p3) ∧
hasTime(:p1,:iP1) ∧ hasTime(:p2,:iP2) ∧ hasTime(:p3,:iP3) ∧
      time:meets(:iP1,:iP2) ∧ time:contains(:iP2,:iP3) ∧
      geo:SpatialObject(:l1) ∧ geo:SpatialObject(:l2) ∧
     spatial:rcc8ec(:l1,:l2) ∧ isInLocation(:pm1,:l1) ∧
       isInLocation(:pm2,:l1) ∧ isInLocation(:pm3,:l2)
```

5.1.6 The Situation module

This module aims at representing relevant situations associated to abnormal behaviors in the industrial system. Therefore, first of all, we introduce the notion of situation. As evoked before, a situation defines an abstract state of affairs associated to a particular scenario of interest.

In this case, a situation is a specific scenario in which the system state shows a particular combination of sensed values for its attributes (observations) that are not desirable and could lead to a failure. Thus, a situation describes "intermediate" or "abnormal" manufacturing conditions that are described with expert knowledge, in the form of constraints. Consequently, from a conceptual point of view, a situation involves a combination of at least, one resource, eventually associated to its location, and at least one sensor measurement, fulfilling the constraints set by the expert. The whole can be linked through spatial, temporal and/or spatio-temporal relationships.

It should be noted, that a situation is actually a general description (concept) about a specific scenario and there may be several instances (individuals) of these situations occurring at different points in time. For example, an abnormal behavior of a particular machine can be represented by a situation. This situation can happen more than once on the same machine or even on another machine of the same type; *i.e.* which means that there can be different instances of the abnormal situation in the ontology. These situations take place over a period of time, and their duration is represented by the property `situationTime`.

The `Cause` class represents the possible causes that provoke a situation. A situation is linked to its causes through the `hasCauses` property. The `Action` concept represents actions (preventive or reactive) needed to mitigate the situation or its severity. The actions can be diverse, such as tasks performed by operators or remote changes in the operating parameters of the machines. They are linked to a situation through the `requiredAction` property. In both cases, the link between a situation and its causes and the required actions is obtained from expert knowledge. The `Constraint` concept represents defined restrictions on certain observable properties, such as a maximum threshold for one value. As a situation is expressed as a combination of constraints, this is represented through the `associatedConstraints` relation. Considering the concepts defined above, the concept of situation is formally defined by the following DL axiom.

$$\begin{aligned}\texttt{Situation} \sqsubseteq\ & \exists \texttt{hasConstraint}.\texttt{Constraint} \\ & \sqcap \exists \texttt{hasObservation}.\texttt{Observation} \\ & \sqcap \exists \texttt{situationTime}.\texttt{time:TemporalEntity} \\ & \sqcap \exists \texttt{involvedInSituation}^{-1}.\texttt{Resource}\end{aligned}$$

For example, a situation can be "the temperature in sector `:l4` exceeds `TEMP-MAX` during the execution of process `:p1` while process `:p2` is also in execution, and both take place in the same sector". This is represented by the `HighTemp2Proc` class, depicted in the expression (5.3). The expression (5.4) shows the resources and processes involved in the situation.

$$\begin{aligned}&\texttt{ProcessingMachine(:pm1)} \wedge \texttt{ManufacturingProcess(:p1)} \wedge \\ &\texttt{ProcessingMachine(:pm2)} \wedge \texttt{ManufacturingProcess(:p2)} \wedge \\ &\texttt{performsProcess(:pm1,:p1)} \wedge \texttt{performsProcess(:pm2,:p2)} \wedge \\ &\texttt{hasDuration(:p1,:iP1)} \wedge \texttt{hasDuration(:p2,:iP2)} \wedge \\ &\texttt{time:during(:iP1,:iP2)} \wedge \texttt{geo:SpatialObject(:l4)} \wedge \\ &\texttt{sosa:Sensor(:s1)} \wedge \texttt{locatedIn(:s1,:l4)} \wedge \\ &\texttt{sosa:Observation(:senseT)} \wedge \texttt{ssn:Property(:temp)} \wedge \\ &\texttt{sosa:madeObservation(:s1,:senseT)} \wedge \texttt{observes(:s1,:temp)} \wedge \\ &\texttt{sosa:hasTime(:senseT,:tObs)} \wedge \texttt{swrlto:during(:tObs,:iP1)} \wedge \\ &\texttt{sosa:hasSimpleResult(:senseT,:v)} \wedge \texttt{swrlb:greaterThan(:v,TEMP-MAX)}\end{aligned} \tag{5.3}$$

$$\begin{aligned}&\texttt{HighTemp2Proc(:htS)} \wedge \texttt{situationTime(:htS,:htS-time)} \wedge \\ &\texttt{involvedInSituation(:pm1,:htS)} \wedge \texttt{concernedBySituation(:p1,:htS)} \wedge \\ &\texttt{involvedInSituation(:pm2,:htS)} \wedge \texttt{concernedBySituation(:p2,:htS)} \wedge \\ &\texttt{hasObservation(:htS,:senseT)} \wedge \texttt{hasConstraint(:htS,:highTemp)} \wedge \\ &\texttt{hasDescription(:htS,"High temp in l4 sector"\^\^xsd:String)}\end{aligned} \tag{5.4}$$

5.1.7 Integration of all the modules

The ontological modules described above reuse others ontologies that have been developed to address specific needs in different domains and following different approaches. Therefore, it is necessary to semantically integrate them in order to have an ontological model that allows context modelling in the Industry 4.0 domain.

The integration among the modules is done mainly through the use of relationships (object properties, equivalence (≡) and subsumption (⊑)) that link concepts from one module with concepts from other modules. First, the integration of the *Resource* module with the rest of the modules is done:

- The *Resource* module is linked with the *Sensor* module through the `sosa:hosts` relation. This relation links a `Resource` with a `sosa:Sensor`. It is used to assert that one or more sensors are attached to a `Resource` in order to measure certain properties of it.

- The *Resource* module is linked with the *Process* module through the `performsProcess` relation. This relation links a `Resource` with a `Process`. It is used to assert that a resource performs one or more processes.

- The *Resource* module is linked with the *Location* module through the `isInLocation` relation. This relation links a `Resource` with a `geo:SpatialObject`. It is used to assert that a resource is located in a specific location.

- The *Resource* module is linked with the *Situation* module through the `involvedInSituation` relation. This relation links a `Resource` with one or more `Situations`. It is used to assert that a resource is involved in one or more particular situation(s).

The *Process* module is linked to the *Resource* module as previously presented. Furthermore, it is also linked to the *Time* module, the *Location* module and the *Situation* module as described below:

- The *Process* module is linked with the *Time* module through the `hasDuration` relation. This relation links a `Process` with a `temporal:ValidPeriod`. It is used to assert the duration of a process.

- The *Process* module is linked with the *Location* module through the `happensIn` relation. This relation links a `Process` with a `geo:SpatialObject`. It is used to assert the location where a process occurs. It is worth highlighting that the location where a process occurs can be inferred from the location of the resources that execute that process.

- The *Process* module is linked with the *Situation* module through the `concernedBySituation` relation. This relation links a `Process` with one or more `Situations`. It is used to assert that a process is concerned by one or more situation(s). Similarly to the previous item, the fact that a situation concerns a process can be inferred from the fact that the situation involves the resource that executes that process.

The *Sensor* module is linked to the *Resource* module as previously presented. Furthermore, it is also linked to the *Time* module, the *Location* module and the *Situation* module as described below:

- The *Sensor* module is linked with the *Time* module through the `hasTime` relation. This relation links an `Observation` with a `temporal:ValidTime`. It is used to assert the time when an observation was made.

- The *Sensor* module is linked with the *Location* module through the `locatedIn` relation. One limitation of the existing SSN ontology is that it does not provide a way to represent the location of a sensor. In order to overcome this limitation, a new property is defined. The `locatedIn` property is used to assert that a sensor is deployed in a certain location. It is worth highlighting that the location where a sensor is located can be inferred from the location of the resource that hosts that sensor (if any).

- The *Sensor* module is linked with the *Situation* module through two relations, the `definedOn` relation and the `hasObservation` relation. The first one links a `Constraint` with a `sosa:ObservableProperty`. It is used to assert that a constraint is defined on a particular property. The second one links one or more `Situation` with one or more `Observation`. It is used to assert that a situation has one or more observations. From the observations we can obtain the necessary information to determine the resources involved in a situation as well as the processes concerned by the situation, its location and its time.

Finally, the *Situation* module is linked to the *Resource* module, the *Process* module and the *Sensor* module as described above. It is also linked to the *Time* module and the *Location* module.

- The *Situation* module is linked with the *Time* module through the `situationTime` relation. This relation links a `Situation` with a `temporal:ValidPeriod`. It is used to assert the duration of a situation. Furthermore, the `Situation` class is a subclass of the `temporal:Fact` class, since a situation occurs over an interval of time

(`Situation` $\sqsubseteq$ `temporal:Fact`), therefore, it has a valid time. As mentioned in the description of the *Situation* module, in the industrial domain there are relations among different concepts that evolve or change over time, such is the case of a resource involved in a situation that happens over a period of time. Therefore, the property that asserts that a resource is in a certain situation (`involvedInSituation`) holds during the interval in which the situation occurs.

- The *Situation* module is linked with the *Location* module through the `occursIn` relation. This relation links a `Situation` with one or more `geo:SpatialObject`. It is used to assert the location or locations where a situation occurs. It is worth highlighting that the location(s) where a situation occurs can be inferred from the location of the resource that are involved by the situation.

All modules are linked together in one way or another as described above. This integration provides a way to represent situations of interest that take into account the context of the resources.

5.2 Ontology alignment with a foundational ontology

In order to further obtain a rigorous conceptualization of the proposed semantic model, we semantically align it with a foundational ontology. Ontology alignment is defined as a set of correspondences between two ontologies [SE05]. As already mentioned in the section where all the conceptual modules are integrated (section 5.1.7), a correspondence establishes a sustained relationship between two entities, one of each ontology, such as: equivalence ($\equiv$), subsumption ($\sqsubseteq$) and disjointedness ($\perp$). In an ontological alignment, a foundational ontology acts as an external source of common knowledge since the concepts defined or reused in the proposed ontology are a specialization of general concepts shared by different domains. This semantic technique allows to relate concepts from different ontologies based on the analysis of their interpretations. This technique is less ambiguous and more rigorous than those based on the names of the entities as it happens with the syntactic matching methods.

The main concepts of the ontological model are aligned with the DOLCE+DnS Ultralite (DUL) ontology, a simplified version of DOLCE (Descriptive Ontology for Linguistic and Cognitive Engineering) introduced in [MBG+03]. The choice of DUL for the alignment is based on the fact that it is a recognized ontology that covers several domains and research fields. Therefore, it favours integration and reuse in other environments. It has a rigorous conceptualization that facilitates the interpretation of the domain primitives.

The SSN ontology, one of the most important components of our ontology is already aligned with DUL[5]. The alignment among the concepts of SSN and DUL is shown in the equations 5.8, 5.9, 5.10, 5.11 and 5.12.

It is important to note that the whole alignment is not trivial and therefore there is not a unique way to align the concepts of our semantic model (or any semantic model) with

[5]https://www.w3.org/TR/vocab-ssn/#DUL_Alignment

the ones from DUL.

Since SSN and DUL are aligned, the second step is the alignment between the SWRLTO and GeoSPARQL ontologies with DUL. The entities defined in these ontologies are at a similar level of abstraction. However, the SWRLTO and DUL ontologies consider different representations of temporal information. DOLCE (and therefore DUL) adopts the 4D fluent approach while SWRLTO is based on a reification approach.

In the 4D-fluent approach, time concepts are represented as 4-dimensional objects where the fourth dimension is time. It divides the world into two categories:

- the *endurants*, that represent entities that can be perceived in a complete way at any instant of time or, in other words, it is information that does not change over time; and

- the *perdurants*, that represent entities that extend in time accumulating different temporal parts. The changes occur in the properties of the temporal part, while the properties of the static part do not change.

Although SWRLTO is based on a reification approach, it has no restrictions regarding its use with other schemes. Indeed, the meaning of `dolce:Perdurant` is consistent with the definition of the `swrlto:Fact` concept, since it represents entities that can be extended in time. For this reason, an equivalence relationship is established between them (equation 5.6). Another alignment of concepts to emphasize is the one defined between the `swrlto:ValidTime` and `dul:Time Interval` classes (equation 5.5). Intuitively, it appears that `swrlto:ValidTime` is more general than `dul:Time Interval` since it involves instants and intervals of time while `dul:Time Interval` is defined as ”any region in a dimensional space whose purpose is the representation of time” in DUL.

The `geo:SpatialObject` class represents everything that can have a spatial representation. This definition is consistent with that of the `dul:PhysicalPlace` concept of DUL, that is "any object that has a proper space region". Therefore, we align these concepts as shown in equation 5.7.

As a result of the alignment between the SWRLTO and GeoSPARQL ontologies with DUL, it can be highlighted that any domain ontology based on DUL can incorporate temporal and spatial reasoning capabilities. Furthermore, since the alignment operates on top level concepts, this does not alter the domain ontologies. This means that each member of `dolce:perdurant` with a particular `dul:Time Interval` is interpreted as a `swrlto:Fact` with a `swrlto:ValidTime`. Therefore, the functions embedded in SWRLTO can be used in rules and queries to reason about time. Similarly, each member of `dul:PhysicalPlace` is interpreted as `geo:SpatialObject`. Therefore, the operators embedded in GeoSPARQL can be used to reason about spatial aspects.

Finally, we present the alignments of the *Resource*, *Process* and *Situation* modules with DUL. From these modules, the main concepts are aligned with DUL.The `Situation` concept is defined as a sub-class of the `temporal:Fact` concept that is already aligned with DUL (equation 5.14). The `Resource` concept is interpreted as the `dul:Object` concept

of DUL since a dul:Object is defined as "any physical, social, or mental object" (equation 5.13). The Process concept is interpreted as the dul:Method concept of DUL since a dul:Method is "a description that defines or uses concepts in order to guide the carrying out of actions aimed at accomplishing a task" (equation 5.15).

Alignment of classes

$$\texttt{swrlto:ValidTime} \equiv \texttt{dul/dolce:Time Interval} \tag{5.5}$$
$$\texttt{swrlto:Fact} \equiv \texttt{dolce:Perdurant} \tag{5.6}$$
$$\texttt{geo:SpatialObject} \equiv \texttt{dul:PhysicalPlace} \tag{5.7}$$
$$\texttt{sosa:Sensor} \sqsubseteq \texttt{dul:PhysicalObject} \tag{5.8}$$
$$\texttt{sosa:Observation} \sqsubseteq \texttt{dul:Situation} \tag{5.9}$$
$$\texttt{sosa:Result} \sqsubseteq \texttt{dul:InformationObject} \tag{5.10}$$
$$\texttt{sosa:FeatureOfInterest} \sqsubseteq \texttt{dul:Event} \sqcup \texttt{dul:Object} \tag{5.11}$$
$$\texttt{sosa:Property} \sqsubseteq \texttt{dul:Quality} \tag{5.12}$$
$$\texttt{Resource} \sqsubseteq \texttt{dul:Object} \tag{5.13}$$
$$\texttt{Situation} \sqsubseteq \texttt{swrlto:Fact} \tag{5.14}$$
$$\texttt{Process} \sqsubseteq \texttt{dul:Method} \tag{5.15}$$

Alignment of object properties

$$\texttt{swrlto:hasValidTime} \sqsupseteq \texttt{dul:is observable at} \tag{5.16}$$
$$\texttt{swrlto:hasGranularity} \sqsubseteq \texttt{dul/dolce:has region} \tag{5.17}$$

Alignment of data properties

$$\texttt{swrlto:hasStartTime} \sqsupseteq \texttt{dul:has interval date} \tag{5.18}$$
$$\texttt{swrlto:hasFinishTime} \sqsupseteq \texttt{dul:has interval date} \tag{5.19}$$
$$\texttt{swrlto:hasTime} \sqsupseteq \texttt{dul:has interval date} \tag{5.20}$$

5.3 Ontology evaluation

Finally, we evaluate the ontology to check that it does not contain pitfalls and that it covers all the requirements identified in section 3.4. In order to detect common mistakes done when developing ontologies we have used OOPS! [PVGPSF14]. In OOPS!, ontology pitfalls are classified into three categories: structural, functional, and usability-profiling. Under each category, fine-grained classification criteria are provided to cope with specific types of mistakes. The ontology is evaluated according to the following three categories:

- **Structural dimension** focuses on mistakes detection on syntax and formal semantics. This category is composed of five criteria:
  - modeling decisions, which evaluates whether users use the ontology implementation language in a correct way;
  - real-world modeling or common sense, which evaluates the completeness of the domain knowledge formalized by the ontology;
  - no inference, which checks whether the desired knowledge can be inferred through ontology reasoning;
  - wrong inference, which refers to the detection of inference that leads to erroneous or invalid knowledge; and
  - ontology language, which assesses the correctness of the ontology development language.
- **Functional dimension** considers the intended use and functionality of the proposed ontology. Under this category, two specific criteria are used to evaluate our proposal:
  - requirement completeness, which evaluates coverage of the domain knowledge that is formalized by the ontology;
  - application context, which evaluates the adequacy of the ontology for a given use case or application.
- **Usability-profiling dimension** evaluates the level of ease of communication when different users use the same ontology. Within this category, two specific criteria are applied for ontology evaluation:
  - ontology understanding, which evaluates the quality of information or knowledge that is provided to users;
  - ontology clarity, which assesses the quality of ontology elements for being easily recognized and understood by users. This criteria are commonly used to check the quality of ontologies when users do not have sufficient domain knowledge.

The evaluation of COInd4 with OOPS! has yield some minor pitfalls, that do not affect the consistency, reasoning or applicability of the ontology.

A first issue mainly regards ”missing domain or range” errors inherited from the SSN ontology. The documentation of SSN states that not adding the domain or range to certain properties is a design decision not to be restrictive with them. However, we have decided to add the missing range and domains. Another minor issue is the case of the ”recursive definitions”. In our case, it was needed to define several recursive relations, such as ”A production equipment can contain another production equipment”. Therefore, we have not considered this pitfall as a mistake.

The final evaluation of the COInd4 ontology by OOPS! appears in Figure 5.4.

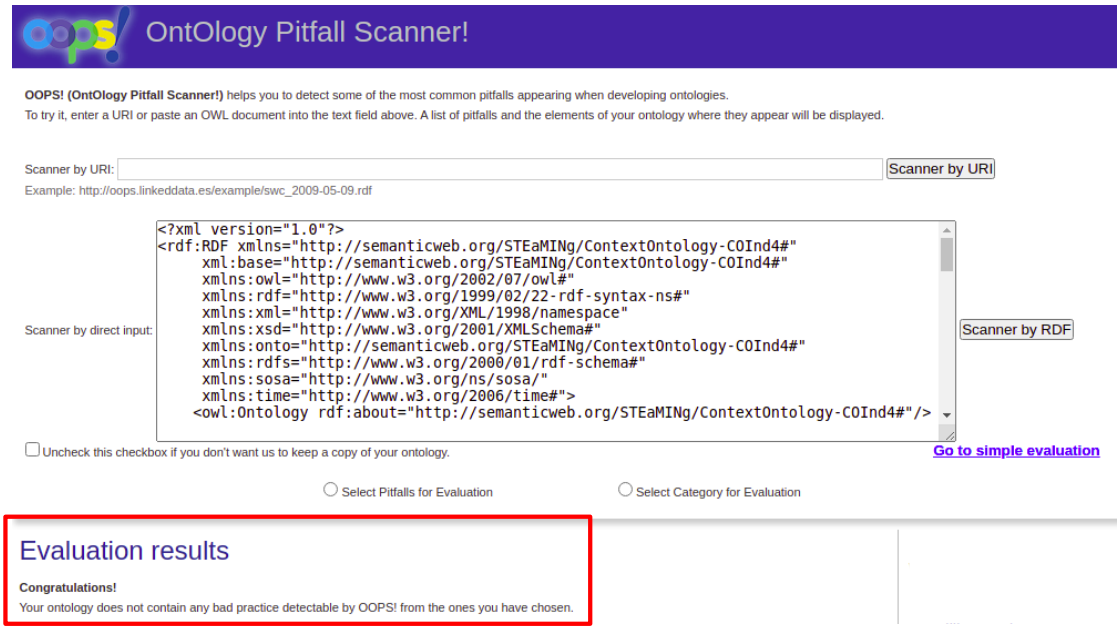

Figure 5.4 – Screenshot of the proposed ontology evaluation results by OOPS!

5.4 Conclusion

This chapter presents an ontology-based approach for context modeling in the industrial domain. The proposal combines existing ontologies with two main goals: (1) to obtain a model that satisfies our requirements with special emphasis on temporal and spatial relations among resources and processes; and (2) to represent particular situations of interest in an industrial scenario. Situations of interest in the industrial domain depend on sensor data, domain knowledge on environments and resources. The ontological model consists of six modules, named the *Resource* module, the *Process* module, the *Sensor* module, the *Location* module, the *Time* module and the *Situation* module. The proposed model is written in OWL DL, a standard language with formal semantics based on logic [BHS05].

Furthermore, our ontological model provides a way to capture the dynamic changes and the evolution of knowledge in time, such as the different situations a machine can go through or the changes in the values of one of its parameters according to the different decisions made. This is a key point to deal with the dynamics of manufacturing processes as mentioned in the introduction to this manuscript. The proposed ontology is generic and extensible to cover a wide spectrum of manufacturing services. Its modular architecture also allows the description of the capabilities of manufacturing resources at different levels of modularity namely, `Machine`, `Workstation`, `Cell` and `Line`. Some examples about how the industrial domain can be represented through our ontological model were given throughout the chapter.

The ontological model is aligned with the DUL foundational ontology, for obtaining a rigorous conceptualization. The proposed approach is evaluated according to different ontology evaluation criteria such as structure, function and usability.

Chapter 6

Stream Reasoning for Abnormal Situation Detection and Diagnosis

Contents

According to the metrics of Lean manufacturing [CBI], as measured by Overall Equipment Effectiveness (OEE), world-class manufacturing factories are working at 85% of their theoretical capacity, while medium-sized ones are at around 60%. Some of the reasons for this are detailed in [Has11]. The main ones are the launching of unnecessary maintenance tasks and, despite this, the breakdown of equipment. This leads to increased maintenance costs and production stoppages. In order to tackle these issues, as evoked in Chapter 4, factories rely on condition monitoring, which is the task of monitoring all the equipment involved in a manufacturing process for early detection of anomalies. In particular, in Industry 4.0 factories, machines and other plant resources are equipped with sensors that collect data continuously. These data can be exploited for condition monitoring.

In this chapter, the focus is on the real-time use of data collected from sensors attached to resources and the environment to detect situations that may lead to failures disrupting production processes, and to determine their possible causes. A situation, as defined in section 5.1.6, can be seen as a combination of one or several sensor measurements linked through spatial, temporal or spatio-temporal relationships, along with constraints given by experts. The detection of these situations requires the interpretation of observations considering the observations context (for example, other spatially or temporally related observations). These observations are generated from heterogeneous data sources. Therefore, data integration for observations interpretation is a key point to consider. As discussed in Chapter 2, Semantic Web technologies are increasingly used to improve the interoperability in scenarios where data is heterogeneous. However, current solutions for reasoning on RDF or OWL data are not appropriate for the dynamic nature of data in an industrial scenario, where the manufacturing processes are executed over time and under different contexts. To deal with this issue, a recent research field called stream reasoning propose to unify reasoning and stream processing. Stream reasoning, presented in section 2.3.2, is based on the continuous processing of data streams together with rich background knowledge to support decision systems [SCDV+19].

This chapter describes in detail the components in charge of Monitoring and Diagnosis introduced in Chapter 4. An approach that uses stream reasoning and classical reasoning methods (1) to detect situations that potentially lead to failures; and (2) to identify the possible causes that generated those situations is presented in this chapter. Knowing the causes helps choosing the most appropriate decision to avoid the interruption of manufacturing processes. The proposed approach enriches the data collected from sensors with contextual information by using the semantic model introduced in Chapter 5.

The remainder of the chapter is structured as follows: section 6.1 provides a detailed description of the Monitoring and the Diagnosis components for situation detection and cause determination. In section 6.2, an illustrative case study is introduced. Finally, in section 6.3 some conclusions are presented.

6.1 Relevant situation detection and cause determination

Certain situations that may lead to machine failures can be detected by interpreting observations in their contexts, for example, if an observation has an abnormal value it may

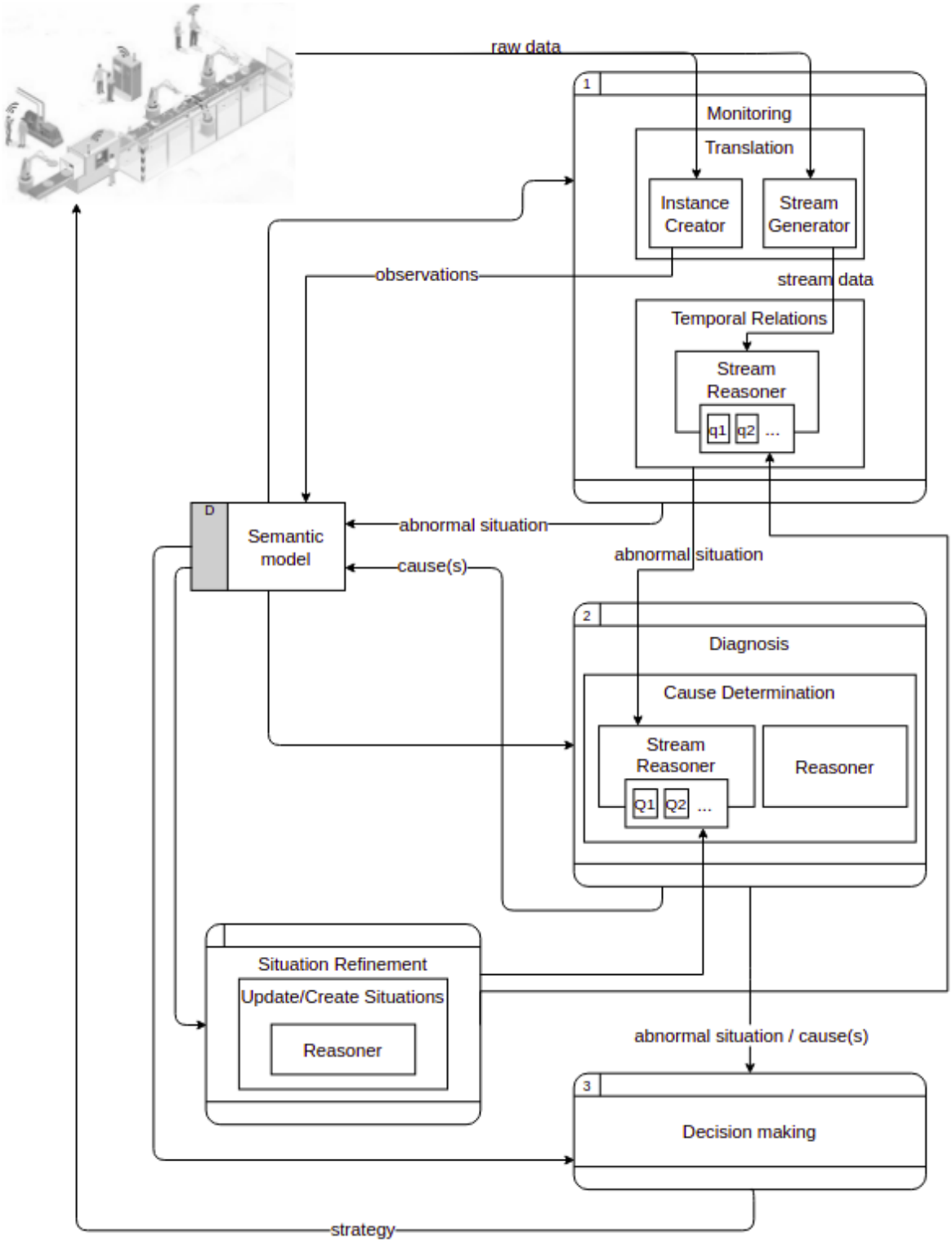

Figure 6.1 – Main components and workflow of the proposed framework.

be because another parameter is having abnormal values too. Let us consider the case where the temperature of a machine and the temperature of one of its components are being monitored. It is known that an increase in the temperature of the machine may be due to an increase in the temperature of its component, or vice versa. This allows the exploitation of expert knowledge about relationships between the values of certain parameters from the machines, from the processes and from their context for interpreting observations. Through the early detection of situations, the maintenance schedule can be adapted or further measures to prevent unexpected downtime can be taken.

The main components of our proposed framework are shown in Figure 6.1. The modules to detect situations in real time are *Translation* and *Temporal relations*. These two are part of the Monitoring component and they are explained in detail below in section 6.1.1. The Diagnosis component has only one module called *Cause Determination*, it is introduced in section 6.1.2. Although the Monitoring and Diagnosis components perform different tasks, we have decided to explain them together since, as mentioned in Chapter 4, their tasks are closely related. In addition, Figure 6.1 also shows the Situation refinement component and the Decision making component. The first one contains the *Update/Create situation* module that aims at refining already defined situations or defining new ones. This module is explained in section 6.1.3. The *Decision Making* component is not part of the task of detecting situations or determining the possible causes, but it exploits this information to support decision making tasks. It is described in section 6.1.4.

6.1.1 The Monitoring component

The modules involved in situation detection are *Translation* and *Temporal Relations*. The following two subsections explain both and describe the interaction among them as well as with the ontological model.

The *Translation* module

This module is responsible for (i) converting acquired data from sensors to RDF streams, and (ii) inserting them as instances into the ontology. Both tasks are performed respectively by the *Stream Generator* and the *Instance creator* sub-modules.

The *Stream Generator* sub-module performs semantic enrichment of the acquired data, using the concepts and relations among them as defined in the ontological model. This allows the module to stream out semantically enriched data streams that are then consumed by the *Stream Reasoner*. The output streams are RDF streams. An RDF stream is an ordered sequence of pairs, where each pair is constituted by an RDF triplet and its timestamp t, (`<Subject,Predicate,Object>`, t) as defined in section 2.3.2.

The *Instance creator* sub-module creates instances from the received data and inserts them in the ontology, *i.e.* it is in charge of populating the ontological model with observations and their corresponding metadata, such as the sensor(s) which made the observation(s), the observed property and the timestamp.

The *Temporal Relations* module

Once the data from the distributed and heterogeneous data sources are available in a homogeneous, contextualized and temporally ordered representation, the streams can be explored to generate new knowledge.

A set of queries, which combine background knowledge extracted from the ontology and some relevant parts of the streams, is registered and executed by the *Stream Reasoner* over the data streams. These queries represent particular situations to be identified and they include mainly temporal dependencies between observations (that can be normal ones or anomalies). There are several possibilities for the implementation of this module. Some of them were discussed in section 2.3.2.

When this module detects situations, they can be converted into RDF streams and be returned as output, *i.e.* this module produces streams of situations as output. This output feeds another stream reasoner in the *Cause determination* module. In this way, the *Temporal Relations* module itself can be seen as an advanced sensor able to produce high level data. Furthermore, the detected situations are stored in the ontological model indicating also the resources involved in that situation. This is represented by the `involvedInSituation` relation explained in section 5.1.7 of Chapter 5.

6.1.2 The Diagnosis component

This component has only one module called *Cause Determination* to determine the possible causes that caused an abnormal situation.

The *Cause Determination* module

The purpose of the *Cause Determination* module is to identify the possible causes that generated a situation detected by the *Temporal Relation* module. For this, two components are used separately: a *Stream Reasoner* and a *Reasoner*.

Stream reasoning is more suitable to highly dynamic data than classical reasoning approaches. Thus, in the case where causes need to be determined in real-time, the *Stream Reasoner* is used to identify the causes. The association between the situations and their possible causes are stored in the ontological model and they are exploited by this module to return them. However, it is possible that some situations do not have identified causes in a real scenario, in which case the system notifies that the causes are unknown.

Therefore, in order to determine the possible causes of a situation classical reasoning approaches can be used. They provide other inferences that can help in determining the causes of a situation. More complex queries can be performed to the ontological model in order to extract information useful for cause determination. For example, it can be necessary to consider the Open World Assumption, which states that the absence of a statement alone cannot be used to infer that the statement is false. In this case, the *Reasoner* can be also used over the ontology to infer the causes, if the real-time requirement is not needed. This last option has some advantages over the previous one. If the cause is identified later,

it is added as an instance to the ontological model and linked to the situation for future use.

In both cases, the Diagnosis component provides the *Decision Making* component with the identified situation together with the possible causes inferred by the reasoner(s). The association between the situation and the causes is also added to the ontological module.

6.1.3 The Abnormal Situation Refinement component

As mentioned in the previous subsection, the *Translation* module, in addition to generating the data streams, populates the ontology with semantically enriched data. The main purpose of the *Update/Create situations* module is to exploit the historical data stored as instances in the ontological model, to refine the already defined situations and eventually to find new ones. This can be accomplished by (i) reasoning over the ontology to derive implicit knowledge, and (ii) interacting with the ontology by issuing one-time queries. These queries can be executed to check how often a situation happens or to check the trend of the values of certain properties to refine the constraints on them and thus refine the situations associated with those constraints.

This module uses a classic reasoner and not a stream reasoner because reasoning is applied to static data stored in the ontological model and not to the data streams.

6.1.4 The Decision Making component

Considering the situations and their causes, it is possible to support decision making tasks to determine what actions to launch to correct the behavior of the machines and avoid breakdowns. The actions to be triggered are diverse. For example, certain maintenance tasks can be launched remotely and performed by the machines themselves or the smart system can issue an alert to notify the operator closest to the machine to inspect it, if human action is needed.

Each application can perform more advanced reasoning and processing on the received data in order to determine the adaptation strategies that are needed to improve the behavior of the industrial system. These strategies may include changes to the operation parameters of a machine or the launching of a maintenance task, that are launched by the Application layer (Figure 4.1).

6.2 An Illustrative case study with two properties

In this section, a case study is presented to illustrate the application of our approach and the advantages of detecting abnormal situations by interpreting observations in their contexts. First, the illustrative case study with two properties whose values have a causal relationship in time is formally described. Second, the illustrative case study is instantiated in an industrial scenario with two sensors deployed in a machine.

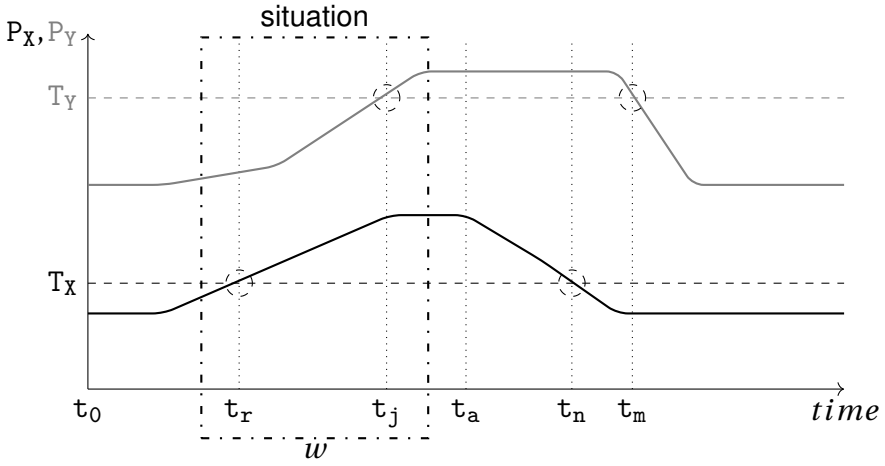

Figure 6.2 – Example of a relevant situation involving two properties and an action triggered to correct the values.

The formal description of the case study (Figure 6.2) is the following: Consider two properties P_X and P_Y. At time t_r property P_X exceeds its threshold T_X indicating a positive deviation from its normal values. At time t_j property P_Y exceeds its threshold T_Y also indicating a positive deviation from its normal values. This deviation is a consequence resulting from the increase of the values of P_X. At time t_a an action is triggered to correct the behavior of P_X and at time t_n, property P_X is under T_X. Then at t_m, property P_Y also falls under T_Y. In this case, the sensed value of P_Y at t_j is interpreted in the context of property P_X exceeding its threshold, since t_r. The action launched at t_a is decided considering this interpretation.

Let us consider the following industrial scenario with a machine M_1 as a part of a production line in a company. This machine has a main component M_2, on which the operation of the machine depends. Both the machine and its component have sensors, called $SensorTM_1$ and $SensorTM_2$, which measure respectively the temperature of the machine and of its component. The goal is to monitor the temperature of the machine and of its component. Formally, P_Y and P_X are the temperature properties of the machine and its component.

As shown in Figure 6.2, during the execution of the task performed by the machine, an increase in the M_2 temperature is observed. Let us assume that this is due to dirty filters in the cooling system. The M_2 temperature deviates from its normal behavior at t_r, while P_Y is normal. The observation made by $SensorTM_2$ at time t_r is represented in the RDF format using the structure of our ontology in Listing 6.1. Later, at time t_j the M_1 temperature also exceeds its threshold.

```
:obsTempM2 a                    sosa:Observation ;
           sosa:observedProperty :Px ;
           sosa:madeBySensor     :SensorTM2 ;
           sosa:hasSimpleResult  Tx^^xsd:float ;
           :hasTime              tr .
```

Listing 6.1 – Example of an observation in RDF using the structure of our ontology.

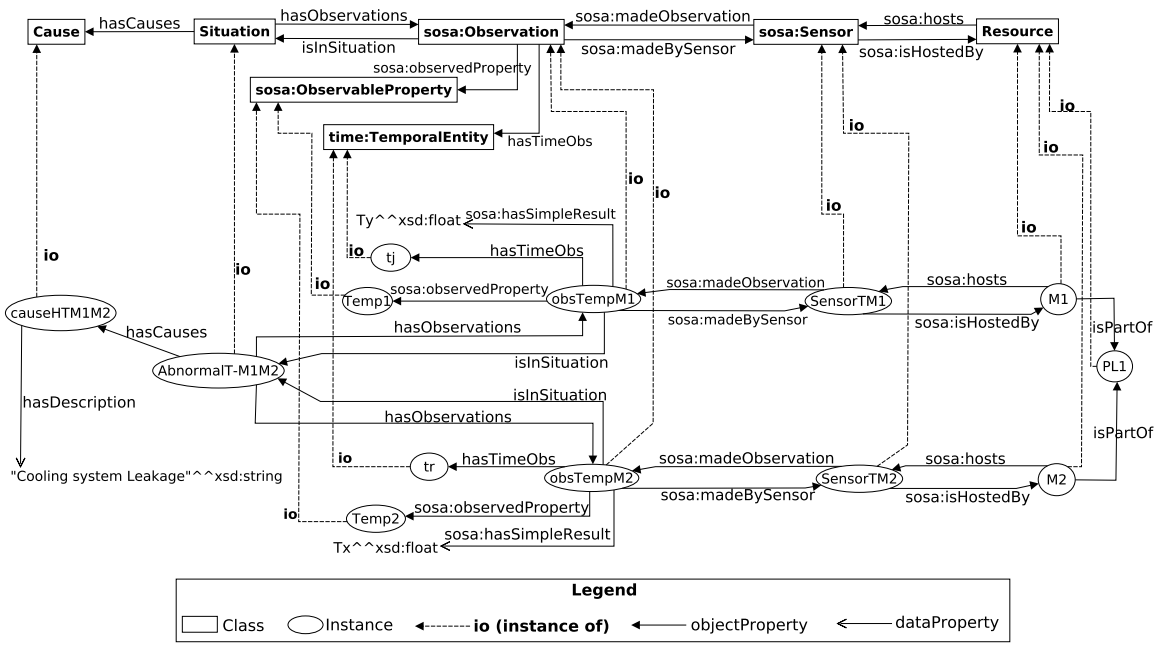

Figure 6.3 – Representation of the scenario presented in the case study using our model.

This deviation is a consequence of the increased M_2 temperature, as M_2 is the main heat source in M_1. Consequentially, a higher M_2 temperature leads to a higher M_1 temperature. Our approach allows to exploit this knowledge about the link between the two temperature values. Through this interpretation of the anomaly in P_Y, it is possible to detect situation `AbnormalT-M1M2`.

The C-SPARQL query shown in Listing 6.2 detects this situation and selects the observations and the resources relevant to the situation of interest. Once the situation is detected, an instance (`AbnormalT-M1M2`) is added to the ontological model, as well as the relations with the corresponding observations (`obsTempM1` and `obsTempM2`). Figure 6.3 shows the instantiation of the ontological model after reasoning.

The detection of this situation and its interpretation avoids a general inspection of M_1 due to its high temperature. Only the state of its component M_2 has to be verified. This allows to take action to correct the behavior of P_Y considering its context. At time t_a, an action is taken about M_2 (cleaning or replacement of the cooling system filters) that solves the problem, which enables the temperature of M_2 to decrease. After some time, this leads to a decrease in the temperature of M_1.

Although the case study presented in this section involves only two properties, it shows the advantages of interpreting an observation in its context and the exploitation of background knowledge. In real cases, more complex situations can be detected using the proposed approach: for example, as an extension to the previous case, it is possible to interpret the anomaly of temperature P_Y regarding the average of the values of temperature P_X measured during a certain time window w, instead of only considering if one of these values is greater than the threshold T_X. The corresponding C-SPARQL query is displayed in Listing 6.3.

```
REGISTER QUERY sit-AbnormalT-M1M2 AS
PREFIX :<http://semanticweb.org/STEaMINg/ContextOntology-COInd4#>
PREFIX f: <http://larkc.eu/csparql/sparql/jena/ext#>
SELECT ?o1 ?o2 ?m1 ?m2
FROM STREAM <Stream-sensorTM1> [RANGE 5s 1s]
FROM STREAM <Stream-sensorTM2> [RANGE 5s 1s]
WHERE {
    :sensorTM1 :isHostedBy ?m1 .
    :sensorTM1 :madeObservation ?o1 .
    ?o1 :hasSimpleResult ?res1 .
    ?o1 :hasTime ?t1 .
    ?s2 :isHostedBy ?m2 .
    ?s2 :madeObservation ?o2 .
    ?o2 :hasSimpleResult ?res2 .
    ?o2 :hasTime ?t2 .
    FILTER (
    f:timestamp(:sensorTM1,:madeObservation,?o1)
    < f:timestamp(?s2,:madeObservation,?o2)
    && ?res1 >= Ty
    && ?res2 >= Tx
    && ?s2 != sensorTM1
    && ?m1 = M1
    && ?m2 = M2 ) .
} ;
```

Listing 6.2 – C-SPARQL query to detect the described situation.

```
REGISTER QUERY sit-AbnormalT-M1M2-avg AS
PREFIX :<http://semanticweb.org/STEaMINg/ContextOntology-COInd4#>
SELECT ?o1 ?o2 ?m1 ?m2
FROM STREAM <Stream-sensorTM1> [RANGE 5s 1s]
FROM STREAM <Stream-sensorTM2> [RANGE 5s 1s]
WHERE {
    :sensorTM1 :isHostedBy ?m1 .
    :sensorTM1 :madeObservation ?o1 .
    ?o1 :hasSimpleResult ?res1 .
    ?o1 :hasTime ?t1 .
    ?s2 :isHostedBy ?m2 .
    ?s2 :madeObservation ?o2 .
    { SELECT ?s2 ( avg(?p2) AS ?average )
      WHERE {
        ?s2 :madeObservation ?o2 .
        ?o2 :hasSimpleResult ?res2 .
        ?o2 :hasTime ?t2 .
      }
      GROUP BY ?s2
      HAVING ( avg(?res2) >= Tx )
    }
    FILTER (
    ?res1 >= Ty
    && ?s2 != :sensorTM1 ) .
} ;
```

Listing 6.3 – C-SPARQL query considering the average of Py.

6.3 Conclusion

This chapter describes the framework introduced in Chapter 4. The main components of it are: (1) the Monitoring component that contains the Translation and Temporal rela-

tions modules to detect abnormal situations in real-time; (2) the Diagnosis component, that has only one module called Cause Determination, to determine the possible cause(s) that generate an abnormal situation; (3) the Situation refinement component, that contains the Update/Create situation module that helps experts in refining already defined situations or in identifying new ones; and (4) the Decision Making component, which is not part of the task of detecting situations or determining the possible causes, but exploits this information to support decision making. All these components rely on the ontological model described in Chapter 5 to semantically enrich data representation and processing.

Our approach uses stream reasoning to detect situations that could lead to failures timely, to make the most appropriate decision to avoid the interruption of the manufacturing processes. The use of stream reasoning allows the integration of data from different data sources, with different underlying meanings, different temporal resolutions as well as the processing of these data in real-time. Furthermore, our proposal also uses classic reasoning approaches to complete the determination of causes that could not be obtained by stream reasoning.

The main limitation of the proposed approach is that not all situations associated with failures are known in advance. Therefore, if these failures happen they are not detected. It is thus required to consult domain experts for decisions about these undetected failures. The domain experts asses the current state of the production system and provide appropriate decisions that can be further capitalized.

In this way, new queries capitalizing the experts' experience need to be registered to the stream reasoner to update the initial set of queries. Thus, when in the future a similar situation needs to be detected, the updated set of queries will detect that situation. The abnormal situation refinement component helps the experts in the creation of these new queries that represent the identified situations, as explained in this chapter.

To summarize, an approach that allows to detect high-level situations from low-level context and sensor information is proposed in this chapter. As already discussed, detecting situations can trigger actions to adapt the process behavior, and this change in behavior can lead to the generation of new situations. These situations can have different levels of severity, and can be nested in different ways. Knowing what potential situations may arise after application of a certain action can help to support decision making. Therefore, the following chapter presents a method to build a hierarchy of all the possible situations depending on the constraints they rely on.

Chapter 7

Situation Hierarchies for supporting Decision Making

Contents

As mentioned in Chapter 4, manufacturing processes are not always executed under optimal conditions. Nevertheless, they can sometimes continue their execution in degraded conditions without being completely stopped. Expert knowledge enables to describe these *"intermediate"* manufacturing conditions. The associated abnormal situations can have different levels of severity, and be nested in different ways. They can impact other processes or resources that participate in these processes and they can trigger other situations that represent a risk of major interruption of the manufacturing process or a risk of accident.

Once a situation that may lead to failures is detected, decisions must be made, such as whether the process should be interrupted or continued under sub-optimal conditions. In order to support the Decision making component of our approach, it is relevant to consider which other situations can be reached from the current situation, to choose the most appropriate action.

As mentioned in section 5.1.6, a situation is expressed as a combination of constraints. In this chapter we propose an approach to establish an order among the situations. The order among the situations is a hierarchy that depends on how situations' constraints are correlated. This order represents a road-map of all the situations, desirable or undesirable, that can be reached from a given one. In this way, it is possible to identify the actions that can be taken to correct the abnormality, considering that certain actions can modify the value of a property and thus change the state of the system, either by satisfying another constraint or, on the contrary, by not satisfying constraints anymore.

The proposed approach uses the lattice theory [Nat98, Dav02, Gra11] to provide an expressive formalization to order the situations in a taxonomic way. In this way, the hierarchy of situations is formally extracted from the situations definitions. The remainder of the chapter is structured as follows: section 7.1 extends the state of the art described in Chapter 3, it presents related work for situation ordering. In section 7.2, the approach for building the lattice is introduced, providing the definitions for the construction of the hierarchy of situations. In section 7.3, the interpretation and exploitation of the lattice to support the decision making is detailed. In Section 7.4, we present some concluding remarks.

7.1 Related work

In this section, we extend the literature review, presented in Chapter 3, with focus on other approaches and research fields that are related to the notion of order among situations of interest.

Most of the proposals coming from the Complex Event Processing (CEP) community [CM12, BDG+07, WDR06] offer relatively simple languages for representing complex events. However, these modeling proposals do not support the definition of complex events hierarchies. To overcome this limitation, TESLA [CM10], an event specification language for CEP, supports content-based event filtering and allows to capture relations among temporally related patterns of events. With the same goal as TESLA but coming

from the ontology research field, in [TBDV+17] the authors present a syntax for Description Logic Event Processing (DELP), permitting reasoning.

Other approaches to define, structurally, the event type composition in cyber-physical systems (CPS) adopt the theory of concept lattice [Wil05]. In [TVG+10], a concept lattice-based event model for CPS is presented. With this model, a CPS event is uniformly represented by three components: the event type, its internal attributes, and its external attributes. The internal and external attributes together characterize the event type. The model allows events to be composed across different components and devices within and among both the cybernetic and physical domains.

Other approach proposes the use of rule-based models such as Adaptive Neuro Fuzzy Interference Systems (ANFIS) models for monitoring wind turbine SCADA (Supervisory Control and Data Acquisition) signals [SS14, SSA13]. In order to obtain turbine condition statements, the authors implement rules given by an expert who is familiar with the behavior of the turbine, typical faults and their root causes. There are two types of rules: generic rules used to highlight anomalies, and specific rules providing specific condition or potential root causes. In this case, it is possible to determine whether one antecedent is contained in another or not. However, there is not a direct way to determine that an antecedent or situation is partially occurring or that other constraints must be satisfied for another situation to occur.

Understanding the relations among situations, with an order among them, allows the choice of the most effective actions to overcome the effects of abnormal situations. In most of the solutions mentioned above and in section 3.2.2, it is possible to observe a hierarchy of events in the sense that complex events are composed of simple ones, but this hierarchy does not directly provide the information that certain complex events may occur partially in other complex events or that they share certain simple events or not. To the best of our knowledge, no works have been published to establish a formal representation of a hierarchy among (abnormal) situations.

The approach presented in this chapter is a method that allows to order not only the complex events already defined based on the simple events that compose them, but also to order certain combinations of simple events that do not completely compose a complex event. Additionally, it is possible to consider occurrences of simple events that imply the occurrence of other simple events for the construction of the situation hierarchy.

7.2 Situation hierarchy

As evoked before, analysing the relations among situations, with an order among them, allows the choice of the most effective actions to overcome the effects of abnormal situations. These actions can, *in fine*, adapt the maintenance schedule or lead to take further measures to prevent unexpected downtime. The idea driving this proposal is to formally represent a hierarchy among situations leading to failures.

In this section, we firstly introduce the definitions that are necessary for the construction of the hierarchy of situations, and secondly, we describe the steps to automatically

build this hierarchy from the situations definitions and its associated lattice.

7.2.1 Definitions

In order to formally represent a hierarchy of situations, let us consider the following structure $\langle \mathscr{S}, \mathscr{C}, \mathscr{R}, \mathscr{T} \rangle$ where:

- $\mathscr{S} = \{s_1, s_2, s_3, \ldots, s_n\}$ is the set of all the situations,
- $\mathscr{C} = \{c_1, c_2, c_3, \ldots, c_m\}$ is the set of all the constraints,
- $\mathscr{R} \subseteq \mathscr{S} \times \mathscr{C}$ is a binary relation that links a situation with a constraint, and
- $\mathscr{T} \subseteq \mathscr{C} \times \mathscr{C}$ is a binary relation that links a constraint with another constraint.

It is worth mentioning again two aspects about situations that were defined in section 5.1.6. The first one is that a situation defines an abstract state of affairs that represents a particular scenario of interest and involves observations linked through spatio-temporal relationships, resources and processes. The second one is that situations are abstract, meaning that there may be different instances of a given situation. Instances of the same situation can happen during different periods of time and involve several resources, but they all satisfy the same constraints.

Set $\mathscr{C}$ contains all the constraints that are associated with one or more situations in the set of situations $\mathscr{S}$. These constraints concern properties of the processes, machines, resources and of the environment in which the tasks are executed. For example, if we consider variables `MC1_Temp` and `MC2_Temp`, corresponding to the temperature of a component of a machine and the temperature of another component of the same machine, then `MC1_Temp` < 40°C and `MC1_Temp>MC2_Temp` are constraints defined on them.

The binary relation $\mathscr{R}$ is used to establish that a constraint is involved in a situation. We write $s_1 \mathscr{R} c_1$ to indicate that the constraint c_1 is involved in the situation s_1. The $\mathscr{R}$ relation is therefore built from the relationships between the situations and the constraints that are extracted from expert knowledge.

The sets $\mathscr{S}$ and $\mathscr{C}$ correspond to the `Situation` class and the `Constraint` class of our ontological model (see Chapter 5), respectively. The association between a situation and its constraints is represented through the `hasConstraint` relation in the ontological model, represented by the binary relation $\mathscr{R}$.

The example below is used to illustrate the definition of the structure and the operators. Let us consider the following sets of constraints and of situations, with six constraints and six situations, respectively:

$$\mathscr{C} = \{c_1, c_2, c_3, c_4, c_5, c_6\} \text{ and } \mathscr{S} = \{s_1, s_2, s_3, s_4, s_5, s_6\}.$$

The corresponding $\mathscr{R}$ relation, built from $\mathscr{S}$ and $\mathscr{C}$, is shown in the first two columns of Table 7.1. For example, the fourth line in this table indicates that situation s_4 happens (first column) if constraints c_2 and c_5 are satisfied (second column). Another way to show relation $\mathscr{R}$ is depicted in Figure 7.1. In this figure, we can observe how different situations

Table 7.1 – Example of situations and associated constraints (in bold face the constraints that are implied by other constraints)

Situations	Constraints	$\mathscr{T}$ Constraints
s_1	c_1, c_3, c_6	c_1, $\mathbf{c_2}$, c_3, $\mathbf{c_4}$, $\mathbf{c_5}$, c_6
s_2	c_1, c_4, c_6	c_1, $\mathbf{c_2}$, c_4, $\mathbf{c_5}$, c_6
s_3	c_2, c_4, c_6	c_2, c_4, $\mathbf{c_5}$, c_6
s_4	c_2, c_5	c_2, c_5
s_5	c_3, c_6	c_3, $\mathbf{c_4}$, $\mathbf{c_5}$, c_6
s_6	c_1, c_6	c_1, $\mathbf{c_2}$, c_6

share a part of the constraints in their definition. For example, s_1 and s_2 share constraints c_1 and c_6.

Some constraints can be more general than others, *i.e.* include others. Such is the case with, for example, constraints c_1 and c_2 defined as `MC1_Temp` < 40°C and `MC1_Temp` < 60°C, respectively. If c_1 is satisfied, then c_2 is necessarily also satisfied. Furthermore, some constraints may imply other constraints due to physical properties extracted from expert knowledge or observations. For this reason, the $\mathscr{T}$ relation is defined to indicate that if a constraint is satisfied, then another one is also satisfied. We write $c_1 \mathscr{T} c_2$.

Consider the implications among the constraints from the set of all constraints $\mathscr{C}$ shown in Figure 7.1 (arrows labeled with $\mathscr{T}$). These implications are inferred from the order relations (<,>,≤,≥), from observations, or extracted from expert knowledge. Taking into account these relations, the constraints associated with each situation are established as shown in Table 7.1 (third column). This allows to associate both explicit and implicit (implied) constraints to a situation.

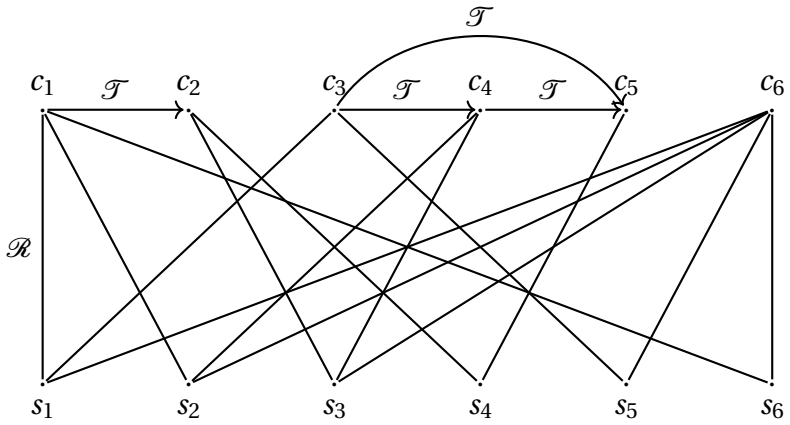

Figure 7.1 – Situations with constraints ($\mathscr{R}$) and implications among the constraints ($\mathscr{T}$).

In order to extend the formalization between a situation and a constraint to a set of situations and a set of constraints, two operators are defined below, based on the use of both $\mathscr{R}$ and $\mathscr{T}$ relations. These operators are defined on a set of situations or constraints because each situation can involve several constraints, and several situations can have constraints in common.

The first operator $\lceil \mathscr{X} \rceil$ enables the retrieval of the set of constraints associated to a set of situations.

Definition 4 *For a situation set $\mathscr{X}$, $\mathscr{X} \subseteq \mathscr{S}$, let*

$$\lceil \mathscr{X} \rceil := \{c \in \mathscr{C} | \forall x \in \mathscr{X} : x \mathscr{R} c \vee \exists c' \in \mathscr{C} : x \mathscr{R} c' \wedge c' \mathscr{T} c\}$$

Considering situations s_1 and s_2 of the example presented above, the constraints related to both situations are $\lceil \{s_1, s_2\} \rceil = \{c_1, c_2, c_4, c_5, c_6\}$.

The second operator $\lfloor \mathscr{Y} \rfloor$ conversely enables the retrieval of the set of situations involving a set of constraints $\mathscr{Y}$.

Definition 5 *For a constraint set $\mathscr{Y}$, $\mathscr{Y} \subseteq \mathscr{C}$, let*

$$\lfloor \mathscr{Y} \rfloor := \{s \in \mathscr{S} | \forall y \in \mathscr{Y} : s \mathscr{R} y \vee \exists c' \in \mathscr{C} : s \mathscr{R} c' \wedge c' \mathscr{T} c\}$$

This operator retrieves all the situations involving at least all the constraints in the set $\mathscr{Y}$. Therefore, considering the example presented above, if the constraints c_2 and c_5 are considered, then the situations involving those constraints are $\lfloor \{c_2, c_5\} \rfloor = \{s_1, s_2, s_3, s_4\}$. Let us note that these situations may involve other constraints, *e.g.* s_3 with c_4 and c_6.

7.2.2 The lattice construction

Using the elements of the structure $\langle \mathscr{S}, \mathscr{C}, \mathscr{R}, \mathscr{T} \rangle$ and the two operators $\lceil . \rceil$ and $\lfloor . \rfloor$ previously defined, the construction of the lattice representing the hierarchy of situations is detailed below.

In our approach, we group situations and their constraints as ordered pairs $(\mathscr{X}, \mathscr{Y})$ where $\mathscr{X}$ is a set of situations and $\mathscr{Y}$ is a set of constraints such that $\lceil \mathscr{X} \rceil = \mathscr{Y}$ and $\lfloor \mathscr{Y} \rfloor = \mathscr{X}$. The first component of the pair is a set including all the situations that share all the constraints belonging to the second component. The second component is therefore the set including all the common constraints among the situations of the first component.

In order to find all the pairs and thus the nodes of the lattice, given a set of situations $\mathscr{S}$, a set of constraints $\mathscr{C}$, and relations $\mathscr{R}$ and $\mathscr{T}$, Algorithm 1 is applied. Firstly, $\lceil \{s\} \rceil$ is computed for each situation $s \in \mathscr{S}$ (lines 3-5). Then, for any two sets in this set of sets ($consSet$), their intersection is calculated. If this intersection is not yet contained in $consSet$, it is added to it (lines 6-12). This step is repeated until no new sets are generated. If set $\mathscr{C}$ of all the constraints and the empty set ({}) are not in $consSet$, they are also added to it (lines 13-18). These two sets yield the minimum and maximum nodes of the lattice, respectively. Finally, for every set $\mathscr{O}$ in $consSet$, $\lfloor \mathscr{O} \rfloor$ is computed (lines 19-21). In the end, set ($setofPairs$) of all the $(\mathscr{X}, \mathscr{Y})$ pairs that satisfy $\lceil \mathscr{X} \rceil = \mathscr{Y}$ and $\lfloor \mathscr{Y} \rfloor = \mathscr{X}$ is obtained.

Considering the example in Figure 7.1 and Table 7.1, if the set of situations $\mathscr{S}$ and the set of constraints $\mathscr{C}$ are given as input to the algorithm, then the output are the nodes of the lattice shown in Figure 7.2. Let us note that since the situations are predefined, the search for all pairs can be done offline. If a situation needs to be added, it is either added

Algorithm 1 Calculate all the pairs $(\mathscr{X},\mathscr{Y})$ where $\mathscr{X} \subseteq \mathscr{S}$, $\mathscr{Y} \subseteq \mathscr{C}$, $\lceil \mathscr{X} \rceil = \mathscr{Y}$ and $\lfloor \mathscr{Y} \rfloor = \mathscr{X}$ ($setofPairs$)

Require: a set of Situations $\mathscr{S}$ and a set of Constraints $\mathscr{C}$
Ensure: $\{(\mathscr{X},\mathscr{Y}) | \mathscr{X} \subseteq \mathscr{S} \wedge \mathscr{Y} \subseteq \mathscr{C} \wedge \lceil \mathscr{X} \rceil = \mathscr{Y} \wedge \lfloor \mathscr{Y} \rfloor = \mathscr{X}\}$

$consSet \leftarrow \{\}$ //$consSet$ *is a set of constraint sets*
$setofPairs \leftarrow \{\}$
for all $s \in \mathscr{S}$ **do**
  $consSet \leftarrow consSet \cup \{\lceil \{s\} \rceil\}$
end for
for all $\mathscr{O}_1 \in consSet$ **do**
  for all $\mathscr{O}_2 \in consSet$ **do**
    if $\mathscr{O}_1 \cap \mathscr{O}_2 \notin consSet$ **then**
      $consSet \leftarrow consSet \cup \{\{\mathscr{O}_1 \cap \mathscr{O}_2\}\}$
    end if
  end for
end for
if $\mathscr{C} \notin consSet$ **then**
  $consSet \leftarrow consSet \cup \{\mathscr{C}\}$
end if
if $\{\} \notin consSet$ **then**
  $consSet \leftarrow consSet \cup \{\{\}\}$
end if
for all $\mathscr{O} \in consSet$ **do**
  $setofPairs \leftarrow setofPairs \cup \{(\lfloor \mathscr{O} \rfloor, \mathscr{O})\}$
end for

to a node or a new node is created. The addition of a new situation to the hierarchy has an impact on the structure of the hierarchy.

Once all the pairs are found, the next step is to order them in a lattice to build the hierarchy of situations.

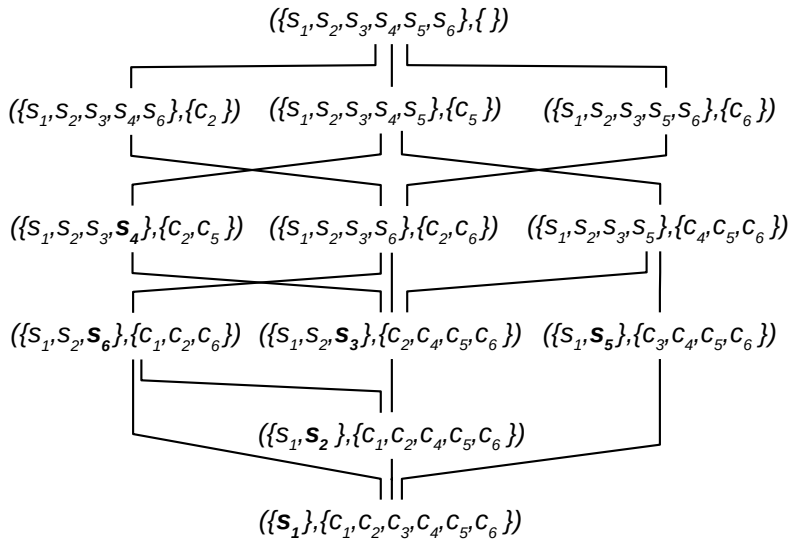

Figure 7.2 – Lattice representing the hierarchy of situations (the situations defined exactly by all the constraints of the second component of the pair in the node are shown in bold face)

A **lattice** is an algebraic structure that consists of a partially ordered set in which every two elements have a unique *supremum* (also called *least upper bound* or *join*) and a unique *infimum* (also called *greatest lower bound* or *meet*). A **partial order** is a pair $(\mathscr{P}, \preceq)$ where $\mathscr{P}$ is a set and $\preceq$ is a binary relation over $\mathscr{P}$ where $\preceq$ is reflexive, anti-symmetric and transitive.

For our lattice, we consider $\mathscr{L} = \{(\mathscr{X}, \mathscr{Y}) | \mathscr{X} \subseteq \mathscr{S} \wedge \mathscr{Y} \subseteq \mathscr{C} \wedge \lceil \mathscr{X} \rceil = \mathscr{Y} \wedge \lfloor \mathscr{Y} \rfloor = \mathscr{X}\}$ and the following binary relation defined over it:

Definition 6 *Let $(\mathscr{X}, \mathscr{Y})$ and $(\mathscr{X}', \mathscr{Y}')$ be two pairs where $\mathscr{X}$, $\mathscr{X}'$ are sets of situations and $\mathscr{Y}$, $\mathscr{Y}'$ are sets of constraints. The situations in $\mathscr{X}'$ are reachable from $\mathscr{X}$ if the constraints in $\mathscr{Y} \cap \mathscr{Y}'$ are satisfied, noted as $(\mathscr{X}, \mathscr{Y}) \preceq (\mathscr{X}', \mathscr{Y}') \Leftrightarrow \mathscr{X} \subseteq \mathscr{X}' \wedge \mathscr{Y}' \subseteq \mathscr{Y}$.*

7.2.3 Lattice proof

The fact that the hierarchy is a lattice allows us to know the situations that can be reached from the current situation, knowing also the intermediate situations (this is inherited from the fact that a lattice is a partial order).

In addition, let us suppose that two situations are happening simultaneously and these situations are represented in two different nodes of the lattice. Each set of situations from those nodes have a larger common set of situations with the constraints that are common to all situations in both sets of the two nodes. Dually, each set of situations have a smaller common subset of situations, which comprises all the constraints that all the situations in both sets of the nodes have. This allows to know from two situations that are happening, which situations can be reached by the production system: situations that satisfy the smaller number of constraints between the two situations that are happening (*supremum*) or situations that satisfy the greater number of constraints in common between the two situations that are happening (plus some other constraint(s)) (*infimum*).

Theorem 1 *$(\mathscr{L}, \preceq)$ is a lattice.*

Proof: The proof is done in two parts. In the first part, it is proved that $(\mathscr{L}, \preceq)$ is a partially ordered set. In the second part, it is proved that for any two elements of the lattice they have a unique *supremum* and a unique *infimum*.

Part 1

For $(\mathscr{L}, \preceq)$ to be a partially ordered set, $\preceq$ must be reflexive, anti-symmetric and transitive.

- First it is proven that $\preceq$ is reflexive, i.e.

$$\forall (\mathscr{X}, \mathscr{Y}) \in \mathscr{L} \mid (\mathscr{X}, \mathscr{Y}) \preceq (\mathscr{X}, \mathscr{Y}).$$

  By the definition of $\preceq$,

$$(\mathscr{X}, \mathscr{Y}) \preceq (\mathscr{X}, \mathscr{Y}) \Leftrightarrow \mathscr{X} \subseteq \mathscr{X} \wedge \mathscr{Y} \subseteq \mathscr{Y}$$

Since each set is included in itself, $\preceq$ is reflexive.

- Now we prove that $\preceq$ is anti-symmetric, i.e.

$$\forall(\mathscr{X},\mathscr{Y}),(\mathscr{X}',\mathscr{Y}') \in \mathscr{L} \mid (\mathscr{X},\mathscr{Y}) \preceq (\mathscr{X}',\mathscr{Y}') \wedge (\mathscr{X}',\mathscr{Y}') \preceq (\mathscr{X},\mathscr{Y}) \Rightarrow (\mathscr{X},\mathscr{Y}) = (\mathscr{X}',\mathscr{Y}')$$

Let two pairs $(\mathscr{X},\mathscr{Y})$, $(\mathscr{X}',\mathscr{Y}')$ belong to $\mathscr{L}$, and

$$(\mathscr{X},\mathscr{Y}) \preceq (\mathscr{X}',\mathscr{Y}') \text{ and } (\mathscr{X}',\mathscr{Y}') \preceq (\mathscr{X},\mathscr{Y}).$$

Applying the definition of $\preceq$ on both terms of the conjunction we obtain respectively:

$$\mathscr{X} \subseteq \mathscr{X}' \wedge \mathscr{Y}' \subseteq \mathscr{Y} \text{ and } \mathscr{X}' \subseteq \mathscr{X} \wedge \mathscr{Y} \subseteq \mathscr{Y}',$$

Thus, on the one hand we have

$$\mathscr{X} \subseteq \mathscr{X}' \text{ and } \mathscr{X}' \subseteq \mathscr{X}$$

which means that $\mathscr{X} = \mathscr{X}'$, and on the other hand

$$\mathscr{Y}' \subseteq \mathscr{Y} \text{ and } \mathscr{Y} \subseteq \mathscr{Y}'$$

which means that $\mathscr{Y} = \mathscr{Y}'$.

As $\mathscr{X} = \mathscr{X}'$ and $\mathscr{Y} = \mathscr{Y}'$, we conclude that $(\mathscr{X},\mathscr{Y}) = (\mathscr{X}',\mathscr{Y}')$. Therefore, $\preceq$ is anti symmetric.

- Finally, we show that $\preceq$ is transitive, i.e.

$$(\mathscr{X},\mathscr{Y}) \preceq (\mathscr{X}',\mathscr{Y}') \wedge (\mathscr{X}',\mathscr{Y}') \preceq (\mathscr{X}'',\mathscr{Y}'') \Rightarrow (\mathscr{X},\mathscr{Y}) \preceq (\mathscr{X}'',\mathscr{Y}'').$$

Let $(\mathscr{X},\mathscr{Y})$, $(\mathscr{X}',\mathscr{Y}')$, $(\mathscr{X}'',\mathscr{Y}'')$ belong to $\mathscr{L}$, and consider that

$$(\mathscr{X},\mathscr{Y}) \preceq (\mathscr{X}',\mathscr{Y}') \text{ and } (\mathscr{X}',\mathscr{Y}') \preceq (\mathscr{X}'',\mathscr{Y}'').$$

Now, if we apply the definition of $\preceq$ on both terms of the conjunction we get,

$$\mathscr{X} \subseteq \mathscr{X}' \wedge \mathscr{Y}' \subseteq \mathscr{Y} \text{ and } \mathscr{X}' \subseteq \mathscr{X}'' \wedge \mathscr{Y}'' \subseteq \mathscr{Y}',$$

respectively. Considering sets $\mathscr{X}$, $\mathscr{X}'$, and $\mathscr{X}''$ we have

$$\mathscr{X} \subseteq \mathscr{X}' \text{ and } \mathscr{X}' \subseteq \mathscr{X}''$$

which means that $\mathscr{X} \subseteq \mathscr{X}''$ because $\subseteq$ is transitive. Similarly,

$$\mathscr{Y}'' \subseteq \mathscr{Y}' \text{ and } \mathscr{Y}' \subseteq \mathscr{Y}$$

which means that $\mathscr{Y}'' \subseteq \mathscr{Y}$.

As $\mathscr{X} \subseteq \mathscr{X}''$ and $\mathscr{Y}'' \subseteq \mathscr{Y}$ then we conclude that $(\mathscr{X},\mathscr{Y}) \preceq (\mathscr{X}'',\mathscr{Y}'')$. Therefore, $\preceq$ is transitive.

Part 2

For $(\mathscr{L}, \preceq)$ to be a lattice, any two elements of the lattice must have a unique *supremum* and a unique *infimum*. Firstly, three lemmas are introduced:

Lemma 1 *Let $\mathscr{Y}$ be a set of constraints and $\mathscr{X}$ a set of situations, then*

$$\mathscr{Y} \subseteq \lceil \lfloor \mathscr{Y} \rfloor \rceil \text{ and } \mathscr{X} \subseteq \lfloor \lceil \mathscr{X} \rceil \rfloor.$$

Proof: $\lfloor \mathscr{Y} \rfloor$ includes all the situations involving the constraints in set $\mathscr{Y}$ and the situations involving constraints that are included by the constraints in set $\mathscr{Y}$. Then, $\lceil \lfloor \mathscr{Y} \rfloor \rceil$ includes all the constraints associated with the set of situations $\lfloor \mathscr{Y} \rfloor$. Therefore, the set of constraints $\mathscr{Y}$ is contained in $\lceil \lfloor \mathscr{Y} \rfloor \rceil$.
The reasoning is the same for the set of situations $\mathscr{X}$. □

Lemma 2 *Let $\mathscr{Y}, \mathscr{Y}'$ be sets of constraints and $\mathscr{X}, \mathscr{X}'$ be sets of situations, then*

$$\mathscr{Y} \subseteq \mathscr{Y}' \Rightarrow \lfloor \mathscr{Y}' \rfloor \subseteq \lfloor \mathscr{Y} \rfloor \text{ and } \mathscr{X} \subseteq \mathscr{X}' \Rightarrow \lceil \mathscr{X}' \rceil \subseteq \lceil \mathscr{X} \rceil.$$

Proof: We prove that $\lceil \mathscr{X}' \rceil \subseteq \lceil \mathscr{X} \rceil$. By definition of $\lceil . \rceil$ on $\mathscr{X}'$,

$$\lceil \mathscr{X}' \rceil = \{c \in \mathscr{C} | \forall x \in \mathscr{X}' : x \mathscr{R} c \vee \exists c' \in \mathscr{C} : x \mathscr{R} c' \wedge c' \mathscr{T} c\}$$

This is for all $b \in \mathscr{X}'$, and by hypothesis $\mathscr{X} \subseteq \mathscr{X}'$. Therefore,

$$\begin{aligned} \lceil \mathscr{X}' \rceil &= \{c \in \mathscr{C} | \forall x \in \mathscr{X}' : x \mathscr{R} c \vee \exists c' \in \mathscr{C} : x \mathscr{R} c' \wedge c' \mathscr{T} c\} \\ &\subseteq \{c \in \mathscr{C} | \forall x \in \mathscr{X} : x \mathscr{R} c \vee \exists c' \in \mathscr{C} : x \mathscr{R} c' \wedge c' \mathscr{T} c\} = \lceil \mathscr{X} \rceil \end{aligned}$$

The same reasoning can be applied to prove $\mathscr{Y} \subseteq \mathscr{Y}' \Rightarrow \lfloor \mathscr{Y}' \rfloor \subseteq \lfloor \mathscr{Y} \rfloor$.
This lemma shows that the more constraints are required, the fewer situations involve all of them. Conversely, the more situations we consider, the fewer constraints they have in common. □

Lemma 3 *Let $\mathscr{Y}, \mathscr{Y}'$ be sets of constraints and $\mathscr{X}, \mathscr{X}'$ be sets of situations, then*

$$\lfloor \mathscr{Y} \rfloor \cup \lfloor \mathscr{Y}' \rfloor \equiv \lfloor \mathscr{Y} \cup \mathscr{Y}' \rfloor \text{ and } \lceil \mathscr{X} \rceil \cup \lceil \mathscr{X}' \rceil \equiv \lceil \mathscr{X} \cup \mathscr{X}' \rceil.$$

Proof: Let us consider $y \in \lfloor \mathscr{Y} \rfloor \cup \lfloor \mathscr{Y}' \rfloor$, we prove that $y \in \lfloor \mathscr{Y} \cup \mathscr{Y}' \rfloor$.
As mentioned before, if $y \in \lfloor \mathscr{Y} \rfloor \cup \lfloor \mathscr{Y}' \rfloor$, then $y \in \lfloor \mathscr{Y} \rfloor \vee y \in \lfloor \mathscr{Y}' \rfloor$. Applying the definition of $\lfloor . \rfloor$ to both terms of the disjunction, we obtain

$$\begin{aligned} &y \in \{s \in \mathscr{S} | \forall y \in \mathscr{Y} : s \mathscr{R} y \vee \exists c' \in \mathscr{C} : s \mathscr{R} c' \wedge c' \mathscr{T} c\} \vee \\ &y \in \{s \in \mathscr{S} | \forall y \in \mathscr{Y}' : s \mathscr{R} y \vee \exists c' \in \mathscr{C} : s \mathscr{R} c' \wedge c' \mathscr{T} c\} \end{aligned}$$

By the definition of the disjunction, we can rewrite the statement above as follows

$$y \in \{s \in \mathscr{S} | \forall y \in \mathscr{Y} \cup \mathscr{Y}' : s \mathscr{R} y \vee \exists c' \in \mathscr{C} : s \mathscr{R} c' \wedge c' \mathscr{T} c\}$$

Applying the definition of $\lfloor . \rfloor$ to the statement above, we get

$$y \in \lfloor \mathscr{Y} \cup \mathscr{Y}' \rfloor$$

Therefore, we can conclude that $\lceil \mathscr{Y} \rceil \cup \lceil \mathscr{Y}' \rceil \subseteq \lceil \mathscr{Y} \cup \mathscr{Y}' \rceil$.
Similarly, we can prove $\lceil \mathscr{Y} \cup \mathscr{Y}' \rceil \subseteq \lceil \mathscr{Y} \rceil \cup \lceil \mathscr{Y}' \rceil$ considering that $y \in \lfloor \mathscr{Y} \cup \mathscr{Y}' \rfloor$ and concluding that $y \in \lfloor \mathscr{Y} \rfloor \cup \lfloor \mathscr{Y}' \rfloor$.
The same reasoning can be applied to prove $\lceil \mathscr{X} \rceil \cup \lceil \mathscr{X}' \rceil \equiv \lceil \mathscr{X} \cup \mathscr{X}' \rceil$. □

Having defined the three lemmas and their respective proofs, we continue with the proof of the second part of the theorem.

For any two pairs $(\mathscr{X}, \mathscr{Y})$ and $(\mathscr{X}', \mathscr{Y}')$ we obtain:

- the *infimum* of $(\mathscr{X}, \mathscr{Y})$ and $(\mathscr{X}', \mathscr{Y}')$ as $(\mathscr{X}, \mathscr{Y}) \wedge (\mathscr{X}', \mathscr{Y}') := (\mathscr{X} \cap \mathscr{X}', \lceil \lfloor \mathscr{Y} \cup \mathscr{Y}' \rfloor \rceil)$
- the *supremum* of $(\mathscr{X}, \mathscr{Y})$ and $(\mathscr{X}', \mathscr{Y}')$ as $(\mathscr{X}, \mathscr{Y}) \vee (\mathscr{X}', \mathscr{Y}') := (\lfloor \lceil \mathscr{X} \cup \mathscr{X}' \rceil \rfloor, \mathscr{Y} \cap \mathscr{Y}')$

To prove the existence of the unique *infimum*, we have to show that $(\mathscr{X}, \mathscr{Y}) \wedge (\mathscr{X}', \mathscr{Y}')$ is smaller than both $(\mathscr{X}, \mathscr{Y})$ and $(\mathscr{X}', \mathscr{Y}')$, and any other common child of $(\mathscr{X}, \mathscr{Y})$ and $(\mathscr{X}', \mathscr{Y}')$ is also a child of $(\mathscr{X}, \mathscr{Y}) \wedge (\mathscr{X}', \mathscr{Y}')$.

First we prove that

$$(\mathscr{X} \cap \mathscr{X}', \lceil \lfloor \mathscr{Y} \cup \mathscr{Y}' \rfloor \rceil) \preceq (\mathscr{X}, \mathscr{Y}).$$

By the definition of $\preceq$, we have

$$\mathscr{X} \cap \mathscr{X}' \subseteq \mathscr{X} \text{ and } \mathscr{Y} \subseteq \lceil \lfloor \mathscr{Y} \cup \mathscr{Y}' \rfloor \rceil$$

From $\mathscr{Y} \subseteq \mathscr{Y} \cup \mathscr{Y}'$ applying Lemma 2 twice we obtain $\lceil \lfloor \mathscr{Y} \rfloor \rceil \subseteq \lceil \lfloor \mathscr{Y} \cup \mathscr{Y}' \rfloor \rceil$. In addition, Lemma 1 expresses that $\mathscr{Y} \subseteq \lceil \lfloor \mathscr{Y} \rfloor \rceil$. Thus, combining the last two statements we conclude that

$$\mathscr{Y} \subseteq \lceil \lfloor \mathscr{Y} \rfloor \rceil \subseteq \lceil \lfloor \mathscr{Y} \cup \mathscr{Y}' \rfloor \rceil.$$

The same method can be used to prove that $(\mathscr{X} \cap \mathscr{X}', \lceil \lfloor \mathscr{Y} \cup \mathscr{Y}' \rfloor \rceil) \preceq (\mathscr{X}', \mathscr{Y}')$

Now we prove that any other common child of $(\mathscr{X}, \mathscr{Y})$ and $(\mathscr{X}', \mathscr{Y}')$ is also a child of $(\mathscr{X}, \mathscr{Y}) \wedge (\mathscr{X}', \mathscr{Y}')$. Let us consider $(\mathscr{X}'', \mathscr{Y}'')$ a child of $(\mathscr{X}, \mathscr{Y})$ and $(\mathscr{X}', \mathscr{Y}')$. Then, by the definition of $\preceq$ we have

$$(\mathscr{X}'', \mathscr{Y}'') \preceq (\mathscr{X}, \mathscr{Y}) \Rightarrow \mathscr{X}'' \subseteq \mathscr{X} \wedge \mathscr{Y} \subseteq \mathscr{Y}''$$
$$(\mathscr{X}'', \mathscr{Y}'') \preceq (\mathscr{X}', \mathscr{Y}') \Rightarrow \mathscr{X}'' \subseteq \mathscr{X}' \wedge \mathscr{Y}' \subseteq \mathscr{Y}''.$$

$(\mathscr{X}'', \mathscr{Y}'')$ is a child of $(\mathscr{X}, \mathscr{Y}) \wedge (\mathscr{X}', \mathscr{Y}')$ iff (1) $\mathscr{X}'' \subseteq \mathscr{X} \cap \mathscr{X}'$ and (2) $\lceil \lfloor \mathscr{Y} \cup \mathscr{Y}' \rfloor \rceil \subseteq \mathscr{Y}''$.
The proof of (1) is trivial since by hypothesis we have $\mathscr{X}'' \subseteq \mathscr{X}$ and $\mathscr{X}'' \subseteq \mathscr{X}'$, then

$$\mathscr{X}'' \subseteq \mathscr{X} \cap \mathscr{X}'.$$

The proof of (2) is written below,

$$\left.\begin{array}{l} \mathscr{Y} \subseteq \mathscr{Y}'' \xrightarrow[\text{L. 2}]{} \lfloor \mathscr{Y}'' \rfloor \subseteq \lfloor \mathscr{Y} \rfloor \\ \mathscr{Y}' \subseteq \mathscr{Y}'' \xrightarrow[\text{L.2}]{} \lfloor \mathscr{Y}'' \rfloor \subseteq \lfloor \mathscr{Y}' \rfloor \end{array}\right\} \Rightarrow \lfloor \mathscr{Y}'' \rfloor \subseteq \lfloor \mathscr{Y} \rfloor \cup \lfloor \mathscr{Y}' \rfloor \xrightarrow[\text{L.3}]{} \lfloor \mathscr{Y}'' \rfloor \subseteq \lfloor \mathscr{Y} \cup \mathscr{Y}' \rfloor \xrightarrow[\text{L.2}]{} \lceil \lfloor \mathscr{Y} \cup \mathscr{Y}' \rfloor \rceil \subseteq \lceil \lfloor \mathscr{Y}'' \rfloor \rceil$$

Therefore, we proved that any common child of $(\mathscr{X},\mathscr{Y})$ and $(\mathscr{X}',\mathscr{Y}')$ is also a child of $(\mathscr{X},\mathscr{Y}) \wedge (\mathscr{X}',\mathscr{Y}')$.

The proof of uniqueness is done by supposing that there are two different infimums and then concluding that they are the same by applying the fact that $\preceq$ is anti-symmetric.

Similarly, $(\mathscr{X},\mathscr{Y}) \vee (\mathscr{X}',\mathscr{Y}')$ is greater than both $(\mathscr{X},\mathscr{Y})$ and $(\mathscr{X}',\mathscr{Y}')$, and it is a child of any common parent of these two pairs. Only the proof for the *infimum* is developed here since the proof for the *supremum* is similar. □

7.3 Case study for lattice interpretation and exploitation

An illustrative case study is described in this section to highlight how the lattice can be used and the advantages it offers to support decisions that need to be made when an abnormal situation is detected in an industrial framework.

The case study is based on a manufacturing production line composed of several machines. These machines are equipped with sensors on different components. The sensors collect data on the properties described in Table 7.2.

Several abnormal situations that could lead to failures in this scenario are defined by experts. Each of them is expressed as a set of constraints. In this case study, we focus on situations covering the following types of failures: hydraulic oil leakage, cooling system filter obstructions, converter and rotor malfunctions and global malfunctions of the production line. The defined abnormal situations are shown in Table 7.3, with a short description of what they represent. The constraints concerned by the abnormal situations are described along the associated properties in Table 7.2.

Some situations represent conditions that can lead to the same potential failure, indicating different levels of severity. For example, situations s_1 and s_2 both represent situations that could lead to the same failure but s_2 indicates a higher severity since the temperature threshold is higher than the temperature threshold for s_1. In this case, higher impact actions should be taken.

The lattice resulting from the scenario described above is shown in Figure 7.3. By observing the hierarchy of situations, in the upper part of the lattice the constraints which are involved in most of the situations are verified. Further down in the lattice, the situations are more specific as they include more constraints. The most specific constraints are close to the bottom of the diagram, meaning that these situations embed those that appear higher up in the lattice.

The lattice provides a structure that represents the order in which the situations may arise according to which constraints are verified. Considering that certain actions can modify the value of a property and thus change the state of the industrial system, either

Table 7.2 – Constraints definition.

Set of constraints $\mathscr{C}$			
ID	Properties	Restriction	Device
c_1	Oil temp.	$>40°C$	M_1
c_2	Oil temp.	$>60°C$	M_1
c_3	Transformer temp.	$>45°C$	M_1T_1
c_4	Controller temp.	$>40°C$	M_1Ct_1
c_5	Generator curr.	<800 A	M_1G_1
c_6	Platform temp.	$<35°C$	PL_1
c_7	Platform temp.	$>40°C$	PL_1
c_8	Gearbox temp.	$>40°C$	M_2GB_1
c_9	Gearbox temp.	$>60°C$	M_2GB_1
c_{10}	Generator speed	<500 rpm	M_2G_1
c_{11}	Environment temp.	$<25°C$	PL_1
c_{12}	Power output	>2000 kW	PL_1
c_{13}	Power output	<500 kW	PL_1
c_{14}	Power output	<200 kW	PL_1
c_{15}	Conv. water temp.	$>60°C$	M_3Cv_1
c_{16}	Conv. water temp.	$>80°C$	M_3Cv_1
c_{17}	Trans. grid temp.	$<35°C$	M_3T_1
c_{18}	Generator temp.	$>45°C$	M_3G_1
c_{19}	Converter temp.	$>60°C$	M_3Cv_1
c_{20}	Converter temp.	$>80°C$	M_3Cv_1
c_{21}	Rotor speed	<200 rpm	M_4R_1
c_{22}	Rotor speed	<100 rpm	M_4R_1
c_{23}	Rotor Pitch angle	$<5°$	M_4R_1

by satisfying another constraint or on the contrary by not satisfying a constraint anymore, the lattice allows the analysis of the actions to take. It implies, from the decision support point of view, reaching a node situated lower in the lattice if new constraints are satisfied, or higher in the other case.

For example, if constraints c_{15}, c_{17} and c_{18} are satisfied, it means that the situation s_6 is happening, *i.e.* there is a cooling filter obstruction in machine M_3. The lattice, and in particular, its node $(\{s_6, s_7, s_{10}\}, \{c_{15}, c_{17}, c_{18}\})$, shows that situation s_7, more critical than s_6, is reachable if no action is taken or if the action only imply the satisfaction of c_{16}. In general, the actions aim at correcting the abnormal property values causing them to return to normal values. In this case, a *filter change* action in the cooling system can make the values of properties `Converter water temp.` and `Generator temp.` decrease below the respective thresholds, *i.e.* the constraints c_{15} and c_{18} would no longer be satisfied. This would lead the process at node $(\{s_6, s_7, s_8, s_9, s_{10}\}, \{c_{17}\})$ where only constraint c_{17} is satisfied leading to a new state of the production system. Ideally, it is always intended to go up in the lattice to node $(\mathscr{S}, \{\})$, where none of the constraints are satisfied, meaning that no abnormal situation is present or partially present.

Table 7.3 – Situations and their concerned constraints (the constraints that are implied by other constraints are in bold face)

Set of situations $\mathscr{S}$		
Sit.	Const. ($\mathscr{T}$)	Description
s_1	c_1, c_3, c_4, c_5, c_6	$\mathtt{M}_1$ oil leakage
s_2	$\mathbf{c_1}, c_2, c_3, c_4, c_5, c_6$	$\mathtt{M}_1$ oil leakage
s_3	c_6, c_8, c_{10}, c_{11}	Increase $\mathtt{M}_2$ oil temp.
s_4	$c_6, \mathbf{c_8}, c_9, c_{10}, c_{11}$	Increase $\mathtt{M}_2$ oil temp.
s_5	$c_7, \mathbf{c_8}, c_9, c_{10}, c_{11}$	Increase $\mathtt{M}_2$ oil temp.
s_6	c_{15}, c_{17}, c_{18}	$\mathtt{M}_3$ filter obstruction
s_7	$\mathbf{c_{15}}, c_{16}, c_{17}, c_{18}$	$\mathtt{M}_3$ filter obstruction
s_8	c_6, c_{17}, c_{19}	$\mathtt{M}_3\mathtt{Cv}_1$ malfunction
s_9	$c_6, c_{17}, \mathbf{c_{19}}, c_{20}$	$\mathtt{M}_3\mathtt{Cv}_1$ malfunction
s_{10}	$\mathbf{c_{15}}, c_{16}, c_{17}, c_{18}, c_{19}$	$\mathtt{M}_3\mathtt{Cv}_1$ malfunction
s_{11}	c_{12}, c_{21}, c_{23}	$\mathtt{M}_4\mathtt{R}_1$ malfunction
s_{12}	$c_{12}, \mathbf{c_{21}}, c_{22}, c_{23}$	$\mathtt{M}_4\mathtt{R}_1$ malfunction
s_{13}	c_{13}, c_{21}, c_{23}	PL global malfunction
s_{14}	$\mathbf{c_{13}}, c_{14}, c_{21}, c_{23}$	PL global malfunction

A node in the lattice represents a possible state of the industrial system. It should be noted that the minimum of the lattice, node $(\{\}, \mathscr{C})$, may represent an unreachable state: Since this node includes all the constraints ($\mathscr{C}$), it may be that two constraints cannot be satisfied at the same time because they are exclusive, such as constraints c_6 and c_7. However, this node is necessary for the hierarchy to be a lattice. If this happens in another node of the lattice that it is not the minimum, then it denotes a problem in the definition of the $\mathscr{R}$ or the $\mathscr{T}$ relations, since it would be a situation that could never be satisfied because it concerns constraints that cannot be satisfied at the same time.

For each node defined as $(\mathscr{X}, \mathscr{Y})$, all situations in $\mathscr{X}$ are at least partially occurring, *i.e.* a part of their constraints is satisfied. When one of the situations in $\mathscr{X}$ involves exactly all the constraints in $\mathscr{Y}$ and no other, this means that this situation is happening. This is formally addressed in Definition 7 (these situations appear in bold face in Figures 7.2 and 7.3).

Definition 7 *For a pair* $(\mathscr{X}, \mathscr{Y})$, $\mathscr{X} \subseteq \mathscr{S}$ *and* $\mathscr{Y} \subseteq \mathscr{C}$, *let*

$$\|(\mathscr{X}, \mathscr{Y})\| := \{s \in \mathscr{X} | \forall\, y \in \mathscr{Y} : s \mathscr{R} y \wedge \nexists c' \in \mathscr{C} - \mathscr{Y} : s \mathscr{R} c'\}$$

For example, $\|(\{s_6, s_7, s_{10}\}, \{c_{15}, c_{17}, c_{18}\})\| = \{s_6\}$ means that situation s_6 happens in node $(\{\mathbf{s_6}, s_7, s_{10}\}, \{c_{15}, c_{17}, c_{18}\})$ because it only involves constraints c_{15}, c_{17} and c_{18}.

Regarding the nodes where $\|(\mathscr{X}, \mathscr{Y})\|$ is empty, this means that situations in $\mathscr{X}$ are partially occurring, so that they are potential situations that the system could reach. The discovery of these combinations of constraints in common among certain situations can give rise to the definition of new situations that allow the early stage detection of certain *relevant* situations, allowing preventive decisions to be made.

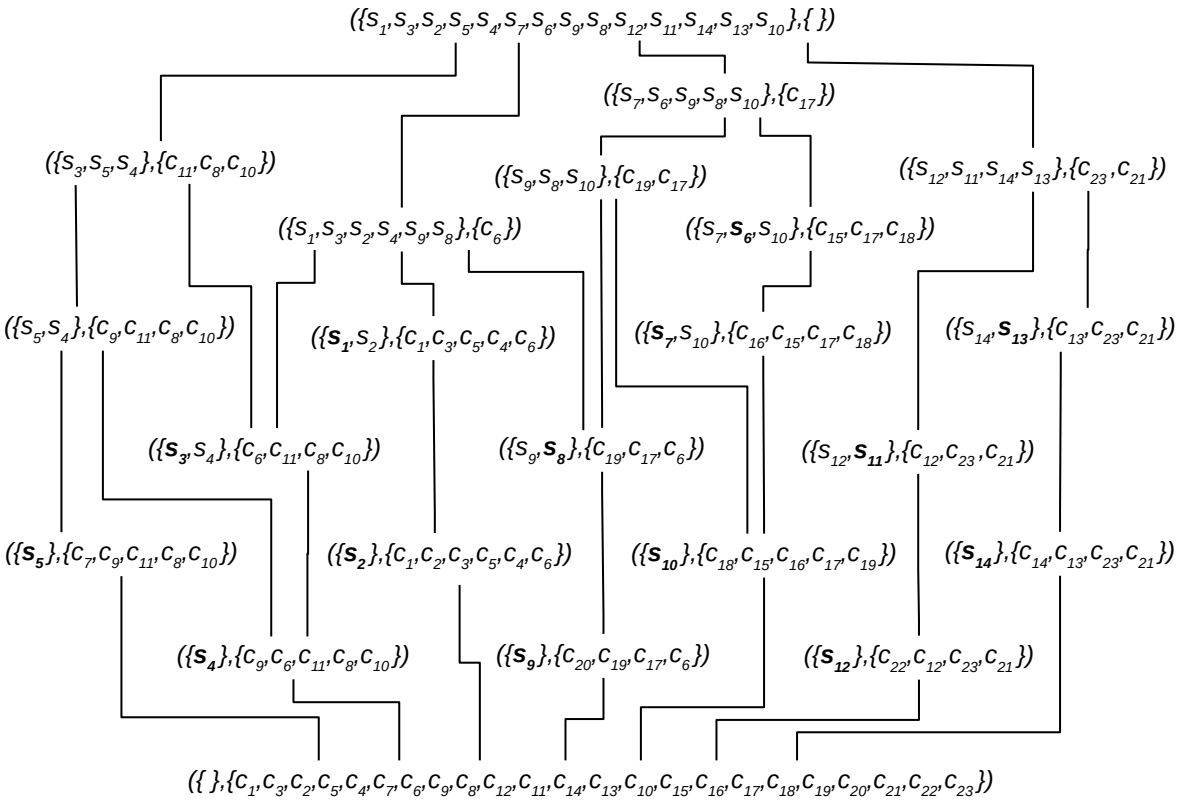

Figure 7.3 – Hierarchy of the situations defined in the illustrative case study.

7.4 Conclusion

This chapter presents an approach that uses the lattice theory to represent an order among situations that can lead to potential failures. This lattice is automatically built from existing knowledge coming from experts and from the ontological model. The lattice represents a road-map of all the situations that can be reached from a given one, desirable or not. This allows the identification of the actions that can be taken to correct a certain abnormal behavior of the process.

Although the case study presented in section 7.3 comes from the Industry 4.0 context, the proposed approach can be applied to other application domains, where real time monitoring is needed. In fact, in any monitoring application, the notions of normal or abnormal situations exist. Expert knowledge allows the description of these situations enabling then the building of the hierarchy for further use.

It is also to be remarked that in the presented case study, the $\mathscr{T}$ relation represents a hierarchy between the constraints in the sense that one constraint may be more specific than another, so that when the last one is satisfied, the more general constraint is also satisfied. The $\mathscr{T}$ relation can also represent another type of relationship between the constraints, such as the fact that two different properties have a physical link, meaning that the values of one directly affect the values of the other.

Chapter 8

Implementation of the proposed framework

Contents

In the previous chapters of Part II, the components of the proposed framework have been introduced. In Chapter 5, an ontological model is developed to represent the industrial domain and it is used as the foundation of a smart system. In Chapter 6, an hybrid semantic approach is proposed to automate abnormal situation detection, based on the combined use of classic reasoning and stream reasoning. In Chapter 7, a lattice-based approach is presented to represent a road-map of all the situations that can lead to potential failures, in order to support decision making tasks.

In order to apply our approach, we have developed a software prototype. The software uses deductive approaches, domain ontologies and ontology reasoning, and stream reasoning to analyze industrial data and to detect abnormal situations that can lead to failures.

This chapter is organized as follows. Section 8.1 describes the proposed framework design and implementation. The technologies and tools used for the development of the proposed framework are introduced as well as the implementation of the core functionalities of it. A proof of concept is presented in section 8.2 through an illustrative case study from the industrial domain. Section 8.3 describes the limits of the prototype and it also presents some concluding remarks.

8.1 Implementation of the framework components

For the development of each component of the proposed framework, several software and tools are used. As mentioned in Chapter 4, the proposed framework is a smart system in which each component can also be seen as a smart system. Figure 8.1 shows the core components and technologies used to implement the aforementioned framework.

The proposed framework is implemented in Java 1.8. The main technologies used are C-SPARQL[1], OWLAPI[2] and SWRLAPI[3]. The OWLAPI is a Java API for creating, manipulating and serialising OWL Ontologies. The SWRLAPI is also a Java API for working with the SWRL language. The SWRLAPI uses the OWLAPI to manage OWL ontologies and provides a Drools-based SWRL rule engine implementation to execute SWRL rules. C-SPARQL is both a query language to process RDF streams and an engine that provides continuous query capabilities. It supports timestamped RDF triplets as input and uses a periodic execution strategy to continuously execute queries over these RDF streams. It has the capability of integrating both RDF streams and static background knowledge, also represented as RDF triplets. Given that streams are intrinsically infinite, data are usually read through time windows using the CQL window concept [ABW04]: queries are executed on all the triplets which happen during a given time interval.
In C-SPARQL, continuous queries are divided into static and dynamic parts and streaming data is transformed into non-streaming data within a specified window in order to apply standard algebraic operations, such as aggregate functions like `COUNT`, `COUNT DISTINCT`,

[1] http://streamreasoning.org/resources/c-sparql
[2] https://github.com/owlcs/owlapi
[3] https://github.com/protegeproject/swrlapi

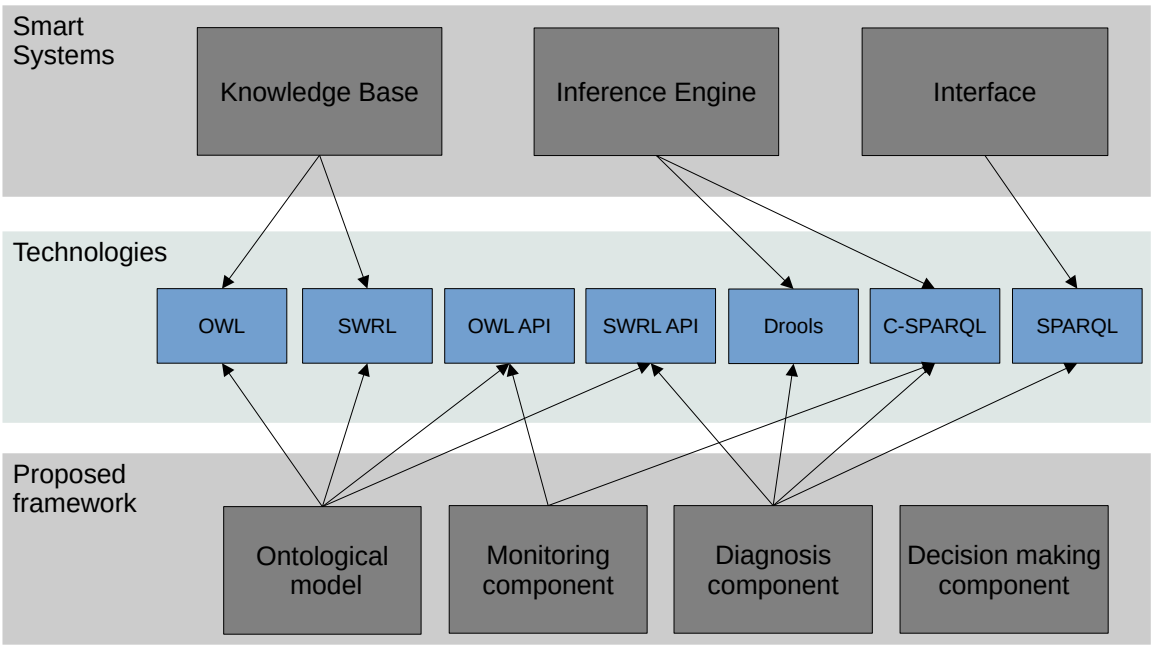

Figure 8.1 – Implementation technology for the proposed framework.

`MAX`, `MIN` and `AVG`. The static parts are loaded into relations, and the continuous queries are executed by processing the stream data against these relations.

Different SSN-based domain ontologies can be loaded in the prototype as well as the appropriate C-SPARQL queries and SWRL rules. Furthermore, different SPARQL-like query processing engines coupled with different rule languages such as CQELS can be integrated. The system is open source and can be found here[4].

8.1.1 The Ontological model implementation

In this section, we briefly describe the technologies used to implement the ontological model. It is composed by:

- The classes, properties and constraints that define the domain concepts,
- A set of ontological assertion axioms that describe the instances for particular applications,
- A set of if-then rules to infer new facts.

The ontology is written in OWL2, a sound and complete language with a formal semantic based in SROIQ Description Logic (presented in section 2.2.2) and is available here [5]. The OWLAPI is used to write the ontological statements. Rules expressed as OWL axioms are limited by some restrictions that enable the language to be decidable. Thus, SWRL has been considered to formulate the rules, and SWRLAPI is used to implement them in the ontological model. In addition, Protégé [6] and VOWL [7] are used for visualiza-

[4]https://gitlab.insa-rouen.fr/fgiustozzi/STEaMINg-SR_SitDet
[5]https://gitlab.insa-rouen.fr/fgiustozzi/STEaMINg-Ontology
[6]https://protege.stanford.edu/
[7]http://vowl.visualdataweb.org/webvowl.html

tion of the ontology.

8.1.2 The Monitoring component implementation

The Monitoring component, in charge of situation detection, is composed of the following modules:

Translation module

This module contains two sub-modules called *Stream Generator* and *Instance creator*. These sub-modules are used to convert the data collected from the sensors into RDF streams and insert them into the ontological model as instances.

- The *Stream Generator* sub-module streams out semantic enriched data streams that are then consumed by the *Stream Reasoner*. The output streams are RDF streams. The semantic enrichment of the acquired data is done by using the concepts and relations among them as defined in the ontological model. The C-SPARQL engine provides methods to generate streams. For this sub-module we focus on a default stream generator that produces simulated sensor observations for different parameters such as temperature, speed and vibration. The flow of observations begins by generating new observations at the desired frequency, which is indicated by a parameter. Listing 8.1 shows the code to generate the streams of the RDF graph shown in Figure 8.2 and the results are shown in Listing 8.2.

```
RdfQuadruple q = new RdfQuadruple("Sensor", "madeObservation", "O",
                                  System.currentTimeMillis());
this.put(q);
q = new RdfQuadruple("O","observedProperty","Temp", System.currentTimeMillis());
this.put(q);
q = new RdfQuadruple("O","hasSimpleResult", v + "^^http://www.w3.org/2001/XMLSchema#integer",
                     System.currentTimeMillis());
this.put(q);
```

Listing 8.1 – Code to generate RDF streams.

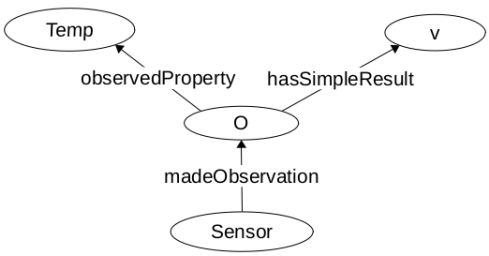

Figure 8.2 – RDF graph example.

```
Sensor madeObservation O . (1599636552842)
O      observedProperty Temp . (1599636552911)
O      hasSimpleResult v^^http://www.w3.org/2001/XMLSchema#integer . (1599636552968)
```

Listing 8.2 – RDF stream example (RDF triplet . timestamp).

- The *Instance creator* uses the OWLAPI to interact with the ontological model. This sub-module populates the ontological model with observations and their corresponding metadata, such as the sensors which made the observation, the observed property and the timestamp. Listing 8.3 shows the methods, from the OWLAPI, to add the instances and relations from the Figure 8.2 to the ontology. First, the instances (`Sensor`, `O`, `Temp`) are obtained from the ontology, or created in case they do not exist in the ontology. Second, it is indicated to which class of the ontology they belong. Third, the corresponding relationships between the instances are established. Finally, the above-mentioned is added to the ontology.

```
OWLIndividual sensor = factory.getOWLNamedIndividual(IRI.create(ns,"Sensor"));
OWLIndividual o = factory.getOWLNamedIndividual(IRI.create(ns,"O"));
OWLIndividual p = factory.getOWLNamedIndividual(IRI.create(ns,"Temp"));

OWLClassAssertionAxiom sensorType = factory.getOWLClassAssertionAxiom(Sensor, sensor);
OWLClassAssertionAxiom oType = factory.getOWLClassAssertionAxiom(Observation, o);
OWLClassAssertionAxiom pType = factory.getOWLClassAssertionAxiom(ObservableProperty, p);

OWLObjectPropertyAssertionAxiom madeobs =
               factory.getOWLObjectPropertyAssertionAxiom(madeObservation, sensor, o);
OWLObjectPropertyAssertionAxiom oP =
               factory.getOWLObjectPropertyAssertionAxiom(observedProperty,o,p);
OWLDataPropertyAssertionAxiom ohasSimpleR =
               factory.getOWLDataPropertyAssertionAxiom(hasSimpleResult,o,
                              v+"^^http://www.w3.org/2001/XMLSchema#double");

ontology.add(sensorType);
ontology.add(obsType);
ontology.add(pType);
ontology.add(madeobs);
ontology.add(oP);
ontology.add(ohasSimpleR);
```

Listing 8.3 – Code to add an observation and its metadata into the ontology.

Temporal Relations module

This module contains the *Stream Reasoner* sub-module which is implemented using C-SPARQL. This module explores the data streams to generate new information. For the engine to process data streams, they must be registered in the engine. This is done using the `registerStream(stream)` method provided by the C-SPARQL engine.

A set of queries, which represent particular situations, are registered and executed by the C-SPARQL engine over the data streams. This is done using the `registerQuery(query)` method provided by the C-SPARQL engine. These queries combine background knowledge extracted from the ontology and some parts of the streams that are relevant. They include mainly temporal dependencies between observations and anomalies. C-SPARQL represents queries as query graphs, with query evaluation performed through graph pattern matching over the graphs formed by the incoming data streams. It offers two types of processing models: either the query evaluation is periodic through windows of time, or the query evaluation is not periodic but triggered by the arrival of new triplets.

A simple C-SPARQL query is shown in Listing 8.4. The query name is registered on line

1 and prefixes used in the query are declared on lines 2 and 3. The query runs against the input RDF streams in the time frame of 15 seconds, sliding the window by 5 seconds (line 5). The chosen time frame is arbitrary and can be changed as desired. It produces pairs of values (line 4): the sensor name (`?s`) and the average value of the observations (`?avg`). To get the observation's value, `?o` individuals are bound with the data value `?v` through `sosa:hasSimpleResult` property (line 7-8). The query uses aggregate functions such as `AVG` (line 4) to calculate the average value of observations which are grouped by sensor name (line 10). Finally, the list of output pairs are filtered out to include only the ones where the average of the observation's values is greater than `T` (line 11).

```
1  REGISTER QUERY reasoning AS
2  PREFIX : <http://semanticweb.org/STEaMINg/ContextOntology-COInd4#>
3  PREFIX sosa: <http://www.w3.org/ns/sosa/>
4  SELECT ?s ( avg(?v) AS ?avg )
5  FROM STREAM <Stream> [RANGE 15s STEPS 5s]
6  WHERE {
7    ?s   sosa:madeObservation ?o .
8    ?o   sosa:hasSimpleResult ?v .
9  }
10 GROUP BY ?s
11 HAVING ( avg(?v) > T )
```

Listing 8.4 – Code to link `M1` and `M2` with `SIT`.

The *Stream Reasoner* sub-module produces situations streams as output. This output feeds another stream reasoner in the Cause determination module and the detected situations are stored in the ontology together with other relevant information, for example the resources concerned by the detected situation. As evoked in section 2.3.2, C-SPARQL supports closed-world and time-aware reasoning on stream data. However, it is not intended to have any effect on the underlying ontology. Therefore, in order to store the detected situation together with other relevant information, this sub-module invokes OWLAPI constructs for asserting new OWL individuals holding all situation information. Listing 8.5 shows an example of the OWLAPI constructors used to add to the ontology the facts that two machines (`M1` and `M2`) are concerned by a detected situation (`SIT`). Figure 8.3 (a) shows the relevant part of the ontology for this example before detecting the situation, while Figure 8.3 (b) shows the ontology with the instance of the detected situation as well as the machines concerned by that situation.

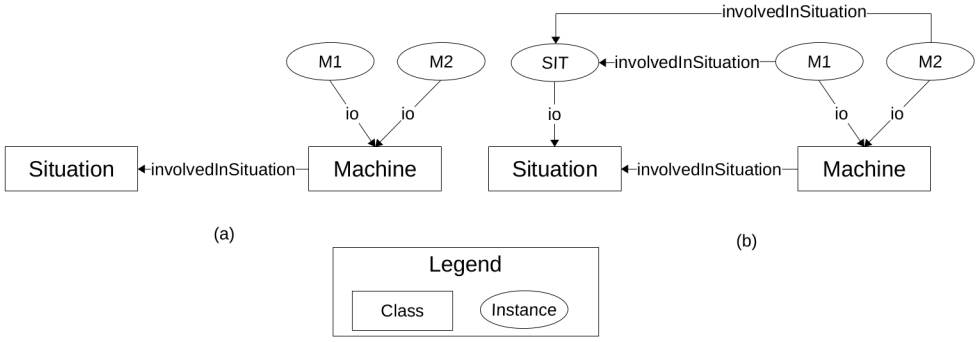

Figure 8.3 – Part of the ontological model before detecting `SIT` (a) and after detecting `SIT` (b).

```
OWLObjectProperty involvedIn = factory.getOWLObjectProperty(IRI.create(ns + "involvedInSituation"));
OWLObjectPropertyAssertionAxiom invM1 = factory.getOWLObjectPropertyAssertionAxiom(involvedIn,M1,SIT);
ontology.add(invM1);
OWLObjectPropertyAssertionAxiom invM2 = factory.getOWLObjectPropertyAssertionAxiom(involvedIn,M2,SIT);
ontology.add(invM2);
```

Listing 8.5 – Code to link `M1` and `M2` with `SIT`.

8.1.3 The Diagnosis component implementation

The *Cause Determination* is the only module of the Diagnosis component. The goal of this component is to identify the possible causes that generated an abnormal situation detected by the previous component.

Cause Determination module

In order to determine the possible cause(s) that lead to a certain abnormal situation, this module contains two sub-modules, one dedicated to real-time cause determination, and the other to be used when the cause(s) are not known in advance.

- The *Stream Reasoner* sub-module is implemented using C-SPARQL. This module explores the situations streams generated by the Monitoring component to identify the causes associated to those situations. This sub-module can be used to identify situation cause(s) in real-time, when situations can be generated at high frequency. Queries to identify the causes of the detected situations are registered and executed by the C-SPARQL engine over the situation streams.

- The *Reasoner* sub-module is used over the ontology to infer the causes associated to the detected situation (in this case, the situation is an instance in the ontology). This option can be used when it is not necessary to determine causes in real-time. For example, it is possible that some situations do not have identified causes in a real scenario or that certain facts that are necessary for the determination of causes are not explicitly expressed in the ontology, in which case the Cause Determination module notifies that the causes are unknown. In order to determine the possible causes of a detected situation, this sub-module offers classic reasoning mechanisms that provide other characteristics to those provided by the *Stream Reasoner* sub-module, such as the Open World Assumption.

  Identifying the causes of a detected situation can include the call of SWRLAPI methods for doing SWRL-based reasoning and the execution of SPARQL queries on the ontology. For example, a query could be: "Has the filter of a machine been changed?" If the change of the filter is explicitly represented in the ontological model then it is known that the change was made. In case the change is not explicitly represented in the ontological model, the query returns unknown. Therefore, we can not assure whether the change was made or not. Thus, the non-change of the filter may be the cause of the detected situation.

If the cause is identified later, it can be added to the ontology and linked to the situation for future use.

In both cases, the module provides the Decision Making component with the detected situation together with the possible causes in case they are known. The association between the situation and the causes is also added to the Ontology.

8.1.4 The Decision making component implementation

For this component, the algorithm 1 presented in section 7.2.2 was implemented in the Java language and is available here[8]. Furthermore, the operators $\lfloor . \rfloor$ and $\lceil . \rceil$ (defined in section 7.2.1) were implemented in order to implement the algorithm. The situation hierarchy is returned in the DOT[9] format file, which is a graph description language. For this, the jgrapht[10] package was used. Various programs can process DOT files, allowing the quick visualization of the situation hierarchy.

8.2 Proof of concept of the proposed framework

An illustrative case study is described in this section to highlight how the proposed framework can be used and the advantages it offers to support decisions that need to be made when an abnormal situation is detected in an industrial framework.

8.2.1 Case study description

The case study is based on a manufacturing production line (Figure 8.4), named `PL1`, composed of four machines: `M1`, `M2`, `M3` and `M4`. These machines and the production line are equipped with sensors. The sensors collect data on the properties described in Table 8.1. This scenario is formally represented using the ontological model introduced in Chapter 5 and it is shown in Figure 8.5.

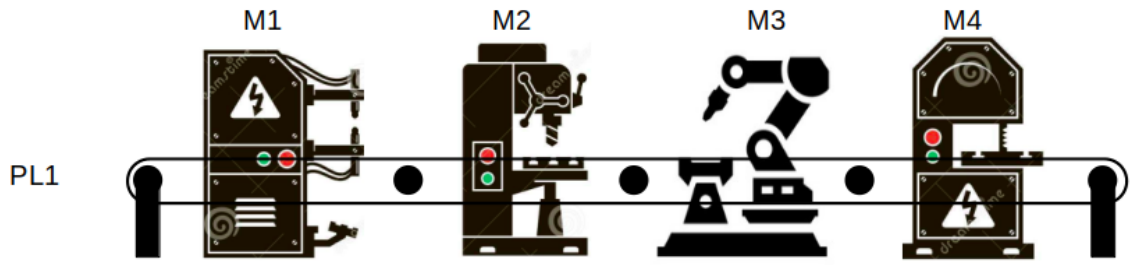

Figure 8.4 – Production line.

Several abnormal situations that leads to failures in this scenario are defined by experts. Each of them is expressed as a set of constraints. We focus on situations covering the following types of failures: machines malfunctions and global malfunctions of

[8] https://gitlab.insa-rouen.fr/fgiustozzi/STEaMINg-SituationHierarchy
[9] https://en.wikipedia.org/wiki/DOT_(graph_description_language)
[10] https://jgrapht.org/javadoc-1.1.0/org/jgrapht/package-summary.html

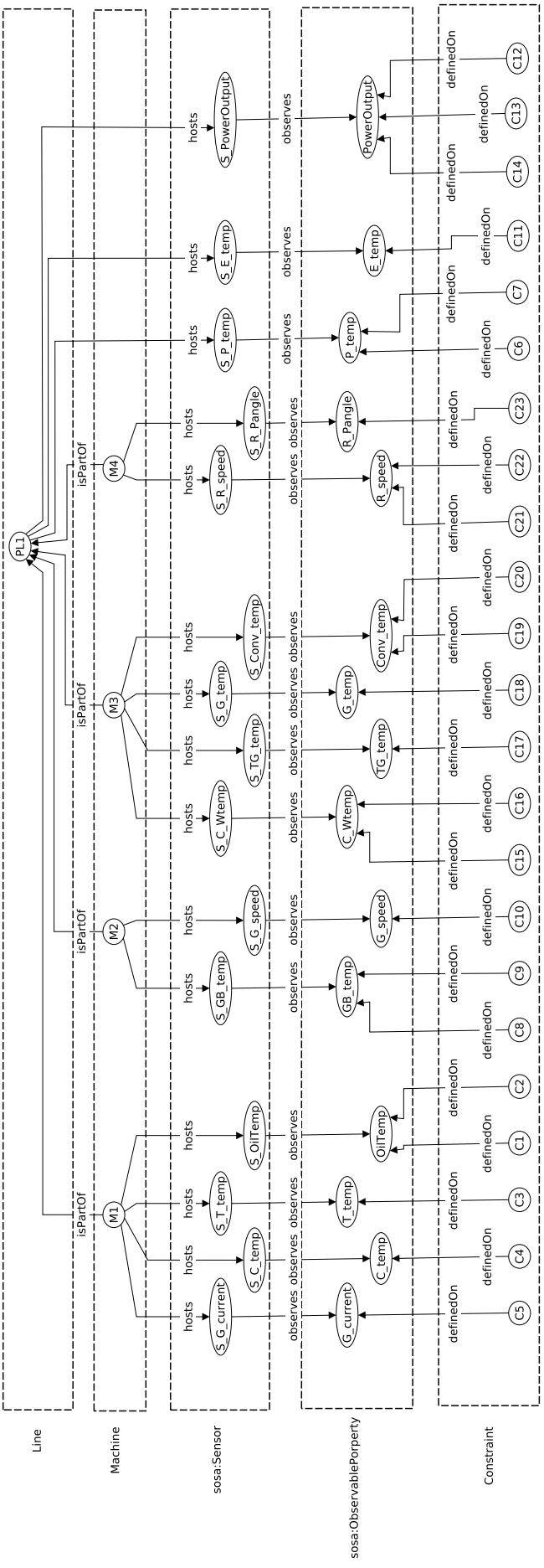

Figure 8.5 – Representation of the scenario using our semantic model.

Table 8.1 – Sensors attached to each Resource and their respective Observable properties.

Resource	Sensor	ObservableProperty
PL1	S_P_temp	P_temp
	S_E_temp	E_temp
	S_PowerOutput	PowerOutput
M1	S_G_current	G_current
	S_C_temp	C_temp
	S_T_temp	T_temp
	S_OilTemp	OilTemp
M2	S_GB_temp	GB_temp
	S_G_speed	G_speed
M3	S_C_Wtemp	C_Wtemp
	S_TG_temp	TG_temp
	S_G_temp	G_temp
	S_Conv_temp	Conv_temp
M4	S_R_speed	R_speed
	S_R_Pangle	R_Pangle

the production line including hydraulic oil leakage and cooling system filter obstructions. The abnormal situations are briefly describe below:

- The S1 situation is associated with an oil leak on the M1 machine. This situation happens when the constraints OilTemp > 40°C (c_1), T_temp > 45°C (c_3), C_temp > 40°C (c_4), G_current > 800 A (c_5) and P_temp < 35°C (c_6) are satisfied at least once in a 20 second period of time.

- The S2 situation is a more severe form of S1, also associated with an oil leak on the M1 machine. The S2 situation happens when the constraints OilTemp > 60°C (c_2), T_temp > 45°C (c_3), C_temp > 40°C (c_4), G_current > 800 A (c_5) and P_temp < 35°C (c_6) are satisfied at least once in a 20 second period of time. S2 is more severe than S1 because the oil temperature exceeds 60°C. More urgent actions are required to avoid the failure of the M1 machine.

- The S3 situation is associated with an increase in the oil temperature of the M2 machine. This situation happens when the constraints P_temp < 35°C (c_6), GB_temp > 40°C (c_8), G_speed < 500 rpm (c_{10}) and E_temp < 25°C (c_{11}) are satisfied at least once in a 25 second period of time.

- As with S1 and S2, the S4 situation is a more severe version of S3. The S4 situation happens when the constraints P_temp < 35°C (c_6), GB_temp > 60°C (c_9), G_speed < 500 rpm (c_{10}) and E_temp < 25°C (c_{11}) are satisfied at least once in a 25 second period of time. Since the GB temperature exceeds 60°C, immediate actions are required to avoid the failure of the M2 machine.

- The S5 situation is an even more severe situation. It happens when the constraints P_temp > 40°C (c_7), GB_temp > 60°C (c_9), G_speed < 500 rpm (c_{10}) and E_temp < 25°C

(c_{11}) are satisfied at least once in a 25 second period of time. S5 is more severe than S3 and S4 because the GB temperature exceeds 60°C and the platform temperature exceeds 40°C. Due to its degree of severity, drastic actions must be taken to avoid the failure of the M2 machine.

- The S6 situation is associated with a cooling system failure of the M3 machine. This situation happens when the constraints C_Wtemp > 60°C (c_{15}), TG_temp < 35°C (c_{17}) and G_temp > 45°C (c_{18}) are satisfied at least once in a 15 second period of time.

- As the S6 situation, the S7 situation is associated with a cooling system failure of the M3 machine. The S7 situation happens when the constraints C_Wtemp > 80°C (c_{16}), TG_temp < 35°C (c_{17}) and G_temp > 45°C (c_{18}) are satisfied at least once in a 15 second period of time. S7 is more severe than S6 because the converter water temperature exceeds 80°C. In the same way, that the most severe situations above the actions to take must be prompt to avoid in this case the failure of the cooling system of the M3 machine.

- The S8 situation is associated with a malfunction of the M3 machine. This situation happens when the constraints P_temp < 35°C (c_6), TG_temp < 35°C (c_{17}) and Conv_temp > 60°C (c_{19}) are satisfied at least once in a 20 second period of time.

- The S9 situation is more severe than S8. The S9 situation happens when the constraints P_temp < 35°C (c_6), TG_temp < 35°C (c_{17}) and Conv_temp > 80°C (c_{20}) are satisfied in at least once in a 20 second period of time. In this case, as the converter temperature exceeds 80°C, more urgent actions are required to avoid the failure of the M3 machine.

- The S10 situation is associated with a malfunction of the M3 machine as the two previous situations. The S10 situation happens when the constraints C_Wtemp > 80°C (c_{16}), TG_temp < 35°C (c_{17}), G_temp < 45°C (c_{18}) and Conv_temp > 60°C (c_{19}) are satisfied at least once in a 20 second period of time. S10 is more severe than S8 and S9 because the GB temperature exceeds 60°C and the platform temperature exceeds 40°C. In order to avoid a failure of M3, immediate actions must be taken.

- The S11 situation is associated with a malfunction of the M4 machine. This situation happens when the constraints PowerOutput > 2000 KW (c_{12}), R_speed < 200 rpm (c_{21}) and R_Pangle < 5° (c_{23}) are satisfied at least once in a 30 second period of time.

- The S12 situation is associated with the same malfunction as the S11 situation. The S12 situation happens when the constraints PowerOutput > 2000 KW (c_{12}), R_speed < 100 rpm (c_{22}) and R_Pangle < 5° (c_{23}) are satisfied in least once in a 30 second period of time. S12 is more severe than S11 because the rotor speed of the M4 machine is lower than 100 rpm. It requires immediate action to prevent the failure of the M4 machine.

- The S13 situation is associated with a global malfunction of the PL1 production line. This situation happens when the constraints PowerOutput < 500 KW (c_{13}), R_speed

< 200 rpm (c_{21}) and R_Pangle $< 5°$ (c_{23}) are satisfied at least once in a 35 second period of time.

- As the S13 situation, the S14 situation is also associated with a malfunction of the PL1 production line. The S14 situation happens when the constraints PowerOutput < 200 KW (c_{14}), R_speed < 200 rpm (c_{21}) and R_Pangle $< 5°$ (c_{23}) are satisfied at least once in a 35 second period of time. S14 is more severe version of S13 because the power output of the PL1 production line is lower than 200 KW.

8.2.2 Abnormal situation detection and cause determination

As mentioned in the section 8.1.2, the situations defined above are detected through C-SPARQL queries. In this section, we describe a particular query and see the effect it has on the ontological model when the situation is detected.

We must emphasize that for this case study all the data streams of the properties (ObservableProperties) defined in table 8.1 are generated using the *RDFStream* class provided by C-SPARQL to generate RDF streams. Queries can also be executed on data streams that are generated and published by other systems or users, however it is necessary to know the structure of the RDF stream to be able to execute queries on them.

The C-SPARQL query presented in Listing 8.6 has the purpose of detecting the situation S6, defined previously. The query name is registered on line 1 and prefixes used in the query are declared on lines 2 and 3. The query is executed on RDF streams that correspond to the properties C_Wtemp, TG_temp and G_temp in the time frame of 15 seconds, sliding the window by 5 seconds (line 5-7). The chosen time frame is arbitrary and can be changed as desired. It produces pairs of values (line 4): the machine name (?m) and the production line of which it is a part of (?pl). In order to obtain the production line to which the machine belongs, we indicate in the query that the C-SPARQL engine must use our ontological model as background knowledge (line 8). Line 10 enables to obtain the production line to which the machine belongs, as shown in Figure 8.6. To get the observation's values, ?o1, ?o2 and ?o3 individuals are respectively bound with the data values ?v1, ?v2 and ?v3 through the appropriate properties (sosa:madeObservation and sosa:hasSimpleResult) (line 11-19). Finally, the list of output pairs are filtered out to include only the ones where the observation's values satisfy the restrictions in the FILTER clause (line 20-23).

```
1  REGISTER QUERY S6-detection AS
2  PREFIX : <http://semanticweb.org/STEaMINg/ContextOntology-COInd4#>
3  PREFIX sosa: <http://www.w3.org/ns/sosa/>
4  SELECT ?m ?pl
5  FROM STREAM <Stream_C_Wtemp> [RANGE 15s STEPS 5s]
6  FROM STREAM <Stream_TG_temp> [RANGE 15s STEPS 5s]
7  FROM STREAM <Stream_G_temp> [RANGE 15s STEPS 5s]
8  FROM <http://semanticweb.org/STEaMINg/ContextOntology-COInd4#>
9  WHERE {
10    ?m          :isPartOf            ?pl .
11    ?m          sosa:hosts           sosa:S_C_Wtemp .
12    :S_C_Wtemp  sosa:madeObservation ?o1 .
13    ?o1         sosa:hasSimpleResult ?v1 .
14    ?m          sosa:hosts           sosa:S_TG_temp .
```

```
15    :S_TG_temp  sosa:madeObservation ?o2 .
16    ?o2         sosa:hasSimpleResult ?v2 .
17    ?m          sosa:hosts           sosa:S_G_temp .
18    :S_G_temp   sosa:madeObservation ?o3 .
19    ?o3         sosa:hasSimpleResult ?v3 .
20  FILTER (
21    ?v1 > 60.0 &&
22    ?v2 < 35.0  &&
23    ?v3 > 45.0 ) .
24  }
```

Listing 8.6 – C-SPARQL query to detect the S6 situation.

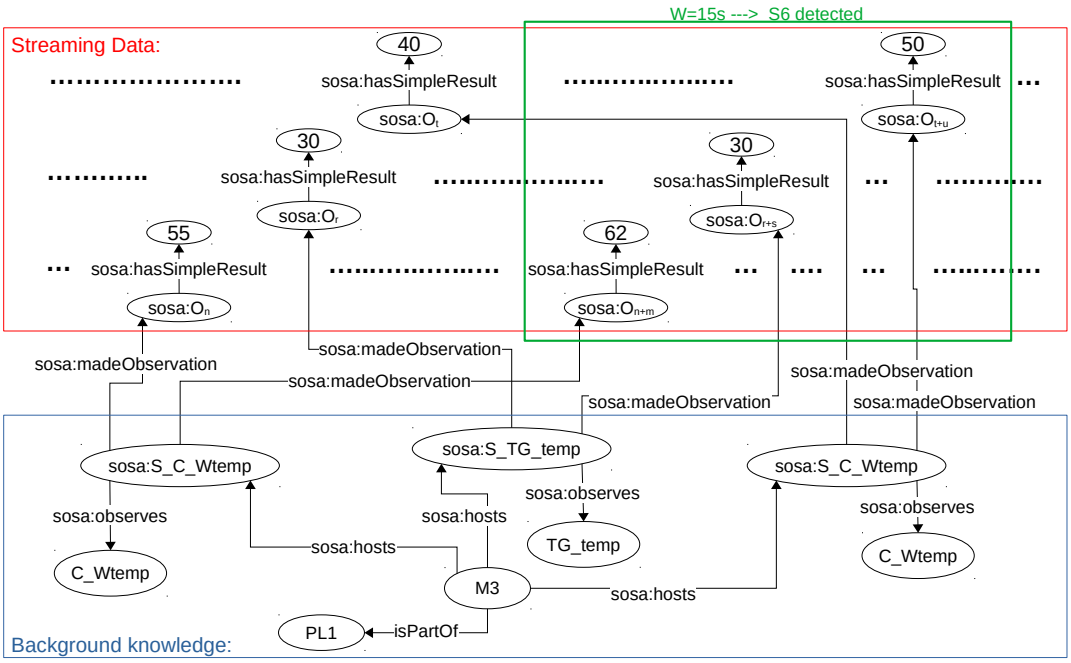

Figure 8.6 – Background knowledge and streaming data for S6 situation detection.

Once the situation is detected, it is added as an instance to the ontology as well as the relationships representing the resources concerned by this situation. Figure 8.7 shows a part (relevant to this case) of the ontological model before and after the detection of the S6 situation.

The next step is to determine the possible cause(s) of the S6 situation. The S6 situation is associated with a failure of the cooling system of the M3 machine. By expert knowledge we know that this situation can be caused by polluted filters in the cooling system. Therefore, the following rule is used to add this fact to the ontology. Figure 8.8 shows the added cause associated with the detected situation.

$$\texttt{Situation-S6(?sit)} \Rightarrow \texttt{hasCause(?sit,:PollutedFilters)}$$

It is worth mentioning that in case the cause is not known beforehand, it can be added to the ontological model. Thus, the next time the situation is detected the cause is automatically determined.

Once the situation and its possible cause(s) have been detected, the next step is to make a decision to avoid the failure associated with the situation. To determine what

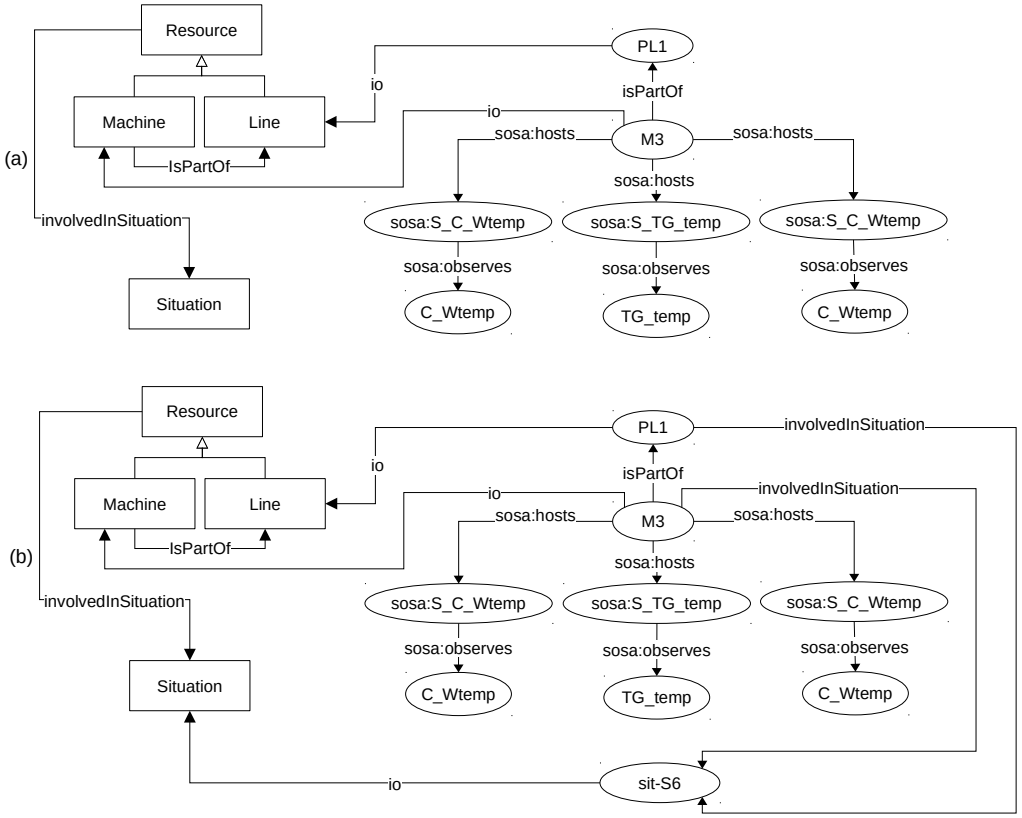

Figure 8.7 – Part of the ontological model: (a) before detecting the S6 situation; and (b) after detecting it.

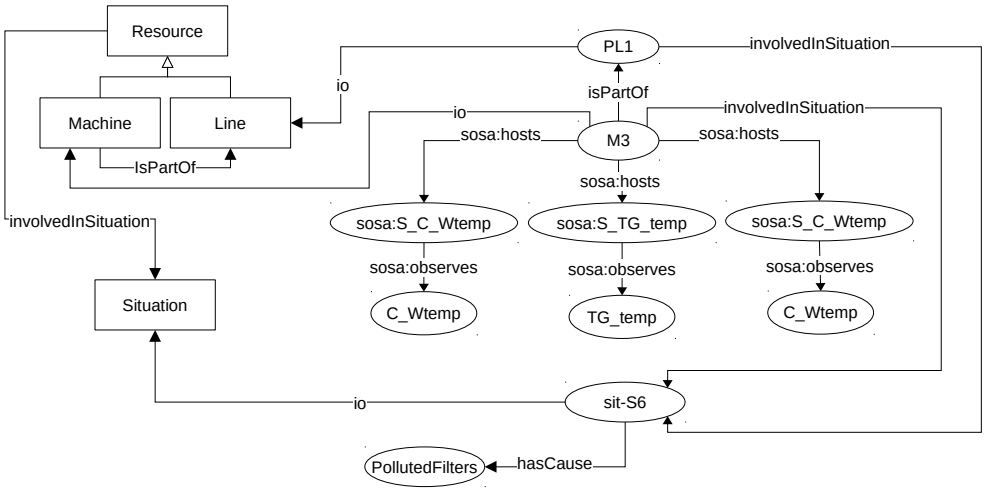

Figure 8.8 – Part of the ontological model after cause determination.

action to make, we use the approach proposed in Chapter 7 to build the hierarchy of situations, based on the constraints on which the situations rely. The figure 8.9 shows the hierarchy of situations defined in the case study.

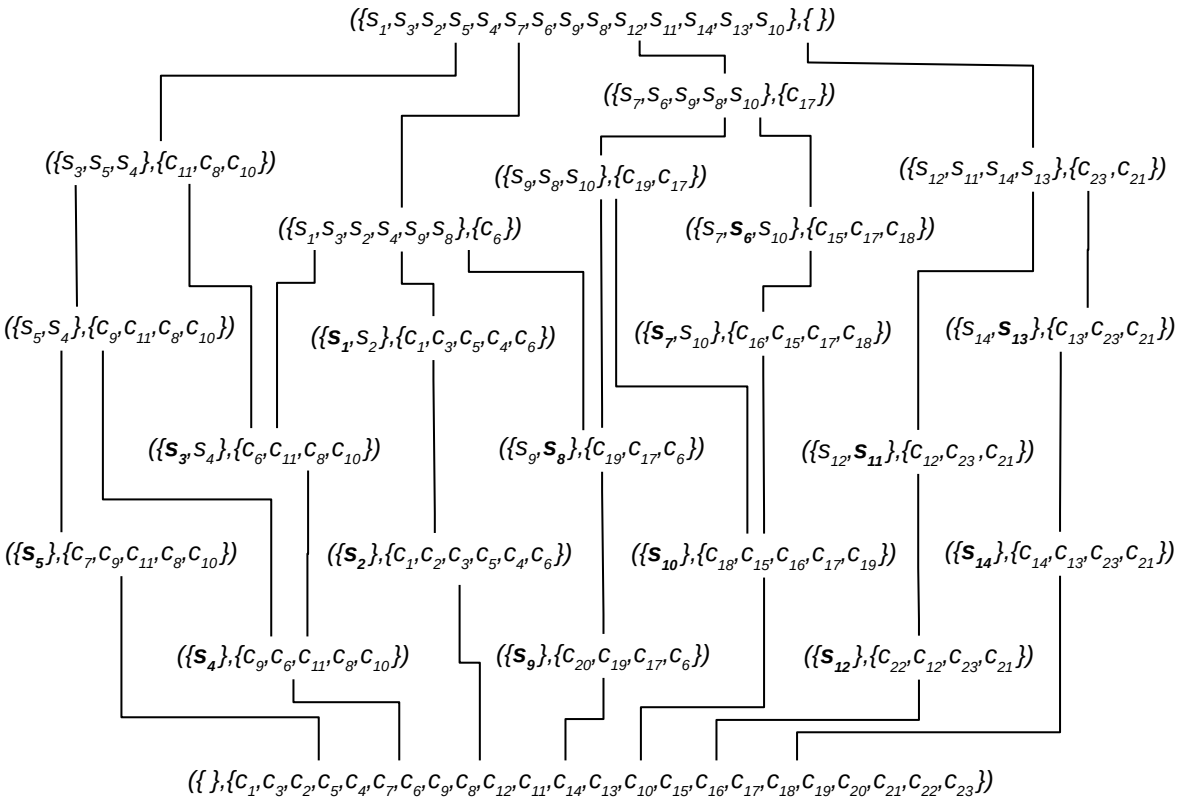

Figure 8.9 – Hierarchy of the situations defined in the illustrative case study.

Having the situation detected and its possible cause, the Decision making component can use both to build a strategy to correct the abnormal behavior. For example, one action could be to replace the polluted filters which would lead to the constraints associated with the S6 situation no longer being met, *i.e.* correction of the abnormal behavior. In case this action cannot be carried out, the system or the operators can decide that the production line continues with the execution of its processes. This could lead to the S7 situation, satisfying the C_Wtemp > 80°C (c_{16}) constraint, and thus requiring more urgent actions since the M3 machine would be closer to a failure. By observing the hierarchy, we can see that the behavior of the M3 machine could get even worse, since the S10 situation could also be reached if the Conv_temp > 60°C (c_{19}) constraint is also satisfied. Unlike situation S6 and S7 which are associated with a failure of the M3 machine's cooling system, situation S10 indicates that the M3 machine is having a global malfunction. So the strategy to follow could be more drastic as for example stopping the execution of the processes of the M3 machine, halting also the production of the whole production line (PL1).

Our approach provides the Decision making component with high level information such as situations and their possible causes from raw data. In addition, the hierarchy of situations allows visualizing the possible intermediate situations that the production system can reach. In this way, it is sought that the Decision making component takes appropriate actions to maintain the reliability and availability of the production line.

8.3 Conclusion

This chapter presents the implementation of the proposed framework. The technologies and tools used for the development are introduced as well as the implementation of the core functionalities of it.

An illustrative case study from the industrial domain is presented to demonstrate the application of the framework. The goal is to show the interaction between the different components and how they modify the semantic model, making it evolve for the representation of what happens in the real factory. The results obtained by our proof of concept using simulated data are encouraging. Unfortunately, it has not been possible to validate the whole approach on real industrial data, even off-line.

Ideally, it would also be interesting to test our proposal in a real production line. This would allow to verify if the decisions made based on the use of our proposal would improve its efficiency and reliability.

Conclusion & Future Work

In the following, we present a summary of these thesis works with their corresponding contributions. Afterwards, some perspectives and possible lines of future work are also presented.

Conclusion

As already evoked all along this manuscript, the management of heterogeneous industrial data is a challenging task in the framework of condition monitoring. Additionally, as the structure and behavior of production systems get more and more complex, the volume of data grows significantly. This is why manufacturing companies search for solutions to handle this heterogeneous industrial data efficiently and perform monitoring and diagnosis tasks in a smart way.

In this framework, there is a need of well-defined models for managing heterogeneous data and exploiting it together with expert knowledge. To develop such a model, domain knowledge of manufacturing has to be structured in a machine-interpretable way and usable by the monitoring system. Furthermore, as the manufacturing domain becomes more knowledge-intensive, a uniform knowledge representation of physical resources and reasoning capabilities is needed to automate the decision-making processes. These decision-making processes include abnormal situation detection and cause determination, maintenance scheduling, and process adaptation. To achieve these objectives, semantic technologies have shown promising results when formalizing knowledge about condition monitoring tasks in several industrial applications.

Existing ontological models to represent the manufacturing domain do not allow the representation of concepts and relationships that evolve over time, such as the different situations under which a manufacturing resource can perform its tasks. These existing ontological models are suitable for information that does not change or changes very little in time. However, the manufacturing domain is highly-dynamic and a production system should be able to adapt itself to changing situations. This requires the ontological model to cope with the dynamic change of knowledge. Different sets of actions can be selected for correcting the abnormal behavior according to the specific physical context of the operating machine. This ensures that production continues without the need to stop it if the abnormal situation is not severe.

To resolve all of the above, taking into account the requirements indicated in the introduction of this manuscript, a novel semantic framework is proposed to address the evo-

lution of semantic models in Industry 4.0. The proposed framework allows to automate and facilitate condition monitoring and diagnosis, and support decision-making in the manufacturing domain. To this end, firstly we propose an ontological model for the manufacturing domain that represents the resources and processes that are part of a factory, with special emphasis on modeling the context of these resources and processes. Relevant situations that combine sensor observations with domain knowledge are also represented in the model. The proposed framework enriches data collected from sensors with contextual information using the ontological model and uses stream reasoning to allow abnormal situation detection in real-time. Furthermore, it also uses classical reasoning approaches to compensate for possible non-detection of the causes by the stream reasoning method. Through the detection of these situations and their causes, appropriate decisions can be made to avoid the interruption of the monitored process. In addition, to support decision-making the proposed framework provides a hierarchy of situations that represents a road-map of the situations that can be reached from a given one, desirable or undesirable. This allows the identification of the actions to take to correct the abnormal behavior, by avoiding or minimizing in this way the interruption of the manufacturing processes.

The contributions of this thesis are summarized below.

In Chapter 4, an overview of the proposed framework to address the evolution of semantic models in Industry 4.0 is introduced. The components of the proposed framework are: (1) the Monitoring component; (2) the Diagnosis component; and (3) the Decision making component. Each of the components operates as an expert does. Each one uses the semantic model to perform its functions and makes it evolve by introducing the corresponding changes. In this way, the tasks of each component are performed considering the updated semantic model which is a virtual representation of what happens in the real factory.

In Chapter 5, an ontology-based approach for context modeling in the industrial domain is proposed. The ontological model satisfies the Industry 4.0 requirements with special emphasis on temporal and spatial relations among resources and processes in order to represent particular situations of interest in an industrial scenario. Furthermore, our ontological model provides a way to capture the dynamic changes and the evolution of concepts in time. The proposed ontology is generic and extensible and its modular architecture allows the description of the capabilities of manufacturing resources at different levels of granularity.

In Chapter 6, the components of our proposed framework are detailed (the Monitoring, the Diagnosis, the Abnormal Situation Refinement and the Decision Making components). The use of stream reasoning allows the integration of data from different data sources, with different underlying meanings, different temporal resolutions as well as the processing of these data in real-time. Data collected from sensors are enriched with contextual information to allow real-time situation detection. Furthermore, our proposal also uses classic reasoning approaches to complete the determination of causes that could not be obtained by stream reasoning.

In Chapter 7, an approach that uses the lattice theory to represent a hierarchy of situ-

ations that can lead to potential failures is presented. The hierarchy of situations is built automatically considering the constraints they rely on. The hierarchy allows the representation of situations that occur partially, *i.e.* when only some of their constraints are satisfied. The proposed approach can be applied to other application domains, where real time monitoring is needed.

Finally, in Chapter 8, the technologies and tools used for the development of the proposed framework are introduced as well as the implementation of the core functionalities of it. An illustrative case study from the industrial domain is presented to demonstrate the application of the proposed framework. Unfortunately, it has not been possible to validate the whole approach on real industrial data. However, the results obtained by our proof of concept using simulated data are encouraging.

Future Work

The contributions presented in this thesis enable to identify several research perspectives.

Future works can be oriented to the enrichment of the proposed semantic model. New types of relations can be explored aside from the spatial and temporal ones addressed in this thesis, in order to allow the representation of richer contextual information. For example, representing the relationship that two production lines execute the same process could be used to divert the production from one production line to the other if the first one is behaving abnormally. Another possibility can be the representation of relations between production lines and work orders, so that depending on whether the production line is close to finishing a work order and whether the deadline is coming up, the decision to continue the production until the order is finished or to stop the production line can be made.

Since the manufacturing domain is highly-dynamic, how to process real-time and heterogeneous data streams is a crucial concern. In order to test scalability and complexity issues to detect situations in real-time, a broader case study with more complex situations, involving more properties observations that are temporal and spatially related, has to be explored. Different implementations of stream reasoning engines should be evaluated in terms of efficiency and scalability, considering also the size of the semantic model.

Another interesting direction to investigate is the identification of causality patterns that could enable the definition of generic queries for certain types of anomalies. These generic queries could be reused in different cases, and perform complex operations among the sensed values.

Regarding the refinement and identification of new situations, it can be interesting to apply machine learning methods to exploit the data stored in the semantic model. As seen in this thesis, the queries are executed during certain time windows every certain intervals of time, therefore the application of machine learning methods can help to determine the duration of the time windows in which a query needs to be executed. In addition, it would also be interesting to dynamically adapt the duration of the time windows.

There are also perspectives associated to the construction of the hierarchy of situa-

tions from real industrial data. Firstly, this construction also raises scalability and complexity issues to build the lattice. Therefore, tests need to be performed with different variations of Algorithm 1 to evaluate their relative efficiency. In addition, another possible future work is to investigate how to add new situations or the refined ones without having to rebuild the hierarchy from scratch.

Finally, a more exhaustive and detailed study can be made on the constraints implications that represent dependencies among constraints for the identification of new situations or the refinement of existing ones. In this way, different hierarchies of situations can be obtained according to the chosen $\mathscr{T}$ relation, that can be used alone or combined with others, allowing different strategies for decision making.

Bibliography

[Abb12] Sunitha Abburu. A survey on ontology reasoners and comparison. *International Journal of Computer Applications*, 57:33–39, 2012. 8, 53

[ABW04] Arvind Arasu, Shivnath Babu, and Jennifer Widom. CQL: A Language for Continuous Queries over Streams and Relations. In Georg Lausen and Dan Suciu, editors, *Database Programming Languages*, pages 1–19, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg. 55, 56, 140

[AD06] Farhad Ameri and Debasish Dutta. An upper ontology for manufacturing service description. In *Proceedings of the ASME Design Engineering Technical Conference*, 2006. 73

[AF01] Alessandro Artale and Enrico Franconi. A survey of temporal extensions of description logics. *Annals of Mathematics and Artificial Intelligence*, 30(1–4):171–210, March 2001. 78, 79

[AH04] Grigoris Antoniou and Frank Harmelen. *A Semantic Web primer*. 01 2004. 50

[AK12] Rosmaini Ahmad and Shahrul Kamaruddin. An overview of time-based and condition-based maintenance in industrial application. *Computers & industrial engineering*, 63(1):135–149, 2012. 11, 68

[All94] Bradley P Allen. Case-based reasoning: Business applications. *Communications of the ACM*, 37(3):40–43, 1994. 57, 101

[AMX+16] Kosmas Alexopoulos, Sotiris Makris, Vangelis Xanthakis, Konstantinos Sipsas, and George Chryssolouris. A concept for context-aware computing in manufacturing: the white goods case. *International Journal of Computer Integrated Manufacturing*, 29(8):839–849, 2016. 77

[AP94] Agnar Aamodt and Enric Plaza. Case-based reasoning: Foundational issues, methodological variations, and system approaches. *AI communications*, 7(1):39–59, 1994. xiii, 58

[APP+19] Sara Antomarioni, Ornella Pisacane, Domenico Potena, Maurizio Bevilacqua, Filippo Emanuele Ciarapica, and Claudia Diamantini. A predictive association rule-based maintenance policy to minimize the probability of breakages: application to an oil refinery. *The International Journal of Advanced Manufacturing Technology*, 105(9):3661–3675, 2019. 71

[AR88] Kevin D Ashley and Edwina L Rissland. A case-based approach to modeling legal expertise. *IEEE Intelligent Systems*, (3):70–77, 1988. 57

[ARFS12] Darko Anicic, Sebastian Rudolph, Paul Fodor, and Nenad Stojanovic. Stream reasoning and complex event processing in etalis. *Semantic web*, 3(4):397–407, 2012. 8, 52

[BB18] Toufik Berredjem and Mohamed Benidir. Bearing faults diagnosis using fuzzy expert system relying on an improved range overlaps and similarity method. *Expert Systems with Applications*, 108:134–142, 2018. 71

[BBC+10] Davide Francesco Barbieri, Daniele Braga, Stefano Ceri, Emanuele Della Valle, and Michael Grossniklaus. C-SPARQL: a continuous query language for RDF data streams. *International Journal of Semantic Computing*, 4(01):3–25, 2010. 54, 55

[BBRC09] Yazid Benazzouz, Philippe Beaune, Fano Ramparany, and Laure Chotard. Context data-driven approach for ubiquitous computing applications. In *2009 Fourth International Conference on Digital Information Management*, pages 1–6, 2009. 61

[BCM+03] Franz Baader, Diego Calvanese, Deborah L. McGuinness, Daniele Nardi, and Peter F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, USA, 2003. 47, 50, 96

[BCR09] O. Brdiczka, J. L. Crowley, and P. Reignier. Learning situation models in a smart home. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 39(1):56–63, 2009. 61

[BDG+07] Lars Brenna, Alan Demers, Johannes Gehrke, Mingsheng Hong, Joel Ossher, Biswanath Panda, Mirek Riedewald, Mohit Thatte, and Walker White. Cayuga: A high-performance event processing engine. 01 2007. 124

[BGJ08] Andre Bolles, Marco Grawunder, and Jonas Jacobi. Streaming sparql-extending sparql to process data streams. In *European Semantic Web Conference*, pages 448–462. Springer, 2008. 55

[BHS05] Franz Baader, Ian Horrocks, and Ulrike Sattler. *Description Logics as Ontology Languages for the Semantic Web*, pages 228–248. Springer Berlin Heidelberg, Berlin, Heidelberg, 2005. 48, 111

[BKE03] Joseph Bauer, Ralf-Detlef Kutsche, and Rudiger Ehrmanntraut. Identification and modeling of contexts for different information scenarios in air traffic. *Technische Universität Berlin, Diplomarbeit*, 2003. 62

[BKR+12] Raphael Barbau, Sylvere Krima, Sudarsan Rachuri, Anantha Narayanan, Xenia Fiorentini, Sebti Foufou, and Ram D. Sriram. OntoSTEP: Enriching product model data using ontologies. *CAD Computer Aided Design*, 2012. 72

[BL07] Stefano Borgo and Paulo Leitão. Foundations for a Core Ontology of Manufacturing, 2007. 73

[BLDdO18] Sérgio Baltazar, Chuan Li, Helder Daniel, and José Valente de Oliveira. A review on neurocomputing based wind turbines fault diagnosis and prognosis. In *2018 Prognostics and System Health Management Conference (PHM-Chongqing)*, pages 437–443. IEEE, 2018. 70

[BLF00] T. Berners-Lee and M. Fischetti. *Weaving the Web: The Past, Present and Future of the World Wide Web by Its Inventor*. Texere, 2000. xiii, 48

[BM18] Rabah Benkercha and Samir Moulahoum. Fault detection and diagnosis based on c4. 5 decision tree algorithm for grid connected pv system. *Solar Energy*, 173:610–634, 2018. 70

[BP11] Sotiris Batsakis and Euripides G.M. Petrakis. SOWL: A framework for handling spatio-temporal information in OWL 2.0. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2011. 79

[BPSM+00] Tim Bray, Jean Paoli, C Michael Sperberg-McQueen, Eve Maler, Franois Yergeau, et al. Extensible markup language (xml) 1.0, 2000. 49

[BS09] Christian Bizer and Andreas Schultz. The berlin sparql benchmark. *International Journal on Semantic Web and Information Systems (IJSWIS)*, 5(2):1–24, 2009. 56

[CBI] CBINSIGHTS: https://www.cbinsights.com/research/future-factory-manufacturing-tech-trends/. (accessed March 2, 2019). 114

[CBL+18] Ana Cachada, Jose Barbosa, Paulo Leitño, Carla AS Gcraldcs, Leonel Deusdado, Jacinta Costa, Carlos Teixeira, João Teixeira, António HJ Moreira, Pedro Miguel Moreira, et al. Maintenance 4.0: Intelligent and predictive maintenance system architecture. In *2018 IEEE 23rd International Conference on Emerging Technologies and Factory Automation (ETFA)*, volume 1, pages 139–146. IEEE, 2018. 12, 69

[CBS+17] Federico Civerchia, Stefano Bocchino, Claudio Salvadori, Enrico Rossi, Luca Maggiani, and Matteo Petracca. Industrial internet of things monitoring solution for advanced predictive maintenance applications. *Journal of Industrial Information Integration*, 7:4 – 12, 2017. Enterprise modelling and system integration for smart manufacturing. 66

[CCR93] Zhan Cui, Anthony G Cohn, and David A Randell. Qualitative and topological relationships in spatial databases. In *International Symposium on Spatial Databases*, pages 296–315. Springer, 1993. 100

[CFJ03] Harry Chen, Tim Finin, and Anupam Joshi. An ontology for context-aware pervasive computing environments. *The Knowledge Engineering Review*, 18(3):197–207, 2003. xiii, 75

[CL05] S Wesley Changchien and Ming-Chin Lin. Design and implementation of a case-based reasoning system for marketing plans. *Expert systems with applications*, 28(1):43–53, 2005. 57

[CL17] S. J. D. Cox and C. Little. Time ontology in OWL. W3C Recommendation. 2017. 78

[CM10] Gianpaolo Cugola and Alessandro Margara. Tesla: A formally defined event specification language. In *Proceedings of the Fourth ACM International Conference on Distributed Event-Based Systems*, DEBS '10, page 50–61, New York, NY, USA, 2010. Association for Computing Machinery. 124

[CM12] Gianpaolo Cugola and Alessandro Margara. Processing flows of information: From data stream to complex event processing. *ACM Comput. Surv.*, 44(3), June 2012. 124

[CNW12] L. Chen, C. D. Nugent, and H. Wang. A Knowledge-Driven Approach to Activity Recognition in Smart Homes. *IEEE Transactions on Knowledge and Data Engineering*, 24(6):961–974, 2012. 61

[CPFJ04] Harry Chen, Filip Perich, Tim Finin, and Anupam Joshi. SOUPA: Standard ontology for ubiquitous and pervasive applications. In *Proceedings of MOBIQUITOUS 2004 - 1st Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services*, pages 258–267, 2004. xiii, 75, 76

[CWHZ18] Xiao-long Chen, Pei-hong Wang, Yong-sheng Hao, and Ming Zhao. Evidential knn-based condition monitoring and early warning method with applications in power plant. *Neurocomputing*, 315:18–32, 2018. 70

[CZX+16] Haibo Cheng, Peng Zeng, Lingling Xue, Zhao Shi, Peng Wang, and Haibin Yu. Manufacturing ontology development based on industry 4.0 demonstration production line. pages 42–47, 09 2016. 15, 73, 82

[DA99] Anind K. Dey and Gregory D. Abowd. Towards a better understanding of context and context-awareness. In *HUC '99: Proceedings of the 1st international symposium on Handheld and Ubiquitous Computing*, pages 304–307. Springer-Verlag, 1999. 9, 42

[DA00] A.K. Dey and G.D. Abowd. Towards a better understanding of context and context-awareness. *Proceedings of the CHI 2000 Workshop on The What, Who, Where, When and How of Context Awareness*, 4:1–6, 2000. 2, 9, 28, 60, 75, 94

[DAS01] Anind K Dey, Gregory D Abowd, and Daniel Salber. A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications. *Human–Computer Interaction*, 16(2-4):97–166, 2001. 9, 60

[Dav02] H. A. Davey, B. A. Priestley. *Introduction to Lattices and Order*. Cambridge University Press, 2 edition, 2002. 124

[DB03] Jos De Bruijn. Using ontologies. enabling knowledge sharing and reuse on the semantic web. *Digital Enterprise Research Institute (DERI) Technical Report*, 2003. 62

[DD79] Phyllis M Deane and Phyllis M Deane. *The first industrial revolution*. Cambridge University Press, 1979. 1, 27

[DGP19] Uday Kumar Diego Galar Pascual, Pasquale Daponte. *The Industry 4.0 Architecture and Cyber-Physical Systems*. CRC Press, 2019. 11, 66

[DMT+07] Charalampos Doukas, Ilias Maglogiannis, Philippos Tragas, Dimitris Liapis, and Gregory Yovanof. Patient fall detection using support vector machines. In Christos Boukis, Aristodemos Pnevmatikakis, and Lazaros Polymenakos, editors, *Artificial Intelligence and Innovations 2007: from Theory to Applications*, pages 147–156, Boston, MA, 2007. Springer US. 61

[dRFID+19] Mike de Roode, Alba Fernández-Izquierdo, Laura Daniele, María Poveda-Villalón, and Raúl García-Castro. Saref4inma: a saref extension for the industry and manufacturing domain. 2019. 74

[DVCB+08] Emanuele Della Valle, Stefano Ceri, Davide Francesco Barbieri, Daniele Braga, and Alessandro Campi. A first step towards stream reasoning. In *Future Internet Symposium*, pages 72–81. Springer, 2008. 54

[FMK10] Flavius Frasincar, Viorel Milea, and Uzay Kaymak. TOWL: Integrating time in OWL. In *Semantic Web Information Management: A Model-Based Perspective*. 2010. 79

[GC18] Muhammet Gul and Erkan Celik. Fuzzy rule-based fine–kinney risk assessment approach for rail transportation systems. *Human and Ecological Risk Assessment: An International Journal*, 24(7):1786–1812, 2018. 71

[Ger13] Ilya Gertsbakh. *Reliability theory: with applications to preventive maintenance*. Springer, 2013. 11, 68

[GF12] M. Garetti and L. Fumagalli. P-PSO ontology for manufacturing systems. In *IFAC Proceedings Volumes (IFAC-PapersOnline)*, 2012. 73

[GGKL07] Sven Groppe, Jinghua Groppe, Dirk Kukulenz, and Volker Linnemann. A sparql engine for streaming rdf data. In *2007 Third International IEEE Conference on Signal-Image Technologies and Internet-Based System*, pages 167–174. IEEE, 2007. 55

[GGM+02] Aldo Gangemi, Nicola Guarino, Claudio Masolo, Alessandro Oltramari, and Luc Schneider. Sweetening ontologies with dolce. In *International Conference on Knowledge Engineering and Knowledge Management*, pages 166–181. Springer, 2002. 47

[GHV05] Claudio Gutierrez, Carlos Hurtado, and Alejandro Vaisman. Temporal RDF. In *Lecture Notes in Computer Science*, 2005. 78, 80

[GMD12] Fausto Giunchiglia, Vincenzo Maltese, and Biswanath Dutta. Domains and context: first steps towards managing diversity in knowledge. *Journal of web semantics*, 12:53–63, 2012. 9, 60

[GP07] D. Guo and Z.K. Peng. Vibration analysis of a cracked rotor using hilbert–huang transform. *Mechanical Systems and Signal Processing*, 21(8):3030 – 3041, 2007. 70

[GPFLCGP04] Asunción Gómez-Pérez, Mariano Fernández-López, Oscar Corcho, and a Gomez-Perez. *Ontological Engeenering*. 2004. 44, 96

[Gra11] George Gratzer. *Lattice Theory: Foundation*. 01 2011. 124

[Gra20] Bernard Grabot. Rule mining in maintenance: Analysing large knowledge bases. *Computers & Industrial Engineering*, 139:105501, 2020. 71

[Gru93] Thomas R. Gruber. A translation approach to portable ontology specifications. *Knowledge Acquisition*, 5(2):199–220, 1993. 7, 28, 40, 43, 44

[Grü09] Michael Grüninger. Using the PSL Ontology. In *Handbook on Ontologies*. 2009. 72

[Gua97] Nicola Guarino. Semantic matching: Formal ontological distinctions for information organization, extraction, and integration. In *International Summer School on Information Extraction*, pages 139–170. Springer, 1997. 46

[GVCF07] Laurent-Walter Goix, Massimo Valla, Laura Cerami, and Paolo Falcarin. Situation inference for mobile users: a rule based approach. In *2007 International Conference on Mobile Data Management*, pages 299–303. IEEE, 2007. 62

[Has11] H. M. Hashemian. State-of-the-Art Predictive Maintenance Techniques. *IEEE Transactions on Instrumentation and Measurement*, 60(1):226–236, Jan 2011. 1, 28, 114

[HB11] Matthew Horridge and Sean Bechhofer. The owl api: A java api for owl ontologies. *Semant. web*, 2(1):11–21, January 2011. 96

[HBS02] Albert Held, Sven Buchholz, and Alexander Schill. Modeling of context information for pervasive computing applications. *Proceedings of SCI*, pages 167–180, 2002. 61

[HH92] Daniel Hennessy and David Hinkle. Applying case-based reasoning to autoclave loading. *IEEE expert*, 7(5):21–26, 1992. 57

[HIR02] Karen Henricksen, Jadwiga Indulska, and Andry Rakotonirainy. Modeling context information in pervasive computing systems. In *International Conference on Pervasive Computing*, pages 167–180. Springer, 2002. 62

[HIR03] Karen Henricksen, Jadwiga Indulska, and Andry Rakotonirainy. Generating context management infrastructure from high-level context models. In *In 4th International Conference on Mobile Data Management (MDM)-Industrial Track*. Citeseer, 2003. 62

[HJC+19] Armin Haller, Krzysztof Janowicz, Simon JD Cox, Maxime Lefrançois, Kerry Taylor, Danh Le Phuoc, Joshua Lieberman, Raúl García-Castro, Rob Atkinson, and Claus Stadler. The modular ssn ontology: A joint w3c and ogc standard specifying the semantics of sensors, observations, sampling, and actuation. *Semantic Web*, 10(1):9–32, 2019. 74, 98

[HKS06] Ian Horrocks, Oliver Kutz, and Ulrike Sattler. The even more irresistible sroiq. In *Proceedings of the Tenth International Conference on Principles of Knowledge Representation and Reasoning*, KR'06, page 57–67. AAAI Press, 2006. 50

[HM10] Terry Halpin and Tony Morgan. *Information modeling and relational databases*. Morgan Kaufmann, 2010. 62

[Hod01] Wilfrid Hodges. Elementary predicate logic. In *Handbook of philosophical logic*, pages 1–129. Springer, 2001. 47

[Hor51] Alfred Horn. On sentences which are true of direct unions of algebras. *The Journal of Symbolic Logic*, 16(1):14–21, 1951. 53

[HPSB+04] Ian Horrocks, Peter F Patel-Schneider, Harold Boley, Said Tabet, Benjamin Grosof, Mike Dean, et al. Swrl: A semantic web rule language combining owl and ruleml. *W3C Member submission*, 21(79):1–31, 2004. 53

[HRWL83] Frederick Hayes-Roth, Donald A Waterman, and Douglas B Lenat. Building expert system. 1983. 41

[HWCH08] S. Huang, T. Wu, H. Chu, and G. Hwang. A decision tree approach to conducting dynamic assessment in a context-aware ubiquitous learning environment. In *Fifth IEEE International Conference on Wireless, Mobile, and Ubiquitous Technology in Education (wmute 2008)*, pages 89–94, 2008. 61

[IRRH03] Jadwiga Indulska, Ricky Robinson, Andry Rakotonirainy, and Karen Henricksen. Experiences in using cc/pp in context-aware systems. In *International Conference on Mobile Data Management*, pages 247–261. Springer, 2003. 61

[JGZ17] Kamran Javed, Rafael Gouriveau, and Noureddine Zerhouni. State of the art and taxonomy of prognostics approaches, trends of prognostics applications and open issues towards maturity at different technology readiness levels. *Mechanical Systems and Signal Processing*, 94:214–236, 09 2017. 13, 70

[JSHL19] Eeva Järvenpää, Niko Siltala, Otto Hylli, and Minna Lanz. The development of an ontology for describing the capabilities of manufacturing resources. *Journal of Intelligent Manufacturing*, 2019. 73

[KCF12] Srdjan Komazec, Davide Cerri, and Dieter Fensel. Sparkwave: continuous schema-enhanced pattern matching over rdf data streams. In *Proceedings of the 6th ACM International Conference on Distributed Event-Based Systems*, pages 58–68, 2012. 56

[KF01] Michel Klein and Dieter Fensel. Ontology versioning on the semantic web. In *Proceedings of the First International Conference on Semantic Web Working*, SWWS'01, page 75–91, Aachen, DEU, 2001. CEUR-WS.org. 78, 79

[KK08] Kwang-Eun Ko and Kwee-Bo Sim. Development of context aware system based on bayesian network driven context reasoning method and ontology context modeling. In *2008 International Conference on Control, Automation and Systems*, pages 2309–2313, 2008. 61

[KK10] Barbara Korel and Simon Koo. A survey on context-aware sensing for body sensor networks. *Wireless Sensor Network*, 2:571–583, 01 2010. 61

[KMS+19] Evgeny Kharlamov, Gulnar Mehdi, Ognjen Savković, Guohui Xiao, Elem Güzel Kalaycı, and Mikhail Roshchin. Semantically-enhanced rule-based diagnostics for industrial internet of things: The sdrl language and case study for siemens trains and turbines. *Journal of Web Semantics*, 56:11–29, 2019. 71

[Kol91] Janet L Kolodneer. Improving human decision making through case-based decision aiding. *AI magazine*, 12(2):52–52, 1991. 57

[LBK15] Jay Lee, Behrad Bagheri, and Hung-An Kao. A cyber-physical systems architecture for industry 4.0-based manufacturing systems. *Manufacturing letters*, 3:18–23, 2015. 11, 66, 67

[Lei10] Frank Leistner. *Mastering organizational knowledge flow*. Wiley Online Library, 2010. 40

[LFK+14] Heiner Lasi, Peter Fettke, Hans-Georg Kemper, Thomas Feld, and Michael Hoffmann. Industry 4.0. *Business & Information Systems Engineering: The International Journal of WIRTSCHAFTSINFORMATIK*, 6(4):239–242, 2014. 66

[LL13] Angelica Nieto Lee and Jose Luis Martinez Lastra. Enhancement of industrial monitoring systems by utilizing context awareness. In *2013 IEEE International Multi-Disciplinary Conference on Cognitive Methods in Situation Awareness and Decision Support (CogSIMA)*, pages 277–284. IEEE, 2013. 77

[LSDS06] Séverin Lemaignan, A Siadat, Jean-Yves Dantan, and A Semenenko. MASON: A proposal for an ontology of manufacturing domain. 2006:195–200, 2006. 73

[Lut03] Carsten Lutz. Description Logics with Concrete Domains – A Survey. *Advances in Modal Logic*, 2003. 78, 79

[LXL+18] Shuangyin Liu, Longqin Xu, Qiucheng Li, Xuehua Zhao, and Daoliang Li. Fault diagnosis of water quality monitoring devices based on multi-class support vector machines and rule-based decision trees. *IEEE Access*, 6:22184–22195, 2018. 71

[LYZC18] Ruonan Liu, Boyuan Yang, Enrico Zio, and Xuefeng Chen. Artificial intelligence for fault diagnosis of rotating machinery: A review. *Mechanical Systems and Signal Processing*, 108:33–47, 2018. 70

[MB97] John McCarthy and Sasa Buvac. Formalizing context (expanded notes). 1997. 62

[MBG+03] Claudio Masolo, Stefano Borgo, Aldo Gangemi, Nicola Guarino, and Alessandro Oltramari. Wonderweb deliverable d18 ontology library. 2003. 107

[McC93] John McCarthy. Notes on formalizing context. 1993. 62

[MD03] Patrick Martin and Alain D'Acunto. Design of a production system: an application of integration product-process. *International Journal of Computer Integrated Manufacturing*, 16(7-8):509–516, 2003. 72

[MGGGGRG15] Víctor Martínez, Francisco Javier Gomez-Gil, Jaime Gomez-Gil, and Ruben Ruiz-Gonzalez. An artificial neural network based expert system fitted with genetic algorithms for detecting the status of several rotary components in agroindustrial machines using a single vibration signal. *Expert Systems with Applications*, 42(17-18):6433–6441, 2015. 71

[Mil08] Nick R Milton. *Knowledge technologies*, volume 3. Polimetrica sas, 2008. xiii, 7, 40, 41

[Min74] Marvin Minsky. A framework for representing knowledge. 1974. 47, 58

[MKB+05] Christopher Matheus, Mieczyslaw Kokar, Kenneth Baclawski, Jerzy Letkowski, Catherine Call, Michael Hinman, John Salerno, and Douglas Boulware. Sawa: An assistant for higher-level fusion and situation awareness. *Proceedings of SPIE - The International Society for Optical Engineering*, 5813, 03 2005. 76

[MKB+06] Christopher J Matheus, Mieczyslaw M Kokar, Kenneth Baclawski, Jerzy a Letkowski, Catherine Call, Michael Hinman, John Salerno, and Douglas Boulware. SAWA: An assistant for higher-level fusion and situation awareness. *Proceedings of SPIE*, 5813:75–85, 2006. 76

[MKP04] Marius Mikalsen and Anders Kofod-Petersen. Representing and reasoning about context in a mobile environment. In *Proceedings of the First International Workshop on Modeling and Retrieval of Context. CEUR Workshop Proceedings*, volume 114, pages 25–35, 2004. 62

[MN07] Marco Mamei and Radhika Nagpal. Macro programming through bayesian networks: Distributed inference and anomaly detection. In *Fifth Annual IEEE International Conference on Pervasive Computing and Communications (PerCom'07)*, pages 87–96. IEEE, 2007. 61

[Mob02] R Keith Mobley. *An introduction to predictive maintenance*. Elsevier, 2002. 11, 68

[Mok98] Joel Mokyr. The second industrial revolution, 1870-1914. *Storia dell'economia Mondiale*, 21945, 1998. 1, 27

[MR07] Boris Motik and Riccardo Rosati. A faithful integration of description logics with logic programming. In *IJCAI*, volume 7, pages 477–482, 2007. 50

[MUVHB14] Alessandro Margara, Jacopo Urbani, Frank Van Harmelen, and Henri Bal. Streaming the web: Reasoning over dynamic data. *Journal of Web Semantics*, 25:24–44, 2014. 56

[MVH+04] Deborah L McGuinness, Frank Van Harmelen, et al. Owl web ontology language overview. *W3C recommendation*, 10(10):2004, 2004. 8, 48, 50

[MWCD06] Cynthia Matuszek, Michael Witbrock, John Cabral, and John DeOliveira. An introduction to the syntax and content of cyc. *UMBC Computer Science and Electrical Engineering Department Collection*, 2006. 75

[NA06] Noy N. and Rector A. Defining n-ary relations on the semantic web. In *W3C Working Group Note*, April 2006. 78, 79

[Nat98] James B Nation. Notes on lattice theory, 1998. 124

[NB18] David Lira Nuñez and Milton Borsato. Ontoprog: An ontology-based model for implementing prognostics health management in mechanical machines. *Advanced Engineering Informatics*, 38:746–759, 2018. 13, 71

[NK04] Natalya F Noy and Michel Klein. Ontology evolution: Not the same as schema evolution. *Knowledge and information systems*, 6(4):428–440, 2004. 4, 29

[Noy01] Natasha Noy. Ontology development 101: A guide to creating your first ontology. 2001. 43, 45

[NP01] Ian Niles and Adam Pease. Towards a standard upper ontology. In *Proceedings of the international conference on Formal Ontology in Information Systems-Volume 2001*, pages 2–9, 2001. 47

[ÖA97] Pinar Öztürk and Agnar Aamodt. Towards a model of context for case-based diagnostic problem solving. *Context*, 99:198–208, 1997. 62

[OBGM11] Mohamed-Zied Ouertani, Salah Baïna, Lilia Gzara, and Gérard Morel. Traceability and management of dispersed product knowledge during design and manufacturing. *Computer-Aided Design*, 43(5):546–562, 2011. 40

[OD11] Martin J. O'Connor and Amar K. Das. A method for representing and querying temporal information in owl. In Ana Fred, Joaquim Filipe, and Hugo Gamboa, editors, *Biomedical Engineering Systems and Technologies*, pages 97–110, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg. 101

[ODSM+08] Martin O'Connor, Ravi D. Shankar, Mark Musen, Amar Das, and Csongor Nyulas. The swrlapi: A development environment for working with swrl rules., 01 2008. 96

[PC09] Panagiotis Papadopoulos and Liana Cipcigan. Wind turbines' condition monitoring: an ontology model. In *2009 International Conference on Sustainable Power Generation and Supply*, pages 1–4. IEEE, 2009. 72, 94

[PDT12] Hervé Panetto, Michele Dassisti, and Angela Tursi. Onto-pdm: Product-driven ontology for product data management interoperability within manufacturing process environment. *Advanced Engineering Informatics*, 26(2):334–348, 2012. 72

[PDTC07] Peter Plessers, Olga De Troyer, and Sven Casteleyn. Understanding ontology evolution: A change detection approach. *Journal of Web Semantics*, 5(1):39–49, 2007. 4, 29

[PDTPH11] Danh Le Phuoc, Minh Dao-Tran, Josiane Xavier Parreira, and Manfred Hauswirth. A native and adaptive approach for unified processing of linked streams and linked data. In *International Semantic Web Conference*, 2011. 56

[PDZ10] Ying Peng, Ming Dong, and Ming Jian Zuo. Current status of machine prognostics in condition-based maintenance: a review. *The International Journal of Advanced Manufacturing Technology*, 50(1-4):297–313, 2010. 13, 70

[PH12] M. Perry and J. Herring. Ogc geosparql - a geographic query language for rdf data. *OGC Implementation Standard*, Sept, 2012. 99

[POC11] Han-Saem Park, Keunhyun Oh, and Sung-Bae Cho. Bayesian network-based high-level context recognition for mobile context sharing in cyber-physical system. *International Journal of Distributed Sensor Networks*, 7(1):650387, 2011. 61

[PVGPSF14] María Poveda-Villalón, Asunción Gómez-Pérez, and Mari Carmen Suárez-Figueroa. OOPS! (OntOlogy Pitfall Scanner!): An On-line Tool for Ontology Evaluation. *International Journal on Semantic Web and Information Systems (IJSWIS)*, 10(2):7–34, 2014. 109

[PZCG13] Charith Perera, Arkady Zaslavsky, Peter Christen, and Dimitrios Georgakopoulos. Context aware computing for the internet of things: A survey. *IEEE communications surveys & tutorials*, 16(1):414–454, 2013. 60

[RBBSD11] Ana Rossello-Busquet, Lukasz J Brewka, José Soler, and Lars Dittmann. Owl ontologies and swrl rules applied to energy management. In *2011 UkSim 13th International Conference on Computer Modelling and Simulation*, pages 446–450. IEEE, 2011. 53

[RCC92] David A. Randell, Zhan Cui, and Anthony G. Cohn. A Spatial Logic based on Regions and Connection. *3rd International Conference On Knowledge Representation And Reasoning*, pages 165–176, 1992. 99

[Rie88] C Riesbeck. An interface for case-based knowledge acquisition. In *Proceedings of the DARPA Case-Based Reasoning Workshop*, pages 312–326. Morgan Kaufmann Publishers, Inc, 1988. 57

[Rif11] Jeremy Rifkin. The third industrial revolution. how lateral power is transforming energy, the economy, and the world. 2011. 1, 27

[Row07] Jennifer Rowley. The wisdom hierarchy: representations of the dikw hierarchy. *Journal of Information Science*, 33(2):163–180, 2007. 3, 28, 29

[SAA+00] G Schreiber, H Akkermans, A Anjewierden, R De Hoog, N R Shadbolt, and Bob Wielinga. *Knowledge Engineering and Management: The CommonKADS Methodology*, volume 99. 2000. 89

[SAB03] B Samanta and KR Al-Balushi. Artificial neural network based fault diagnostics of rolling element bearings using time-domain features. *Mechanical systems and signal processing*, 17(2):317–328, 2003. 70

[SAW94a] Bill Schilit, Norman Adams, and Roy Want. Context-aware computing applications. In *1994 First Workshop on Mobile Computing Systems and Applications*, pages 85–90. IEEE, 1994. 61

[SAW94b] Bill N. Schilit, N. Adams, and R. Want. Context-aware computing applications. In *IEEE Workshop on Mobile Computing Systems and Applications*, pages 85–90, 1994. 9, 60

[SBG99] Albrecht Schmidt, Michael Beigl, and Hans-W Gellersen. There is more to context than location. *Computers & Graphics*, 23(6):893–901, 1999. 62

[SCDV+19] Heiner Stuckenschmidt, S Ceri, Emanuele Della Valle, Frank Harmelen, and P Milano. Towards expressive stream reasoning. *Proceedings of the Dagstuhl Seminar on Semantic Aspects of Sensor Networks*, 03 2019. xiii, 3, 29, 55, 114

[SE05] Pavel Shvaiko and Jérôme Euzenat. A survey of schema-based matching approaches. In Stefano Spaccapietra, editor, *Journal on Data Semantics IV*, pages 146–171, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg. 107

[SHS08] Amit Sheth, Cory Henson, and Satya S. Sahoo. Semantic Sensor Web. *IEEE Internet Computing*, 12(4):78–83, July 2008. 74

[SKB05] Barry Smith, Anand Kumar, and Thomas Bittner. *Basic Formal Ontology for Bioinformatics*. IFOMIS Reports, 2005. 47

[Sla91] Stephen Slade. Case-based reasoning: A research paradigm. *AI magazine*, 12(1):42–42, 1991. 41

[SLP04] Thomas Strang and Claudia Linnhoff-Popien. A Context Modeling Survey. In *Workshop on Advanced Context Modelling, Reasoning and Management, UbiComp 2004 - The Sixth International Conference on Ubiquitous Computing*, volume Workshop o, pages 1–8, 2004. 63

[SLPF03a] Thomas Strang, Claudia Linnhoff-Popien, and Korbinian Frank. Applications of a context ontology language. *Proceedings of SoftCOM 2003*, pages 14–18, 2003. 62

[SLPF03b] Thomas Strang, Claudia Linnhoff-Popien, and Korbinian Frank. Cool: A context ontology language to enable contextual interoperability. In *IFIP International Conference on Distributed Applications and Interoperable Systems*, pages 236–247. Springer, 2003. 62

[SMASC09] Cesar Sanín, Leonardo Mancilla-Amaya, Edward Szczerbicki, and Paul CayfordHowell. Application of a multi-domain knowledge structure: the decisional dna. In *Intelligent systems for knowledge management*, pages 65–86. Springer, 2009. 58, 59

[SMB+17] C. Santos, A. Mehrsai, A.C. Barros, M. Araújo, and E. Ares. Towards industry 4.0: an overview of european strategic roadmaps. *Procedia Manufacturing*, 13:972 – 979, 2017. Manufacturing Engineering Society International Conference 2017, MESIC 2017, 28-30 June 2017, Vigo (Pontevedra), Spain. 1, 27

[SMMS02] Ljiljana Stojanovic, Alexander Maedche, Boris Motik, and Nenad Stojanovic. User-driven ontology evolution management. In *International Conference on Knowledge Engineering and Knowledge Management*, pages 285–300. Springer, 2002. 4, 29

[SMZ14] Abdenour Soualhi, Kamal Medjaher, and Noureddine Zerhouni. Bearing health monitoring based on hilbert–huang transform, support vector machine, and regression. *IEEE Transactions on Instrumentation and Measurement*, 64(1):52–62, 2014. 70

[Sow99] John F Sowa. *Knowledge representation: logical, philosophical and computational foundations*. Brooks/Cole Publishing Co., 1999. 7, 40

[SS09] Cesar Sanin and Edward Szczerbicki. Experience-based knowledge representation: Soeks. *Cybernetics and Systems: an international journal*, 40(2):99–122, 2009. 58

[SS14] Meik Schlechtingen and Ilmar Ferreira Santos. Wind turbine condition monitoring based on scada data using normal behavior models. part 2: Application examples. *Applied Soft Computing*, 14:447 – 460, 2014. 71, 125

[ŠS19] Dušan Šormaz and Arkopaul Sarkar. SIMPM – Upper-level ontology for manufacturing process plan network generation. *Robotics and Computer-Integrated Manufacturing*, 2019. 73

[SSA13] Meik Schlechtingen, Ilmar Ferreira Santos, and Sofiane Achiche. Wind turbine condition monitoring based on scada data using normal behavior models. part 1: System description. *Applied Soft Computing*, 13(1):259 – 270, 2013. 71, 125

[SST07] Cesar Sanin, Edward Szczerbicki, and Carlos Toro. An owl ontology of set of experience knowledge structure. *J. UCS*, 13(2):209–223, 2007. 58, 59

[STH+12] Cesar Sanin, Carlos Toro, Zhang Haoxi, Eider Sanchez, Edward Szczerbicki, Eduardo Carrasco, Wang Peng, and Leonardo Mancilla-Amaya. Decisional dna: A multi-technology shareable knowledge structure for decisional experience. *Neurocomputing*, 88:42–53, 2012. 58, 59

[Sto04] Ljiljana Stojanovic. Methods and tools for ontology evolution. 2004. 4, 29

[Swa01] Laura Swanson. Linking maintenance strategies to performance. *International journal of production economics*, 70(3):237–244, 2001. 11, 68

[SWG17] Bernard Schmidt, Lihui Wang, and Diego Galar. Semantic framework for predictive maintenance in a cloud environment. In *10th CIRP Conference on Intelligent Computation in Manufacturing Engineering, CIRP ICME'16, Ischia, Italy, 20-22 July 2016*, volume 62, pages 583–588. Elsevier, 2017. 71

[SWWP10] J Schwarzenbach, L Wilkinson, M West, and M Pilling. Mapping the remote condition monitoring architecture. *Research Programme. Rail Safety and Standards Boards (RSSB) LTD. RSSB Core Report*, 2010. 72

[TB09] Jonas Tappolet and Abraham Bernstein. Applied temporal RDF: Efficient temporal querying of rdf data with SPARQL. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2009. 78, 79

[TBDV+17] Riccardo Tommasini, Pieter Bonte, Emanuele Della Valle, Erik Mannens, Filip Turck, and Femke Ongenae. Towards ontology-based event processing. pages 115–127, 02 2017. 125

[TBP15] Carlos Toro, Iñigo Barandiaran, and Jorge Posada. A Perspective on Knowledge Based and Intelligent Systems Implementation in Industrie 4.0. *Procedia Computer Science*, 60:362 – 370, 2015. Knowledge-Based and Intelligent Information & Engineering Systems 19th Annual Conference, KES-2015, Singapore. 86

[TS16] Lane Thames and Dirk Schaefer. Software-defined cloud manufacturing for industry 4.0. *Procedia CIRP*, 52:12 – 17, 2016. The Sixth International Conference on Changeable, Agile, Reconfigurable and Virtual Production (CARV2016). 67

[TSC+12] Carlos Toro, Eider Sanchez, Eduardo Carrasco, Leonardo Mancilla-Amaya, Cesar Sanín, Edward Szczerbicki, Manuel Graña, Patricia Bonachela, Carlos Parra, Gloria Bueno, et al. Using set of experience knowledge structure to extend a rule set of clinical decision support system for alzheimer's disease diagnosis. *Cybernetics and Systems*, 43(2):81–95, 2012. 57

[TVG+10] Ying Tan, Mehmet C. Vuran, Steve Goddard, Yue Yu, Miao Song, and Shangping Ren. A concept lattice-based event model for cyber-physical systems. In *Proceedings of the 1st ACM/IEEE International Conference on Cyber-Physical Systems*, ICCPS '10, page 50–60, New York, NY, USA, 2010. Association for Computing Machinery. 125

[UG96] Michael Uschold and Michael Grüninger. Ontologies: Principles, methods and applications. *The Knowledge Engineering Review*, 11, 01 1996. xiii, 45, 46, 96

[UPML15] Mohammad Kamal Uddin, Juha Puttonen, and Jose Luis Martinez Lastra. Context-sensitive optimisation of the key performance indicators for fms. *International Journal of Computer Integrated Manufacturing*, 28(9):958–971, 2015. 77

[UYC+11] Zahid Usman, Robert Ian Marr Young, Nitishal Chungoora, Claire Palmer, Keith Case, and Jenny Harding. A manufacturing core concepts ontology for product lifecycle interoperability. In *Lecture Notes in Business Information Processing*, 2011. 72

[W3C] W3C. Composite capabilities / preferences profile (cc/pp). http://www.w3.org/Mo-bile/CCPP. 61

[WA94] Ian Watson and Salha Abdullah. Developing case-based reasoning systems: a case study in diagnosing building defects. In *IEE Colloquium on Case Based Reasoning: Prospects for Applications (Digest No. 1994/057)*, pages 1–1. IET, 1994. 57

[Wan16] K Wang. Intelligent predictive maintenance (ipdm) system–industry 4.0 scenario. *WIT Transactions on Engineering Sciences*, 113:259–268, 2016. 12, 69

[WDD94] John H Williams, Alan Davies, and Paul R Drake. *Condition-based maintenance and machine diagnostics*. Springer Science & Business Media, 1994. 11, 68

[WDR06] Eugene Wu, Yanlei Diao, and Shariq Rizvi. High-performance complex event processing over streams. In *Proceedings of the 2006 ACM SIGMOD International Conference on Management of Data*, SIGMOD '06, page 407–418, New York, NY, USA, 2006. Association for Computing Machinery. 124

[WDTP04] X.H. Wang, Da Qing Zhang, Tao Gu, and H.K. Pung. Ontology based context modeling and reasoning using OWL. *IEEE Annual Conference on Pervasive Computing and Communications Workshops, 2004. Proceedings of the Second*, pages 18–22, 2004. xiii, 43, 62, 76, 77

[WF06] Chris Welty and Richard Fikes. A reusable ontology for fluents in owl. In *Proceedings of the 2006 Conference on Formal Ontology in Information Systems: Proceedings of the Fourth International Conference (FOIS 2006)*, page 226–236, NLD, 2006. IOS Press. 79

[Wil05] Rudolf Wille. *Formal Concept Analysis as Mathematical Theory of Concepts and Concept Hierarchies*, pages 1–33. Springer Berlin Heidelberg, Berlin, Heidelberg, 2005. 125

[WM94] Ian Watson and Farhi Marir. Case-based reasoning: A review. *Knowledge Engineering Review*, 9(4):327–354, 1994. 57

[WSJ17] M. Wollschlaeger, T. Sauter, and J. Jasperneite. The future of industrial communication: Automation networks in the era of the internet of things and industry 4.0. *IEEE Industrial Electronics Magazine*, 11(1):17–27, 2017. 10, 66

[WTL+17] Jiafu Wan, Shenglong Tang, Di Li, Shiyong Wang, Chengliang Liu, Haider Abbas, and Athanasios V Vasilakos. A manufacturing big data solution for active preventive maintenance. *IEEE Transactions on Industrial Informatics*, 13(4):2039–2047, 2017. 11, 68

[WWLZ16] Shiyong Wang, Jiafu Wan, Di Li, and Chunhua Zhang. Implementing smart factory of industrie 4.0: An outlook. *International Journal of Distributed Sensor Networks*, 12(1):3159805, 2016. 1, 27

[XLC+18] Feixiang Xu, Xinhui Liu, Wei Chen, Chen Zhou, and Bingwei Cao. Ontology-based method for fault diagnosis of loaders. *Sensors*, 18(3):729, 2018. 71, 94

[XYWV12] Feng Xia, Laurence T. Yang, Lizhe Wang, and Alexey Vinel. Internet of things. *International Journal of Communication Systems*, 25(9):1101–1102, 2012. 66

[YK15] S. Yin and O. Kaynak. Big data for modern industry: Challenges and trends [point of view]. *Proceedings of the IEEE*, 103(2):143–146, 2015. 67

[YL06] Stephen S Yau and Junwei Liu. Hierarchical situation modeling and reasoning for pervasive computing. In *The Fourth IEEE Workshop on Software Technologies for Future Embedded and Ubiquitous Systems, and the Second International Workshop on Collaborative Computing, Integration, and Assurance (SEUS-WCCIA'06)*, pages 6–pp. IEEE, 2006. 76

[ZM15] Cecilia Zanni-Merk. Krem: A generic knowledge-based framework for problem solving in engineering. In *KEOD*, pages 381–388, 2015. 7, 57

[ZMS19] Cecilia Zanni-Merk and Edward Szczerbicki. Building collective intelligence through experience: a survey on the use of the krem model. *Journal of Intelligent & Fuzzy Systems*, 37(6):7141–7153, 2019. xiii, 42

[ZON15] J Zhu, Soh-Khim Ong, and Andrew YC Nee. A context-aware augmented reality assisted maintenance system. *International Journal of Computer Integrated Manufacturing*, 28(2):213–225, 2015. 77

[ZTL15] K. Zhou, Taigang Liu, and Lifeng Zhou. Industry 4.0: Towards future industrial opportunities and challenges. In *2015 12th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD)*, pages 2147–2152, 2015. 11, 68

[ZYC+19] Rui Zhao, Ruqiang Yan, Zhenghua Chen, Kezhi Mao, Peng Wang, and Robert X Gao. Deep learning and its applications to machine health monitoring. *Mechanical Systems and Signal Processing*, 115:213–237, 2019. 70

[ZZWH19] Shen Zhang, Shibo Zhang, Bingnan Wang, and Thomas G Habetler. Machine learning and deep learning algorithms for bearing fault diagnostics-a comprehensive review. *arXiv preprint arXiv:1901.08247*, 2019. 70